Noun–Noun Compound Analysis: A Holistic Perspective

Murhaf Fares

Department of Informatics
Faculty of Mathematics and Natural Sciences
University of Oslo

Thesis submitted for the degree of Philosophiæ Doctor
2019

Series of dissertations submitted to the
Faculty of Mathematics and Natural Sciences, University of Oslo
No. 2159

ISSN 1501-7710

Cover: Hanne Baadsgaard Utigard.
Print production: Reprosentralen, University of Oslo.

إلى كل من غنى «جنّة جنّة جنّة» ...

فحين تصيرُ الحيَاةُ طبيعيّةً
سوف نحزن كَالآخرين لَأشيَاءَ شخصيّةٍ
خَبَّأتهَا عَنَاوينُ كُبرَى،
فلم نَنتبِه لنزيف الجُروح الصغيرةِ فينَا.
— محمود درويش

Kanskje du spør i angst,
udekket, åpen:
hva skal jeg kjempe med,
hva er mitt våpen?
— Nordahl Grieg

Acknowledgments

First and foremost, I am thankful to my supervisors, Stephan Oepen and Erik Velldal. Your genuine interest in my work has never faded throughout the years, even when we had to stare at too many numbers trying to discover some patterns or discuss which `ARG` is the 'right' one for a given noun–noun compound. It is safe to say that you provided me with the essential scientific and methodological tools—including hedges—to carry out my research. You respected my opinion and decisions at different important crossroads; from experimental design to—well—typographical considerations. For all of the above, I consider myself fortunate to have you as supervisors.

I am thankful to Martha Palmer, and the NLP Group at CU Boulder, for hosting me in the first half of 2017 and many fruitful discussions on semantic representation frameworks and neural networks.

Many friends and colleagues, perhaps unknowingly, have contributed to making life in the dark days—literally and figuratively—brighter. To the best friend in writing and otherwise, Emanuele Lapponi, thank you for many consoling (and writing) sessions; you have been a true friend since my first day at IFI. To the current and former members of the Language Technology Group, Andrei, Arne, Eivind, Elisabeth, Farhad, Jan Tore, Jeremy, Lilja, Samia and Vinit, thank you for the feedback and support on different occasions and for creating a friendly work environment. To the ESC10 group, our coffee breaks made it easier to get through the day, even though I was not the most 'devout' espresso drinker. To the SPACE team, believe it or not, our work together somehow made me more determinant to finish my PhD and kept me sane throughout those years.

To my parents, Maha and Mahmoud, if I manage to make you proud, know that this would have never been possible without your limitless trust and support. I am privileged to have loving parents who worked hard to pave the way for me and be where I am today. To my wonderful brothers, Anas and Munes, you continue to surround me with love and care even from across the ocean.

حتَى تحترق النجوم ... ,Zeina

Contents

1 Introduction **1**
- 1.1 Motivation and Overview . . . 3
- 1.2 Contributions . . . 6
  - 1.2.1 Limitations . . . 8
- 1.3 Thesis Outline . . . 9

2 Noun–Noun Compounds in Linguistics and NLP **13**
- 2.1 Defining Noun–Noun Compounds . . . 14
  - 2.1.1 A Multitude of Studies and Names . . . 14
  - 2.1.2 Finite vs. Infinite . . . 15
  - 2.1.3 Deictic, Novel and Established Compounds . . . 17
  - 2.1.4 Endocentric vs. Exocentric . . . 18
  - 2.1.5 Nominalizations vs. Root Nominals . . . 19
  - 2.1.6 Theoretical vs. Computational . . . 20
  - 2.1.7 A Preliminary Definition . . . 22
- 2.2 Noun–Noun Compound Analysis . . . 22
  - 2.2.1 Identification . . . 23
  - 2.2.2 Bracketing . . . 24
  - 2.2.3 Constituent Sense Disambiguation . . . 25
  - 2.2.4 Interpretation . . . 26
  - 2.2.5 Compositionality . . . 27
- 2.3 A Selection of Studies: Literature Review . . . 28
  - 2.3.1 Lauer (1995) . . . 28
  - 2.3.2 Girju et al. (2005) . . . 29
  - 2.3.3 Kim and Baldwin (2013) . . . 30
  - 2.3.4 Ó Séaghdha and Copestake (2013) . . . 32
  - 2.3.5 Tratz and Hovy (2010) . . . 34
  - 2.3.6 Shwartz and Waterson (2018) . . . 35

2.4 Conclusion 36

3 Annotation of Noun–Noun Compounds and Beyond 39

3.1 Introduction 39

3.2 Compound-Specific Datasets 41

3.2.1 Ó Séaghdha and Copestake (2007) 43

3.2.2 Kim and Baldwin (2008) 45

3.2.3 Tratz and Hovy (2010) 47

3.3 Contrastive Analysis 50

3.4 Linguistic Annotation Frameworks 52

3.4.1 NomBank 53

3.4.2 PCEDT 2.0 55

3.4.3 Other Resources 57

3.5 Conclusion 59

4 Resource Creation 61

4.1 Introduction and Motivation 61

4.2 Overview 63

4.3 Compound Identification 64

4.3.1 Motivation and Past Work 64

4.3.2 Compound Identification Strategies 66

4.3.3 Syntax-Based Identification 67

4.3.4 Results and Discussion 68

4.3.5 Refined Identification Method 71

4.4 Noun–Noun Compound Bracketing 74

4.4.1 Data and Results 75

4.5 Semantic Relations 78

4.5.1 Data, Results and Reflections 79

4.5.2 Correspondence between PCEDT and NomBank 83

4.5.3 Type vs. Token Semantics 87

4.5.4 Compound Interpretation Dataset 90

4.6 Conclusion 91

5 Embeddings and Similarity-Based Classification 95
5.1 Introduction and Motivation . . . 96
5.2 Background . . . 97
5.2.1 Word Embeddings . . . 99
5.3 Word Embedding Models . . . 101
5.4 Linguistic Regularities in Embeddings . . . 103
5.5 Vector Arithmetic for Compounds . . . 105
5.5.1 Experimental Results . . . 107
5.5.2 K-Nearest Neighbors . . . 109
5.6 Conclusion . . . 111

6 Neural Classification 113
6.1 Background: Dima and Hinrichs (2015) . . . 114
6.1.1 Replicating Dima and Hinrichs (2015) . . . 118
6.2 Neural Classification Experiments . . . 121
6.3 Effect of Word Embeddings . . . 122
6.3.1 Text Pre-Processing . . . 123
6.3.2 Vector Dimensionality and Fine-Tuning . . . 126
6.3.3 Size of Training Data and Fine-Tuning . . . 130
6.4 Model Architecture . . . 135
6.5 Model Hyperparameters . . . 140
6.5.1 Sensitivity Analysis . . . 141
6.5.2 Random Search . . . 144
6.6 WordNet Features . . . 148
6.7 Training Data . . . 152
6.8 Conclusion . . . 158

7 Transfer and Multi-Task Learning 163
7.1 Introduction and Motivation . . . 164
7.2 Terminology and Definitions . . . 165
7.2.1 TL vs. MTL . . . 168
7.2.2 Related Work . . . 169
7.3 TL & MTL for Compound Interpretation . . . 173
7.4 Experimental Setup . . . 175
7.4.1 Single-Task Learning Model . . . 175
7.4.2 Transfer Learning Models . . . 176
7.4.3 Multi-Task Learning Models . . . 177

7.5 Experimental Results . . . 178
7.5.1 Evaluation on the Development Split . . . 178
7.5.2 Evaluation on the Test Split . . . 183
7.5.3 What Happened to AIM? . . . 191
7.6 Generalization on Unseen Compounds . . . 191
7.7 Performance Stability . . . 195
7.8 Conclusion . . . 197

8 Summary and Concluding Remarks **201**

References **209**

A Noun Compound Relations **225**
A.1 Nastase and Szpakowicz (2003) . . . 225

B NomBank and PCEDT Relations **227**
B.1 NomBank Relations . . . 227
B.2 PCEDT Functors . . . 228

C Model Hyperparameters **231**

Chapter 1

Introduction

Noun–noun compound analysis in natural language processing (NLP) is a compound problem consisting of several tasks that collectively deal with the syntax and semantics of this linguistic construction. Past work on compound analysis has typically focused—to varying degrees—on five tasks, namely compound (1) identification, (2) bracketing, (3) compositionality prediction, (4) sense disambiguation and (5) interpretation. We will use the following example paragraph to briefly introduce these tasks, using *italics* to highlight compound instances.

> The ultimate goal of this thesis is to distinguish between the meaning of *snake oil product*, *hair product* and *meat product* among other types of noun–noun compounds. Some of the experiments presented in this thesis make use of the *dot product*.

The task of compound identification, simply, refers to identifying noun–noun compounds in written text or speech. In compound bracketing, the goal is to analyze the internal syntactic structure of noun–noun compounds. For example, *snake oil product* is analyzed as a left-branching compound, i.e. [[snake oil] product]. The task of compound compositionality prediction seeks to determine the degree of compositionality through relating the meaning of the compound as a whole to the meaning of its constituents. For example, the predominant contemporary use of *snake oil product* is non-compositional because its meaning cannot be immediately derived from the individual meanings of its constituents.[1] Compound sense disambiguation is concerned with disambiguating the meaning of polysemous constituents in noun–noun

[1] A *snake oil product* is a fraudulent product of little real worth. However, in traditional Chinese medicine the meaning of *snake oil* is compositional; i.e. oil extracted from snakes.

compounds. For example, the constituent *product* in *dot product* refers to the mathematical concept of multiplication, but in *consumer product* it refers to an article or substance for sale. Compound interpretation is the task of determining the thematic relations holding between the constituent parts of noun–noun compounds. In layman's terms, this task amounts to determining that *meat product* is a food product that *contains* meat whereas *hair product* does not contain hair but is used *for* hair care.

In this thesis, we present a study of English noun–noun compound analysis that takes a holistic perspective on the problem. The holistic nature of our work manifests in three respects. First, of the five compound analysis tasks introduced above, we focus primarily on compound interpretation and identification, but we also create a resource for compound bracketing and reflect on (the need for) compound sense disambiguation in our work. Second, we part company with past NLP studies on compound analysis and resituate the problem within general-purpose whole-sentence meaning representation frameworks. Specifically, we introduce a new approach (and a resource) that derives the semantic interpretation of noun–noun compounds from linguistic resources that represent the semantics of phrasal or sentential structures, in contradistinction to the more isolated, compound-centric perspective of much past work. Third, we empirically determine the utility of distributional semantic models as well as of neural networks to classify the relations holding between the compound constituents. Our experimental setup is systemically varied to account for different properties of the models we use, in isolation and in combination.

Overall, this thesis stands at the intersection of many of the recent, and not-so-recent, developments in NLP. In addition to investigating several properties of word embeddings and neural networks, we also experiment with transfer and multi-task learning for compound interpretation—two machine learning techniques that have recently drawn much attention in NLP research. Further, we use manually annotated, well-established meaning representation resources to shed new light on noun–noun compounds, and call for critical reflections on some of the long-held views pertaining to their analysis in NLP.

Before we delve into our theoretical and experimental study of noun–noun compounds—first things first—in the next few pages, we motivate the subject matter of this project, highlight some of its main contributions and limitations and, finally, lay out a road map for the rest of the thesis.

1.1 Motivation and Overview

Frequent, productive and semantically unpredictable; these are three properties of noun–noun compounds which make them ubiquitous, scientifically interesting and challenging. Their frequency has been attested, and quantified in numerous studies; for example, Ó Séaghdha (2008) and Baldwin and Tanaka (2004) report, respectively, that 3% of the words in the British National Corpus (Burnard, 2000) and 3.9% of the Reuters Corpus (Rose et al., 2002) are part of noun–noun compounds. In language production, we often form new compounds on the spot, among other reasons to fulfill some informational need such as referencing objects or entities that do not have lexicalized names. Such an open-ended ability to create new compounds—by simply combining pairs of nouns—is what makes them productive, and perhaps semantically unpredictable also. The following are just a few examples of the compounds (headed by the noun *pill*) that can be found in the Corpus of Contemporary American English (COCA; Davies, 2009): *abortion pill*, *sleeping pill*, *contraception pill*, *poison pill*, *chill pill* and *horse pill*. While most, if not all, of these compounds are readily interpretable by humans, computational models would require more than just the surface form information to interpret them and represent the different meanings they convey.[2]

Noun–noun compounds have been the focus of much work, both in theoretical linguistics (Li, 1972; Downing, 1977; Levi, 1978; Warren, 1978; Finin, 1980; inter alios) and natural language processing (Lauer, 1995; Nakov, 2007; Ó Séaghdha & Copestake, 2009; Tratz & Hovy, 2010; Kim & Baldwin, 2013; Dima & Hinrichs, 2015; inter alios). However, despite this extensive literature on compound analysis (and compound interpretation in particular), past studies do not come close to agreeing on how compounds ought to be interpreted (or analyzed more generally). Indeed, it would be no exaggeration to say that some of the competing views on compound interpretation in theoretical linguistics vary from claiming that only nine relations are sufficient to interpret compounds (Levi, 1978) to simply stating that no finite set of relations is adequate to represent the semantics of noun–noun compounds (Downing, 1977).

The vast majority of past NLP studies on compound interpretation invent

[2]In fact, the meaning of some of these compounds might not be as obvious for humans as well. For example, *poison pill* refers to a defense strategy against a hostile takeover by other companies and *horse pill* is a pill that is too large to swallow.

(or sometimes reuse) what can be described as 'ad hoc' taxonomies of relations. That is not to say that these taxonomies are generally uninformed by linguistic theory, but they often use very specialized (or overly abstract) relations that are not easily integrated with other NLP sub-tasks. Furthermore, many of the past studies tend to start anew and create their own distinct taxonomies of relations that annotate distinct sets of compounds, which complicates the comparison between them. Consequently, not only do we observe a tendency to isolate compound analysis from other NLP tasks, but also limited cross-fertilization among the compound analysis studies themselves. With the aforementioned points in mind, in this work, we position compound analysis within the realm of meaning representation frameworks, unspecific to compounds, which—we argue—is an approach better suited to integrate compound analysis into other tasks and facilitate cross-framework comparison, as will become clearer later in this thesis.

At the heart of our holistic approach to compound analysis is a dataset that links noun–noun compounds to broader general-purpose meaning representation frameworks. To create such a dataset, we first survey the existing compound datasets as well as some widely used whole-sentence meaning representation frameworks in Chapter 3. Then, in Chapter 4, we set out to derive a new compound dataset from the combination of five resources that annotate the same text with various types of syntactic and semantic information. The process of creating the dataset, however, forces us to think about even more foundational aspects of compound analysis. For example, the first step in creating a dataset of compounds is to identify them in running text (i.e. the compound identification task). Past studies rely heavily on an identification method which inherently leads to identifying false positive examples; i.e. it identifies the mere occurrence of noun sequences as compounds. Hence, in Chapter 4, we take a step back and start from the task of compound identification by proposing a new method to identify compounds using syntactic structure.

By deriving our dataset from multiple linguistic resources, we effectively revisit critical questions related to these resources as well as compound analysis in NLP. For example, we use the dataset itself as a means to reflect on, and when possible evaluate, 'cross-framework annotation agreement' based on multiple, parallel syntactic and semantic annotations of noun–noun compounds in our dataset. Additionally, we probe the question of whether the semantics of noun–noun compounds is best approached in a type-based

or token-based perspective, which also leads us to reflect on the annotations of the linguistic resources themselves.

In the second half of this thesis, we put our dataset to use and empirically investigate the utility of relatively recent machine learning models and techniques (e.g. word embeddings and neural networks) for compound interpretation. Our ultimate goal is to be able to computationally predict the semantic relations holding between the compound constituents. In order to come closer to achieving such a goal, however, we first need to consider—and ideally understand—the interplay between the machine learning models we use and compound interpretation (as defined in our dataset, at least). Despite their success and popularity in much current NLP, word embeddings and neural networks remain comparatively less explored in the context of compound interpretation in contrast to the more 'traditional' machine learning methods such as support-vector machines (SVMs). Therefore, in Chapters 5, 6 and 7, we systematically study the impact of different properties and (hyper)parameters of word embeddings and neural networks on compound interpretation. Owing to the parallel annotation in our dataset (which will be explained in some detail in the following section), we also experiment with two machine learning strategies that currently receive a lot of attention in neural NLP and other fields, namely transfer and multi-task learning. These two learning strategies are often used to improve a neural model's generalization by "leveraging the domain-specific information contained in the training signals of related tasks" (Caruana, 1997, p. 41).

Research Objectives

Drawing upon an extensive literature review as well as the recent developments in the field, this doctoral project started with three broad goals in mind. The first goal was to investigate how noun–noun compounds are analyzed (or rather, interpreted) in general-purpose meaning representation frameworks. In other words, we wanted to examine how the semantics of compounds would be represented if one is to break away from the datasets that somehow isolate an otherwise integral linguistic construction from whole-sentence semantics. Second, motivated by the success of distributional semantic models in other NLP tasks, we intended to assess to what degree such models lend themselves to predicting the semantics of noun–noun compounds. Among other things, we were interested in determining the impact of several properties of word

embedding models (when used as input representations to neural networks) on the compound interpretation problem. Third, being a doctoral project in NLP, training machine learning models for compound interpretation is a somewhat obvious goal. Therefore, in light of their widespread use as an effective 'tool' for a range of NLP tasks, we set out to evaluate the ability of neural network models to learn and represent the semantics of noun–noun compounds.

Over the course of the project, each of these three general objectives, inevitably, led us to pursue more specific and methodological questions related to compound analysis as well as the linguistic resources and machine learning techniques we use.

1.2 Contributions

In the following, we group the contributions of this thesis under different overarching themes, but several of these at times interact and overlap, of course.

Back to the Roots: In Chapter 2, we conduct a systematic review of compound analysis in theoretical linguistics and make an attempt at discerning the (dis)similarities between the computational and theoretical approaches to compound analysis. Even though our efforts in this avenue remain at the theoretical level, we believe they can contribute to more systematic, linguistically-informed compound analysis in NLP. Along similar lines, in Chapter 3, we present a survey of the existing compound interpretation datasets and closely examine and contrast three datasets, viz. the ones by Ó Séaghdha and Copestake (2007), Kim and Baldwin (2008) and Tratz (2011).

Building Bridges: One of the main contributions of this thesis is the creation of a new noun–noun compound dataset with bracketing and semantic annotations. We use four resources that annotate (parts of) the Wall Street Journal (WSJ) text in the Penn Treebank (PTB; Marcus et al., 1993) to derive the dataset; these four resources are: (1) the internal annotation of noun phrases in the PTB by Vadas and Curran (2007), (2) DeepBank (Flickinger et al., 2012), (3) the Prague Czech–English Dependency Treebank 2.0 (PCEDT; Hajič et al., 2012) and (4) NomBank (Meyers, 2007). The resulting resource

can be perceived as a collection of datasets each containing more or less the same compounds but with annotations from the four resources named above. More specifically, the dataset contains triple bracketing annotation of the same set of multi-word compounds based on the annotations of the PCEDT, DeepBank and noun phrase annotations by Vadas and Curran (2007). In addition, the compounds are annotated with two types of semantic relations extracted from PCEDT and NomBank. Such a multi-layered annotation allows us to instantiate different versions of the dataset to satisfy some criterion, e.g. in our machine learning experiments on compound interpretation, we focus on two-word compounds and, hence, we instantiate a version of the dataset that only contains two-word compounds with dual semantic annotation from NomBank and PCEDT.

Important as it is, this dataset stands to be more than just a new resource for compound analysis in NLP. First, through this dataset we make new connections (or build bridges) across the resources that were used to derive it. In Chapter 7, for example, we exploit the NomBank relations to learn the PCEDT relations (and vice versa) in various transfer and multi-task learning configurations. Second, by drawing upon different resources, we open the door for contrastive analysis and evaluation of their annotations, both quantitatively and empirically. In concrete terms, in Chapter 4, not only do we extract the bracketing of the same set of compounds from three resources, but we also quantify the 'cross-resource' bracketing agreement. As such, we connect these three resources and enable follow-up work on annotation consistency. Third, since the semantic relations in our dataset are not limited to noun–noun compounds, we believe this dataset sows the seeds of integrating compound analysis with other tasks. For example, the NomBank relations in the dataset follow the same style as the Proposition Bank (PropBank; Palmer et al., 2005) and thus it is possible to combine learning problems from both worlds in a multi-task learning setup, for example.

A Shakeup: By approaching compound analysis through a broader meaning representation perspective, this thesis deviates from all past NLP studies on the topic. Apart from that, in Chapter 4, we also bring forward the question of type-based vs. token-based semantics by showing a few examples from our dataset that are at odds with the type-based perspective, which has been a somewhat unquestioned assumption in previous studies. Even though we end up pursuing a type-based approach, in part for reasons that have to do

with the resources used to create the dataset, we believe that—at the very least—we succeed in demonstrating a need for further critical reflections on some of the underlying assumptions regarding compound analysis in NLP.

No Longer Overlooked: To date, the identification of noun–noun compounds has received scant attention in the voluminous NLP literature on compound analysis. In Chapter 4, we establish why this seemingly simple task is in fact non-trivial and propose a new syntax-based approach to compound identification. We demonstrate how our new method addresses some of the shortcomings of the near-universally used method to compound identification by Lauer (1995).

Tried and Tested: Throughout Chapters 5, 6 and 7, we systematically evaluate various properties of word embeddings and neural networks as well as learning strategies in light of their performance on the compound interpretation task. The experiments presented in those chapters are aimed at understanding the interplay between word embeddings, neural networks and compound interpretation. Hence, we do not just experiment with one embedding model or two, but design a systematically varied experimental setup leading to a total of 60 experiments with word embeddings alone in Chapter 6. By casting such a wide net we are able to empirically determine which properties of word embeddings impact the performance of our neural classifier. Likewise, to gauge the effect of neural hyperparameters, we conduct a sensitivity analysis of our model and complement it with random search experiments for hyperparameter optimization. In Chapter 7, we present a comprehensive series of experiments on transfer and multi-task learning for compound interpretation. To the best of our knowledge, we are the first to put these learning strategies to use for the task of compound interpretation. It is important to highlight, however, that the dual annotation in our dataset plays a central role in facilitating the experiments with transfer and multi-task learning. Lastly, throughout these empirical parts of the thesis, we take special care to reflect on the evaluation metrics we use to measure the performance of our models in view of the relation distribution in our dataset.

1.2.1 Limitations

First, the fact that this work exclusively deals with noun–noun compounds in the English language poses a major limitation we cannot overlook. While some methodological aspects might be applicable to other languages, it is not obvious if we can directly draw parallels with other languages, not least because compounding is largely a syntactic process in English, but a morphological one in many other languages such as German or Norwegian. Second, even though the dataset we create includes bracketing annotations, we do not conduct any experiments to learn compound bracketing. Third, our experiments on compound identification rely on the gold-standard syntactic annotation of the PTB, but to ultimately evaluate the utility of our new method one will need to apply it on the output of automatic parsing. Fourth, we assume that all the compounds in our dataset are compositional; i.e. we do not engage with the compositionality prediction task. This assumption is largely motivated by the annotation guidelines of the resources we use to derive the dataset, e.g. Meyers (2007, p. 9) states that "most idioms and idiom-like units are removed from consideration" in NomBank. Finally, when it comes to the annotation quality, our dataset is—of course—constrained by the quality of the original annotations in the resources we use; the compositionality assumption in NomBank is a case in point.

1.3 Thesis Outline

The thesis is organized in six core chapters, in addition to a concluding chapter.

Chapter 2: Noun–Noun Compounds in Linguistics and NLP

In this chapter, we take a close look at noun–noun compounds, or complex nominals somewhat more generally, and establish some of the key terminology and concepts for the following chapters. We provide a linguistic background on the definition(s) of complex nominals, linguistic approaches to compound interpretation and the different types of compounds discussed in theoretical linguistics. Further, we contrast these theoretical approaches with a selection of computational approaches to compound analysis and introduce the various sub-tasks that are often addressed in the context of compound analysis in

NLP. Lastly, we review a selection of recent NLP studies on compound interpretation.

Chapter 3: Annotation of Noun–Noun Compounds and Beyond

In this chapter, we present a concise survey of the existing taxonomies and datasets for compound interpretation in NLP, and appraise three of the publicly available datasets and contrast their annotations on a handful of compounds. Furthermore, we review several linguistic annotation frameworks that go *beyond* just representing the meaning of noun–noun compound constructions to phrasal and full sentential structures.

Chapter 4: Resource Creation

This chapter describes the process of deriving a new noun–noun compound dataset from a selection of well-established general-purpose linguistic resources. First, however, we present a comprehensive study of compound identification as a prerequisite step towards creating a new compound dataset from annotations over running text. We then define several versions of our dataset that include syntactic and semantic annotation of compounds and evaluate the bracketing agreement among the resources we use. Further, we reflect on the annotations of NomBank and PCEDT and how they relate to each other and define the dataset we use for the compound interpretation sub-task in the rest of the thesis.

Chapter 5: Embeddings and Similarity-Based Classification

In this chapter, we give a brief background on distributional semantic models and detail the motivation behind using embeddings as input representation for compound interpretation. In addition, we introduce the embeddings models we train for our experiments as well as the text pre-processing pipeline and (hyper-)parameters used to train them. We also report baseline results of vector arithmetic methods to predicting the semantic relations of compounds using simple similarity-based techniques, such as the k-nearest neighbors algorithm.

Chapter 6: Neural Classification

In this chapter we present a comprehensive series of experiments on using neural network models to learn the semantic interpretation of noun–noun compounds. These experiments are geared towards determining which aspects or properties of word embeddings as well as neural classification models affect the overall performance on three noun–noun compound datasets, viz. the Tratz, NomBank and PCEDT datasets. We also experiment with different neural architectures to gauge the role of the head and modifier nouns in predicting the compound relations. We conduct a series of experiments on hyperparameter tuning, mainly to validate our choice of hyperparameters and quantify their impact on classification accuracy across the three compound datasets mentioned above. Moreover, we investigate the learning curves of the neural classification model and quantify the effect of random initialization on the final performance of the classifier.

Chapter 7: Transfer and Multi-Task Learning for Compound Interpretation

In this chapter, we empirically evaluate the utility of two learning strategies, transfer and multi-task learning, that can exploit the relationship between our noun–noun compound datasets. Through a comprehensive series of experiments and in-depth error analysis, we investigate whether or not transfer learning (via parameter initialization) and multi-task learning (via parameter sharing) can help improve the performance of our neural classification models. Furthermore, we demonstrate how the dual annotation with relations over the same set of compounds in our dataset can be exploited to improve the overall performance of a neural classifier on the less frequent, but more difficult, relations in PCEDT and NomBank. We gauge also the ability of our neural models to generalize over unseen examples and the influence of lexical memorization on their performance.

Chapter 8: Summary and Concluding Remarks

In this chapter, we summarize our findings and experimental results and discuss potential avenues for future work.

Chapter 2

Noun–Noun Compounds in Linguistics and NLP

> *Although the literature on this creature is extensive and intriguing, all evidence suggests that the beast is mythical.*
>
> —Levi (1978, p. 46)

In this chapter, we try to demystify noun–noun compounds—or the mythical beast per Levi's words.[1] We start with a general linguistic background on the definition(s) of noun–noun compounds and linguistic approaches to compound interpretation. We also review the different types of noun–noun compounds (or complex nominals more generally) that are discussed in theoretical linguistics and examine if, and how, such distinctions are made in computational studies. We then, in § 2.2, introduce the tasks that are often addressed in the context of noun–noun compound analysis in natural language processing. In § 2.3, we delve deeper into one of these tasks, namely the semantic interpretation of noun–noun compounds, and review some of the recent computational studies on this task.

[1] Levi (1978) wrote those remarks in the context of reviewing linguistic tests of 'compoundhood', such as stress patterns.

2.1 Defining Noun–Noun Compounds

Compounding is a productive and frequent construction in English; it is productive in the sense of our open-ended ability to form new compounds and frequent in the sheer number of times this construction is attested in written and spoken language, e.g. Ó Séaghdha (2008) reports that 3% of the words in the British National Corpus (BNC; Burnard, 2000) are part of noun–noun compounds. From a functional perspective, compounds serve at least two purposes: first, compounds are used as a means of 'telegraphic speech' to compress syntactic and semantic information in compact forms (Levi, 1978, p. 56–60). To use an example from Li (1972), the noun–noun compound *cradle song* roughly packs the sentential structure "a song to lull a child in the cradle to sleep". Second, compounds are used as a naming device to refer to entities that have no name, and in this context Downing (1977, p. 824) aptly states that "[c]ompounding thus serves as a back door into the lexicon".[2]

It is therefore not surprising that the syntax and semantics of complex nominals—as a linguistic construction—have received a fair amount of attention and scrutiny in the functional and generative schools of linguistics (Jespersen, 1949; Li, 1972; Downing, 1977; Levi, 1978; Warren, 1978, inter alios). In this section, we take a somewhat historical perspective to briefly review some of the influential linguistic studies that arguably shaped contemporary NLP research on noun–noun compounds. Although we will not pursue many of the notions and distinctions presented in this chapter, we believe it is instructive to review them—albeit briefly—for more linguistically informed computational research on noun–noun compounds, if nothing else.

2.1.1 A Multitude of Studies and Names

Despite the extensive attention and interest, there has been little agreement among the scholars on the definition of noun–noun compounds and how they ought to be interpreted. These variations are reflected in the fact that "[t]here are almost as many names for noun compounds and their relatives as there are linguistic studies of them" (Lauer, 1995, p. 28). These names include: noun compounds (or compound nouns), nominal compounds (or compound nominals), nominalizations, noun sequences, noun–noun compounds,

[2]Downing (1977) here refers to the compounds that survive beyond the momentary use to become established compounds and potentially enter the lexicon, cf. § 2.1.3.

noun+noun compounds and complex nominals. These different names do not necessarily assume the same definition of the construction they refer to (cf. Levi's definition of complex nominals in § 2.1.2). In addition, some of these names are considered a sub-type of others; specifically, nominalization is a type of complex nominals in Levi (1978). We will use the term noun–noun compounds throughout this thesis, except in this chapter where we will use different terms and definitions depending on the studies we are referring to or reviewing. In addition, being the broadest term, we will use 'complex nominals' as a catch-all term to describe the collection of phenomena studied in the works we present in this chapter.

Identifying consistent linguistic tests of 'compoundhood' poses a major source of disagreement. Levi (1978), for example, examines three tests (viz. fronted stress, permanent vs. fortuitous aspect or bond and semantic specialization) and demonstrates that they are mutually inconsistent. She even goes further to say that "there are *no* clear and consistent criteria according to which an entity called nominal compounds may be identified" (Levi, 1978, p. 46). Ryder (1994) also reviews, inter alia, phonological and syntactic tests of compoundhood, such as syntactic inseparability.[3] She arrives at a somewhat similar conclusion that none of the tests "can serve as an absolute test". However, Ryder points to the everlasting quest of morphologists to reliably define what constitutes a word and concludes that "the concept of a noun-noun compound is viable even if discrete boundaries for the category are not" (Ryder, 1994, p. 16). Later in this section, we will return to some of the different types of compounds that are often discussed in these tests.

2.1.2 Finite vs. Infinite

Early linguistic studies of compounds held a wide spectrum of views on the semantic interpretation of complex nominals. These views vary from assuming that nine abstract relations are sufficient to interpret complex nominals (Levi, 1978) to suggesting that they simply cannot be interpreted using a finite set of relations (Downing, 1977).

Levi (1978) studies the syntax and semantics of what she calls "complex nominals" which include three sets of linguistic constructions: noun

[3]Syntactic inseparability or indivisibility is the inability to insert a word between the compound's constituents. In other words, a noun compound can only be modified as a whole e.g. *bad state lawyer* vs. **state bad lawyer*.

compounds (e.g. *apple cake*), nominalizations (e.g. *film producer*) and noun phrases with nonpredicating adjectives (e.g. *electrical outlet*).[4] Grounded in the theories of generative grammar, Levi (1978) seeks to detail the process of deriving complex nominals and their productive and idiosyncratic aspects. She maintains that all complex nominals are derived—from underlying phrase structures—by applying one of two transformational syntactic processes, namely: predicate nominalization and predicate deletion. The result of the first derivation process is the set of complex nominals whose head is a nominalized verb such as *city planner*. The second derivation process refers to transforming a clause to a complex nominal by deleting one of a small set of nine "recoverably deletable predicates" (RDPs), viz. `CAUSE`, `HAVE`, `MAKE`, `USE`, `BE`, `IN`, `FOR`, `FROM` and `ABOUT`.[5] Levi (1978, p. 6) claims that these RDPs represent "the only semantic relations which can underline complex nominals" that are derived by deletion. Levi's set of RDPs has later formed the basis for some of the inventory-based approaches to noun–noun compound interpretation in NLP, e.g. the work by Ó Séaghdha and Copestake (2007) (cf. § 2.3 and § 3.2.1).

Downing (1977) starts her article by critiquing Levi's analysis of the syntax and semantics of complex nominals, among other generative analyses. Downing (1977) points to the fact that some of Levi's RDPs are inherently vague and underspecified, and that Levi chose such predicates to limit the number of underlying relations and show that compounds are not "boundlessly idiosyncratic". As Downing (1977) highlights, Levi derives the analysis of the compounds *headache pills* and *fertility pills* from the same (recoverably deletable) predicate (or relation) `FOR`, even though the former are intended to relieve headache and the latter are used to enhance reproductive fertility. Geared towards investigating the functional role of noun compounds, Downing focuses on the creation and interpretation of novel compounds in English.[6]

[4] Levi (1978) explains that the term "nonpredicating adjectives" refers to adjectives that cannot be used in the predicate position as well as adjectives that lead to nonsynonymous expressions if used in the predicate position in contrast to the prenominal modifier position. To illustrate the latter feature of nonpredicating adjectives, Levi (1978) gives the example of "a logical fallacy" $\neq$ "a fallacy which is logical".

[5] Levi (1978) distinguishes between prenominal modifiers that are derived from the subject of the predicate and those derived from the object of the predicate. If this distinction is taken into account, the number of RDPs becomes 12.

[6] Downing (1977) states that she opts for investigating novel compounds, in contrast to corpus-extracted and varyingly lexicalized compounds, to "avoid entanglement with the historical process" of lexicalization as well as to focus on the functional role of compounding.

However, unlike Levi (1978), Downing (1977) only studies noun compounds (rather than complex nominals) and adopts the definition by Li (1972, p. 19) that noun compounds are "the simple concatenation of any two or more nouns functioning as a third nominal." Downing (1977) conducts a four-part experiment to discover the potential semantic relations underlying noun compounds and investigate the use of compounding as a naming device, among other goals. Pursing the first-mentioned goal, Downing (1977) concludes that there is "no finite set of compounding relationships".[7] This conclusion has subsequently come to influence NLP research on compounds and lead to the so-called paraphrasing approach to compound interpretation (Nakov, 2007). Interestingly though, Downing (1977) also suggests a list of twelve relations—the most common relations observed in her experiments—that any inventory of compound relations should include. Downing (1977, p. 828) emphasizes that she makes no claim that her list "exhausts the possible compounding relationships, or that its members adequately reflect the full, essential, semantic content of the compounds reducible to them."

Finally, Warren (1978) offers one of the earliest comprehensive and systematic descriptive analysis of noun–noun compounds based on a sizable set of 4,557 compound types extracted from an early version of the Brown Corpus (Francis & Kucera, 1979). Based on an extensive data analysis, Warren (1978) proposes a four-level hierarchical taxonomy of six major semantic relations (or participant roles as she calls them) that are subdivided into finer-grained relations. The six major relations are: `Constitute`, `Possession`, `Location`, `Purpose`, `Activity-Actor` and `Resemblance`. Warren's work is noteworthy for its comprehensiveness and influence on NLP research, e.g. Lauer (1995); Barker and Szpakowicz (1998); Girju et al. (2005) define relations similar to Warren's.

2.1.3 Deictic, Novel and Established Compounds

Ryder (1994) distinguishes between three types of "noun–noun combinations" to establish what she actually means by noun–noun compounds. These three groups are: deictic compounds, novel compounds and established compounds.

[7] Of course, Downing (1977) is not the only scholar to maintain such a position; for example, Kay and Zimmer (1990, p. 239) state that "the list of interpretations with different semantic relations holding between the two elements of the compound may be extended indefinitely to the limits of one's ingenuity."

Deictic compounds, like other categories of deixis, cannot be interpreted without contextual information as they are normally created in conversational settings "to satisfy a fleeting discourse need" (Ryder, 1994, p. 8). Deictic compounds are often excluded from linguistic studies of compound interpretation because their meaning depends on extra-linguistic information; for example, Downing (1977) uses the term deictic compounds to describe the subset of "novel compounds used in conversational situations" which she excludes in her study. In this context, Downing (1977) introduces the famous example of deictic compounds, *apple-juice seat*, which is often cited in NLP studies to demonstrate the difficulty of interpreting noun–noun compounds.[8]

Novel compounds, in contrast, are interpretable—to some extent—out of their original context. However, as Ryder explains, "this requirement still does not necessarily limit the kinds of relationships that could hold between the element nouns" (Ryder, 1994, p. 9). Furthermore, novel compounds refer to potentially more permanent states, in contrast to deictic compounds that are generally derived from temporary states (e.g. the apple-juice placed in front of the seat). To illustrate the limited interpretability of novel compounds, Ryder (1994) explains that the hearer only has access to the predictable meaning of *pencil sharpener* (a tool for sharpening pencils), but not its size or shape.

Established compounds are originally novel compounds, but they have been accepted by the language speakers and become part of their lexicon over time. This does not imply that all established compounds are idiomatic, but idiomatic compounds have to be established, otherwise it would be impossible to interpret them. In fact, Ryder (1994) points out that established compounds become like words and undergo semantic drift. She gives the example of *cod fish*, which is the common name for a fish specie, but according to Ryder cod fish originally meant "a fish that is like a cod".[9]

Based on Ryder's discussion of the three types of noun–noun combinations, it is plausible to assume that the boundaries between these three groups are not always clear-cut. As shown above, Downing (1977) treats deictic compounds as a subtype of novel compounds. Moreover, it is not obvious if, and when, a novel compound becomes an established one (Warren, 1978, p. 46). In fact, Spärck Jones (1983) maintains that lexicalized compounds (a type of

[8] "[T]he seat in front of which a glass of apple-juice had been placed" (Downing, 1977, p. 818).

[9] Cod (or codd) means a bag or pouch in Old English, according to the Oxford English Dictionary (OED).

established compounds) and non-lexcalized (novel) compounds form two ends of one wide spectrum of novelty vs. lexicalization. Warren (1978, p. 50) also holds the same view that the distinction between established and novel compounds is a controversial one.

2.1.4 Endocentric vs. Exocentric

Compounds can be classified—along with other linguistic constructions—as endocentric and exocentric based on the relation between the compound as a whole and its constituents (Levi, 1978; Warren, 1978; Ryder, 1994). In addition to the grammatical implications of such a distinction (cf. Bloomfield, 1984, p. 235–236), exocentric compounds are semantically headless constructions whereas endocentric compounds are headed by one of their constituents. In other words, an endocentric compound refers to a specific subset of the set of objects denoted by its head (i.e. a hyponym of its head), but an exocentric compound, in contrast, is not a hyponym of its head constituent. Idiomatic compounds are typical examples of exocentric compounds, e.g. *soap opera* is neither a subtype of soap nor of opera.[10]

2.1.5 Nominalizations vs. Root Nominals

Several linguistic accounts of compound interpretation make yet another distinction between compounds based on whether or not their head is derived from a verb (Levi, 1978; Warren, 1978). These two types of compounds are often referred to as deverbal compounds (headed by nominalizations, e.g. *house rental*) and root compounds (headed by non-derived nouns, e.g. *bed bug*). Levi (1978) assumes this distinction by defining two derivation processes that distinguish between derivation by nominalization and derivation by deletion. Around the same time, Roeper and Siegel (1978) published a study devoted to the particular root–verbal distinction, relating it to semantic predictability: "Root compounds are as unpredictable as ordinary words and must therefore be entered in the atomic lexicon" (Roeper & Siegel, 1978, p. 206). Warren

[10]Bauer (2011) and Fabb (2017), among others, assume a third type of compounds called co-headed compounds or coordinative compounds in which the two constituents exhibit head-like properties, for example *sofa bed*. Levi (1978, p. 6) also refers to this type of compounds but as one of three groups of exocentric compounds (that she excludes from her study).

(1978, p. 57), in contrast, excludes "noun–noun compounds in which one noun is deverbal".

Deverbal compounds can be further classified into different types of nominals depending on whether or not they inherit the argument structure of their base verbs, viz. (1) argument-supporting nominals (or complex event nominals), (2) result nominals and (3) simple event nominals; only the first type inherits the argument structure of the underlying base verb (Grimshaw, 1990; Iordachioaia et al., 2016). The difference between argument-taking and result nominals is perhaps best illustrated by contrasting *exam* (result nominal) and *examination* (argument-supporting nominal) in the following example:

(2.1) The examination of the patients

(2.2) *The exam of the patients

2.1.6 Theoretical vs. Computational

Moving from the theoretical accounts of noun–noun compounds to computational modeling, we ought to ask what does the theory imply for the computational perspective on compound analysis?

In one of the early computational studies on compound interpretation, Spärck Jones (1983) writes that for some of the linguistic distinctions and categorizations to be applicable in computational models, there must be an explicit characterization of them. Hence, based on this observation, some of these distinctions seem to make their way to computational studies of compound analysis while others are either sidestepped or subsumed under one category. For example, the distinction between deictic, novel and established compounds can be applied in computational linguistics provided that we have a method to determine a compound's membership to one of these three groups. In the case of computational models that solely rely on corpora of written text, one can arguably assume that deictic compounds are much less likely to occur, which leaves us with two types, viz. novel and established compounds. These two types can then be distinguished based on their frequency in a given corpus. Indeed, Lapata and Lascarides (2003) present a statistical method to detect novel compounds and "distinguish those which are valid compounds from nonce terms".[11] Likewise, we can assume that lexicalized or idiomatic

[11] As we will show in § 4.3, the method proposed by Lapata and Lascarides (2003) is

compounds, such as *couch potato*, are likely to be identifiable using a simple dictionary look-up method. However, this does not necessarily mean that lexicalized compounds were always excluded from computational studies on compound interpretation, for instance the annotation schemes of both Tratz and Hovy (2010) and Ó Séaghdha and Copestake (2007) include a special relation indicating lexicalization.

Unless there is a specific need for it, the endocentric vs. exocentric distinction is rarely addressed in NLP research. For example, Ó Séaghdha (2008, p. 33) does not distinguish between endocentric and exocentric compounds under the assumption that the semantic relations in his framework denote the relation between the compound's constituents and not the compound as a whole. Nakov (2007) introduces the distinction in the theoretical review of compound analysis, but he does not explicitly specify whether or not it is addressed in his work. Given that many of the compound interpretation studies in NLP were focused on two-word compounds (i.e. compounds consisting of two nouns only), we speculate that they did not make the endocentric–exocentric distinction, partly because they are not directly concerned with the concept of 'headedness'. However, whenever the need for identifying the head in compounds arises, there is no way around determining whether a compound is endocentric or exocentric. For instance, knowing whether a compound is endocentric or exocentric is essential in an application like textual entailment where a compound like *lung cancer* semantically entails cancer, assuming knowledge about the endocentricity of this compound (Nakov, 2013).

Deverbal compounds are treated differently across the NLP literature where some studies focus exclusively on nominalizations whereas others exclude them altogether. Lapata (2002) studies only a particular group of deverbal compounds where the prenominal modifier is either the underlying subject or direct object of the original verb (from which the head noun was derived). Lauer (1995), on the other hand, excludes nominalizations in favor of the so-called propositional compounds (cf. § 2.3).[12] Both Lauer (1995) and

primarily motivated by the fact that the compound identification heuristic they use—which was proposed by Lauer (1995)—identifies false positive examples.

[12]More accurately, Lauer (1995) excludes the same set of deverbal compounds Lapata (2002) studies, i.e. deverbal compounds whose prenominal modifier plays the role of the semantic subject or object of the original verb corresponding to the nominalized head noun. Lauer (1995, p. 62) explains that the compound *peasant rebellion* is excluded because *peasant* plays the role of the subject for the verb *rebel*, but the compound *city dwellers* is not excluded because *city* is neither the subject nor the object of the verb *dwell*.

Lapata (2002) use lexicon-based methods to identify deverbal compounds, which is obviously limited to whatever nominalizations are included in the lexica they use. Neither Ó Séaghdha (2008) nor Tratz (2011) exclude nominalizations from their studies, but the latter pays special attention to this group of compounds by choosing not to define a relation that denotes the object of a nominalized head in order to avoid ambiguity as many compounds can fit this relation as well as others (Tratz, 2011, p. 47).

Based on the discussion above, we believe that carrying over many of the distinctions and categorizations held in theoretical linguistics to computational linguistics or NLP can sometimes imply intensive and informed annotation efforts. Indeed, detecting some of these distinctions have themselves become standalone (sub-)tasks related to compound analysis, for example, detecting the compositionality of noun–noun compounds (cf. § 2.2).

Lastly, Spärck Jones (1983) explains that viewing compound interpretation from a computational perspective adds new problems that are otherwise not encountered in linguistic studies of compounds. For example, identifying whether or not a sequence of words is a sequence of nouns (i.e. PoS tagging) and consequently determining if this sequence of nouns is a valid compound itself—we will refer to this task as compound identification, cf. § 2.2.1.

2.1.7 A Preliminary Definition

We define noun–noun compounds as constructions consisting of two or more nouns that stand in a head–modifier relation. In practice, however, we take a syntax-based approach to define (and identify) noun–noun compounds where we require the sequence of nouns that make up a compound to be dominated by the same parent node (i.e. noun phrase or NP). Furthermore, we also distinguish between compounds based on proper nouns and common nouns, but we will leave the details to Chapter 4. The linguistic resource from which we derive our compound dataset pose their own constraints as well, which we will also return to in Chapters 3 and 4.

2.2 Noun–Noun Compound Analysis

We dedicate the rest of this chapter to reviewing some of the previous studies on noun–noun compound analysis in NLP. However, before we turn to the

actual review, in the following we establish the definition of the sub-tasks related to compound analysis.

In the early 1980s Finin (1980) and Spärck Jones (1983), inter alios, offered extensive views on the problems involved in 'computationally' interpreting noun–noun compounds, such as "recognising that some string of words is actually a string of nouns" (Spärck Jones, 1983, p. 5) and "lexical interpretation (mapping words into concepts)" (Finin, 1980, p. 310).

Overall, we identify at least five different tasks or problems related to noun–noun compound analysis in NLP:

1. Identification of noun–noun compounds in text (or speech), i.e. determining if a sequence of nouns is in fact a compound
2. Sense disambiguation of the compound's constituents
3. Syntactic analysis of the compound's internal structure, i.e. left vs. right bracketing of compounds consisting of more than two constituents
4. Semantic interpretation, i.e. predicting the semantic relation holding between the compound's constituents
5. Predicting the compositionality of noun–noun compounds, i.e. the relatedness between the meaning of the compound as a whole and the meaning of its constituents

These tasks, more or less, cover the thematic focus of almost all the research studies on noun–noun compounds in NLP. However, despite broad agreement on the different problems involved in compound analysis, these tasks are not equally researched in NLP; for example, compound identification has received comparatively less attention than the task of semantic interpretation. Moreover, the approaches to solve these tasks, not unexpectedly, vary in terms of the methods and resources used (cf. § 2.3). In the following, we define each of the five tasks of compound analysis and refer to some of the relevant work in NLP.

2.2.1 Identification

Noun–noun compound identification is the task of determining whether or not a sequence of nouns actually forms a compound, assuming that we already

have a method to identify sequences of nouns in plain text (i.e. PoS tagging). More concretely, compound identification amounts to identifying that there are two noun *sequences* in Example 2.3, *century compounds* and *research studies*, and that only one of them makes a valid noun–noun compound (viz. *research studies*).

(2.3) In the last $century_N$ $compounds_N$ were scrutinized in many $research_N$ $studies_N$.

As mentioned before, noun–noun compound identification in particular has received little attention in recent work in NLP. In Chapter 4, we will elaborate more on why this foundational task has been somewhat neglected, discuss the shortcomings of existing approaches and introduce a syntax-based method that aims to resolve some of these issues. However, for now, it suffices to say that many of the previous studies (cf. § 2.3) have relied on a PoS-based approach, proposed by Lauer (1995), that introduces many false positives, like *century compounds* in the example above.

2.2.2 Bracketing

Compounding, in English, is a recursive process which means that noun–noun compounds can be as long as the following example by Levi (1978, p. 5):

(2.4) *horseback riding school cafeteria breakfast menu substitution list*

We refer to the compounds consisting of three or more nouns as multi-word compounds or N-ary compounds, in contrast to two-word compounds (or binary compounds). This recursive property gives rise to the compound bracketing task, which involves analyzing the internal structure of N-ary compounds by breaking it down into binary subconstituents. Multi-word compounds—like prepositional phrases—are "every way ambiguous" constructions, i.e. the number of possible structural analyses is equivalent to the number of binary trees over the terminals. In other words, the structural or syntactic ambiguity of noun–noun compounds "grows exponentially with their length (following the Catalan sequence)" (Isabelle, 1984, p. 509). Even though not all bracketings necessarily lead to equally plausible readings, this does not eliminate the need to consider them, especially in computational analysis.

Compound bracketing can be beneficial for accent placement in speech-to-text synthesis applications, as shown by Sproat (1994). In addition, bracketing is interrelated with semantic interpretation of noun–noun compounds; for example, the meaning of the compound in Example 2.5 varies depending on how its internal structure is analyzed, in that it could refer to a price report made for consumers (2.5a) or a report on consumer prices (2.5b).

(2.5) *consumer price report*

a. [consumer [price report]]

b. [[consumer price] report]

While many recent NLP studies on compounds focused on two-word compounds (which obviously do not require bracketing), compound bracketing has received some attention, either as a separate task (Resnik, 1993; Lauer, 1995; Pitler et al., 2010; Nakov, 2013) or as part of parsing noun phrases (Bergsma et al., 2010; Vadas & Curran, 2011).

2.2.3 Constituent Sense Disambiguation

Constituent sense disambiguation (or nominal sense disambiguation) is concerned with disambiguating the meaning of polysemous constituents in a given compound. For example, the compound *pilot program* has at least two interpretations depending on the meaning of the prenominal modifier *pilot*—it can be a trial program or a program for aircraft pilots. Like compound identification, this task has received little attention in NLP. McShane et al. (2014) list several reasons why nominal sense disambiguation has not been perceived as central as compound interpretation to compound analysis. Among other reasons, they refer to the fact that some of the previous research studies have focused on domain-specific compound analysis which primarily deals with monosemous nominals, such as the medical domain in Rosario and Hearst (2001). In addition, many of the previous studies on compound analysis revolve around participation in shared tasks or competitions, such as Hendrickx et al. (2013), which in turn do not require sense disambiguation.

We believe the very nature of the existing compound datasets makes certain assumptions with regard to the necessity of sense disambiguation in compound analysis. As we will show in the following chapters, nearly all of the recent noun–noun compound datasets are type-based, i.e. it is assumed that the relation holding between the constituents of a compound is the same

regardless of the context, and hence these datasets do not include the original context from which the compounds were extracted. Thus, these datasets implicitly assume a specific sense of the polysemous constituents in order for the type-based assumption to hold.

Although in theory it is clear that we need sense disambiguation as part of compound analysis, this need not be as clear in practice. Not only because past work, much like ours, focuses on other tasks related to compound analysis, but also because the need for sense disambiguation is not as pressing or frequent in practice. We will elaborate on this in § 4.5.3 in the context of our datasets; however, for now, it suffices to say that we find very few compound instances in our dataset whose constituents express different meanings in different contexts (or occurrences).[13] One such example (from our dataset; cf. Chapter 4) is shown in Examples 2.6 and 2.7, where *coverage* in the former refers to insurance coverage whereas in the latter it refers to media coverage.

(2.6) Industry officials say the Bay Bridge – unlike some bridges – has no *earthquake coverage*, either, so the cost of repairing it probably would have to be paid out of state general operating funds.

(2.7) The Associated Press's *earthquake coverage* drew attention to a phenomenon that deserves some thought by public officials and other policy makers.

2.2.4 Interpretation

The semantic interpretation of noun–noun compounds has been under active research and investigation in computational linguistics for decades, dating back to as early as the work by Su (1969), Finin (1980) and Spärck Jones (1983). Given the disagreement on how to interpret noun–noun compounds in theoretical linguistics, it is no surprise that similar trends are observed in computational linguistics and NLP. Existing approaches to compound interpretation in NLP, by and large, subscribe to one of two competing views on compound interpretation in theoretical linguistics; viz., compounds are interpretable using a finite set of (abstract) semantic relations vs. the number of semantic relations needed to interpret compounds is unbounded (cf. § 2.1.2).

[13] Our dataset, of course, contains polysemous words, but the observation here concerns whether or not these words express *different* meanings when they occur more than once in the dataset. We will address this question in more detail in § 4.5.3.

These two views are often referred to as the inventory-based and paraphrase-based perspectives. The approaches in the former perspective are mainly inspired by the works of Levi (1978) and Warren (1978), among others, but do not come close to agreeing on what kind of relations or taxonomies are adequate to interpret noun–noun compounds. For example, Ó Séaghdha and Copestake (2007) define a coarse-grained set of relations (viz. six relations based on the theoretical work by Levi, 1978), whereas Tratz and Hovy (2010) assume a considerably more fine-grained taxonomy of relations that consists of 43 relations. The paraphrasing perspective takes the same position as Downing (1977) and questions the very assumption that noun–noun compounds are interpretable using a predefined finite set of relations. For example, in one of the early studies on computational models for compound interpretation, Finin (1980, p. 311) maintains that "there can be arbitrarily many possible relationships between the two nouns, each relationship appropriate for a particular context".[14]

In the inventory-based perspective, the task of noun–noun compound interpretation is often approached as a multi-class classification problem over a predefined taxonomy of relations. Assuming a supervised learning setup and a dataset of labeled compounds, the task is simply to learn to classify the semantic relations holding between the constituents of the compounds based on a set of training examples. However, the exact definition and difficulty of this task largely depends on the set of relations used and their distribution in the dataset, among other things. We return to a more in-depth review of some of the recent inventory-based studies in § 2.3.

In the paraphrase-based perspective, compound interpretation amounts to finding the most likely paraphrase of a given compound which can be either a prepositional or verbal paraphrase. Most notable in this strand of research is the work by Nakov and Hearst (2013) who through Internet search queries, attempt to find the set of all possible paraphrasing verbs that might paraphrase a given compound and assign weights to each of the verbs based on its frequency. To interpret the compound *malaria mosquito*, for example, they issue search queries like "mosquito that * malaria" and process the returned search snippets leading to paraphrasing verbs like *carry*, *spread* and

[14]We do not review Finin's (1980) work in this thesis. Briefly though, couched in the role-filling theory, his approach assumes that one of the nouns in the compound denotes an event, and hence has an event structure, and the other noun fills a role in that structure. Finin (1980) makes this assumption for non-derived nouns as well as nominalizations.

cause. They argue that their approach is more apt to capture the nuances of compound semantics like in the case of *malaria mosquito* where the mosquito does not actually *cause* malaria but merely transmits or carries it. This approach was adopted in SemEval-2010 Task 9 and SemEval-2013 Task 4 (Butnariu et al., 2010; Hendrickx et al., 2013, respectively).

2.2.5 Compositionality

Compound compositionality prediction is the task of determining the degree of compositionality through relating the meaning of the compound as a whole to the meaning of its constituents, i.e. determine if the meaning of the compound is derivable from the meaning of its constituents. Semantic compositionality of noun–noun compounds poses a dilemma to researchers in NLP, because lexicalization (or compositionality) form a continuum wherein it is difficult to draw a line between lexicalized and compositional compounds (Spärck Jones, 1983). Therefore, past work offers different views on the granularity of compound compositionality ranging from binary (compositional vs. non-compositional) to six-point scales of compositionality. Most of the previous studies, however, seem to agree on approaching this problem in isolation from the rest of compound analysis tasks (Reddy et al., 2011; Hermann et al., 2012; Farahmand et al., 2015; Yazdani et al., 2015). That said, some of the existing datasets that annotate the semantic relations of compounds single out lexicalized compounds by annotating them with special relations, but they are not otherwise concerned with predicting compound compositionality (Ó Séaghdha & Copestake, 2007; Tratz & Hovy, 2010).

Most of the recent studies use compositionality prediction as a means to compare distributional semantic models such as word embeddings. For example, Schulte im Walde et al. (2013) compare syntax-based and context-based distributional semantic models with regard to their ability to predict the compositionality of German noun–noun compounds. Likewise, Cordeiro et al. (2016) use the compound compositionality datasets by Reddy et al. (2011) and Farahmand et al. (2015), among others, to evaluate distributional semantic models.[15]

[15]The dataset by Reddy et al. (2011) consists of 90 noun–noun compounds annotated on a compositionality scale of six points, whereas the dataset by Farahmand et al. (2015) consists of 1,048 two-word English noun–noun compounds annotated using a binary scale as (non-)compositional and (non-)conventionalized compounds.

2.3 A Selection of Studies: Literature Review

Existing approaches to compound interpretation in NLP vary depending on the taxonomy of compound relations as well as the machine learning models and features used to learn those relations. In this section we review a selection of studies on noun–noun compound interpretation. The choice of these studies is partly motivated by their relevance to our own work in later chapters, but we also include other studies to further explain some of the problems and tasks related to compound analysis. We focus here on the approaches that cast the interpretation problem as an automatic classification task over a finite set of relations. Some of the following studies also address the problem of bracketing, but we do not comment on the bracketing algorithms or models because they are not studied in any interesting depth in this thesis.

2.3.1 Lauer (1995)

Lauer (1995) presents one of the early influential studies on statistical methods for noun–noun compound interpretation (and bracketing), using the Grolier encyclopedia (an eight-million word corpus) to estimate word probabilities. He tests his models on a dataset of 244 three-word bracketed compounds and 282 two-word compounds. The compounds were annotated with one of eight prepositions, which Lauer (1995) takes to approximate the semantics of noun–noun compounds; for example, *airport food* would be interpreted as *food at the airport* in Lauer's framework (Lauer, 1995, p. 155). The prepositions Lauer uses are in fact based on Warren's (1978) study which includes a list of typical paraphrasing prepositions for each of the six major semantic classes in her study (cf. § 2.1.2).

Assuming an inventory of frequent lexical items like prepositions (in contrast to abstract semantic relations) allows Lauer (1995) to use unsupervised statistical models which rely solely on co-occurrence probabilities (or counts) to interpret compounds. However, the notorious polysemy of prepositions casts doubt on their ability to capture the semantics of compounds; for example, the prepositions `on`, `in` and `at` can denote time and location in different contexts, and it is debatable whether the choice of one of these prepositions over the others is a semantic question or a matter of lexical association (Ó Séaghdha, 2008). Additionally, Girju (2009) describes their work to annotate a multi-lingual dataset of compounds—extracted from the

Europarl Corpus (Koehn, 2005)—using Lauer's prepositions; she reports that 79% of the English compounds were annotated with the underspecified preposition `of`. Nonetheless, Lauer's dataset and prepositions were used in subsequent work by Lapata and Keller (2004); Girju et al. (2005); Nakov (2007); Bos and Nissim (2015); inter alios.

2.3.2 Girju et al. (2005)

Girju et al. (2005) study the semantic interpretation and bracketing of noun–noun compounds using a dataset of binary and three-word compounds annotated with the prepositional paraphrases (PPs) by Lauer (1995) and 35 semantic relations defined by the authors. Their dataset consists of compounds extracted from the Wall Street Journal (WSJ) articles in the Text REtrival Conference (TREC-9) document collection as well as an extended version of WordNet (XWN 2.0).[16] They use Lauer's heuristic (cf. § 4.3.1) to identify compounds in randomly selected sentences from the two aforementioned resources. The annotators then manually checked the identified compounds to confirm their validity. Girju et al. (2005) annotate a total of 4,504 binary compounds and 484 three-word compounds for their training split.[17] Additionally, for their test split, they annotate the same set of noun compounds used by Lauer (1995) which consists of 282 binary compounds and 244 three-word compounds. The annotation scheme by Girju et al. (2005) allows assigning more than one relation per compound, and hence 608 compounds were tagged with more than one semantic relation and almost all the compounds that are interpretable using the prepositional paraphrases received more than one PP.[18] However, it is not immediately clear how this property is reflected when evaluating the accuracy of their models. Girju et al. (2005) also asked the annotators to provide information on the order of the head and modifier nouns in the compounds. They find that 34% of the binary compounds are left-headed (i.e. the head precedes the modifier). We suspect that the relatively high percentage of left-headed compounds can be a result of their decision to include proper names as part of the compounds (combined with questionable assumptions about headedness); e.g. in their

[16] `http://www.hlt.utdallas.edu/~xwn/`. Accessed: 5 February 2019.

[17] We use the numbers reported by Girju et al. (2005) in Table 3. However, these numbers do not immediately align with the numbers reported in Table 1 in the same article.

[18] The annotators assigned the relation `Others` to the compounds that encode a relation or prepositional paraphrase other than the ones defined in the annotation guidelines.

framework, the head of the compound *GM car* is *GM* which is a proper name.

Girju et al. (2005) cast the compound interpretation problem as a classification task to predict their own semantic relations and Lauer's prepositional paraphrases. They use three supervised learning models, viz. semantic scattering, iterative semantic specialization and support vector machines.[19] They also 'replicate' the unsupervised model by Lapata and Keller (2004), which relies on web search queries, to predict the PPs by Lauer (1995). Their three supervised models utilize WordNet-based features such as the semantic class of the compound's head and modifier nouns (i.e. the so-called synset in WordNet). The unsupervised model outperforms the three supervised models on Lauer's prepositional paraphrases. In the supervised models, they find that word sense disambiguation impacts the performance of their models on the 35 semantic relations while it almost has no effect on the abstract prepositional paraphrases of Lauer (1995).

2.3.3 Kim and Baldwin (2013)

Kim and Baldwin (2013) also study the semantic interpretation and bracketing of noun compounds. They use an analogy-based method which relies on lexical semantic similarities from WordNet. The main assumption they make is that compounds "which contain similar words in corresponding positions tend to share the same semantics" (Kim & Baldwin, 2013, p. 389). For this study, they extract a dataset of noun–noun compounds from the WSJ of the PTB using a PoS-based identification heuristic, collecting a total of 2,169 binary compounds and 1,571 three-word compounds.[20] The binary compounds and the outermost parts of three-word compounds are annotated with the 20 semantic relations defined by Barker and Szpakowicz (1998). The three-word compounds were, of course, bracketed to allow annotating the 'outermost' pair of nouns. The dataset is equally split between training and test.

In addition to their own dataset, Kim and Baldwin (2013) evaluate their method on the SemEval-2007 task dataset (Girju et al., 2007), primarily to compare their method with others, such as semantic scattering by Moldovan et al. (2004). As mentioned above, Kim and Baldwin (2013) rely solely on

[19]The semantic scattering and iterative semantic specialization algorithms are described, respectively, in Moldovan et al. (2004) and Girju et al. (2003).

[20]The download link provided for the dataset in Kim and Baldwin (2013) no longer works. However, an earlier version of the dataset which only includes binary compounds is available online and described in Kim and Baldwin (2008).

lexical semantic similarity between the compound constituents; i.e., given a compound from the test split, they would compute the similarity between its head and modifier with all the heads and modifiers in the training split and assign it the relation of the most similar compound. To compute the semantic similarity, Kim and Baldwin (2013) use an open-source tool called *WordNet::Similarity* which provides several methods to measure semantic similarity or relatedness based on information from WordNet. Since these measurements are computed between pairs of nouns, Kim and Baldwin (2013) use a weighted function to combine the similarities between the heads and modifiers of two compounds.[21]

Kim and Baldwin (2013) conduct four sets of experiments focusing on compound interpretation (for binary and three-word compounds), the relative contribution of the head and modifier nouns to compound interpretation and, lastly, bracketing. Kim and Baldwin (2013) report that their model achieves an accuracy well above the majority class baseline, which is 43% (achieved by assigning the relation `TOPIC` to all the test compounds). On the SemEval-2007 dataset, their model outperforms the semantic scattering algorithm but not the memory-based method by Nastase et al. (2006); the last two methods, as Kim and Baldwin (2013, p. 398) explain, "require manual word sense information" whereas the model by Kim and Baldwin does not. To determine the contribution of the head and modifier nouns to predicting the semantic relation of a given compound, Kim and Baldwin (2013) experiment with different values of the weighting factor (in the function that adds the head and modifier similarity scores). They find that some semantic relations exhibit stronger correlation with either the head or modifier; for example, the relations `CAUSE` and `POSSESSOR` rely more on the modifier noun. However, since different relations have different correlations with the head and modifier, they decide to use a uniform weighting factor (i.e. the head and modifier similarity scores contribute equally to the combined score).

2.3.4 Ó Séaghdha and Copestake (2013)

Ó Séaghdha and Copestake (2013) present a comprehensive study on the use of kernel-based methods to interpret noun–noun compounds using different

[21] We implement a similar approach in Chapter 5, but instead of WordNet-based similarity we use word embeddings and we do not use a weighting factor to combine the head and modifier similarities (i.e. the two similarity scores contribute equally to the final combined similarity score).

types of lexical and relational information.[22] The defining aspects of Ó Séaghdha and Copestake's (2013) work can be summarized in (1) the focus on kernel methods, (2) the use (and creation) of a noun compound interpretation dataset that embraces the abstract relations by Levi (1978) and (3) the pure statistical method that breaks away from manually-created lexical resources such as WordNet in favor of corpus-based lexical and relational distributional information.

The underlying assumption in their work is that compound interpretation is an analogical process, i.e. the relational meaning of one compound can be derived or predicted from 'similar' compounds. Entailed in this assumption is the ability to define compound similarity, which Ó Séaghdha and Copestake (2013) claim to be found in three types of information: lexical, relational and contextual information. In general, lexical information is information about the individual constituents which can be extracted from lexical resources like WordNet—in the case of Kim and Baldwin (2013)—or corpus-based co-occurrence statistics (as we will apply in Chapter 5). Relational information, on the other hand, captures the contexts in which the constituents occur *together*; the assumption here is that the context will encode the relational similarity between the constituents. Of course, the constituents need not occur as a compound in the same context, but Ó Séaghdha and Copestake (2013) do enforce a constraint on the constituents to be within a window of ten words in the sentence. Finally, contextual information refers to the actual context (or sentence) in which a certain compound occurs. Ó Séaghdha and Copestake (2013, p. 335) argue that "context features are of little value for understanding compounds" based on their earlier work (Ó Séaghdha & Copestake, 2007). Therefore, they do not use contextual information in their models.

Ó Séaghdha and Copestake (2013) extract the lexical and relational information in terms of co-occurrence distributions from three copora: the British National Corpus (BNC; Burnard, 2000), a Wikipedia dump and the 2nd edition of the English Gigaword Corpus (Graff et al., 2005). They pre-process and PoS tag the three corpora as well as parse the BNC and the Wikipedia dump. They need the morpho-syntactic analyses of the corpora to

[22]The work presented in Ó Séaghdha and Copestake (2013) was preceded by several articles by the same authors (Ó Séaghdha & Copestake, 2007; Ó Séaghdha & Copestake, 2008; Ó Séaghdha & Copestake, 2009), but here we review the most recent one only (that is, Ó Séaghdha & Copestake, 2013).

collect the co-occurrence distributions from certain grammatical relations and types (which they detail in Section 4.2 of the same article). As mentioned above, Ó Séaghdha and Copestake (2013) pay special attention to the choice of kernel function in their SVMs and choose functions that are motivated by their intuition on the two types of information considered (i.e. lexical and relational).[23] They train one SVM classifier per relation casting the multi-class classification problem as a set of binary classification problems, i.e. one-vs-all. The final prediction is made based on the relation classifier that yields the highest positive (or least negative) distance to the decision boundary.

Ó Séaghdha and Copestake (2013) use the compound dataset they introduced in Ó Séaghdha and Copestake (2007) which consists of $1,443$ binary compounds annotated with semantic relations at three levels of granularity.[24] Ó Séaghdha and Copestake (2013) train their classifiers on the three levels of granularity and report evaluation results for five-fold cross-validation on the full dataset. Among other things, they find that the lexical features alone lead to better results than the relational features alone. However, their best results are obtained when both types of features (and kernels) are combined. The results they report outperform the then state-of-the-art performance on the dataset of six coarse-grained relations.

Lastly, Ó Séaghdha and Copestake (2013) also experiment with classification using only the head or modifier noun—which is similar in spirit to the experiments Kim and Baldwin (2013) conduct on the relative contribution of the head and modifier nouns. Ó Séaghdha and Copestake (2013) find that, overall, better results are achieved when both nouns are considered, but they also observe a somewhat similar pattern to Kim and Baldwin's where certain relations tend to benefit more from the head or the modifier, though the vast majority of relations seem to perform better with the head only in contrast to the modifier only.

[23] We refer to the original article by Ó Séaghdha and Copestake (2013) for a detailed discussion and comparison between the different kernel functions for each type of information, such as linear vs. distributional kernels for lexical information.

[24] The Ó Séaghdha and Copestake (2007) dataset will be described in more detail in § 3.2.1.

2.3.5 Tratz and Hovy (2010)

Unsatisfied with the heterogeneity of past work on compound interpretation, Tratz and Hovy (2010) start afresh and create the largest manually annotated dataset of noun–noun compounds. The compounds in their dataset were extracted from "two principal sources: an in-house collection of terms extracted from a large corpus using part-of-speech tagging and mutual information and the Wall Street Journal section of the Penn Treebank" (Tratz, 2011, p. 49). Their dataset consists of 17,509 unique, out-of-context noun–noun compounds annotated with a taxonomy of 43 semantic relations; we will take a closer look at the dataset and taxonomy of relations in § 3.2.3.

Tratz and Hovy (2010) train a maximum entropy (ME) classifier to learn the semantic interpretation of the compounds in their dataset as well as the dataset by Ó Séaghdha and Copestake (2007). Their classifier, for both datasets, relies on a large number of features extracted from two lexical resources (viz. WordNet and Roget's Thesaurus), surface form features (such as suffixes) and n-gram features (3- and 4-grams) from the Web 1T Corpus (Brants & Franz, 2006). The relatively large number of their WordNet-based features is noteworthy; Tratz and Hovy (2010) list 14 feature types that they extract from WordNet such as the synonyms and hypernyms of all the nouns and verbs in a word's definition. They perform 10-fold cross-validation on their own dataset and 5-fold cross-validation on the dataset by Ó Séaghdha and Copestake (2007) to allow direct comparison with the results reported by Ó Séaghdha and Copestake (2009). Tratz and Hovy (2010) report an accuracy of 79.3% on their dataset and 63.6% on the one by Ó Séaghdha and Copestake (2007), which is similar to the previously obtained accuracy on the latter dataset.[25] Tratz and Hovy (2010) assess the impact of their feature types through an ablation study that alternates between including one feature type or excluding it (while keeping the rest of the features). They report that their novel WordNet feature, which uses word definition terms, has an equally positive influence on both datasets as the more traditional WordNet hypernym features. The n-gram features, however, yield mixed

[25] Tratz and Hovy (2010) experiment with SVMs using the same features of their maximum entropy classifier and arrive at the same results for both datasets. Further, they report that the SVM classifier's performance is highly sensitive to tuning the SVM cost parameter C, which is not unexpected. However, based on what Tratz and Hovy (2010) report, they seem to be more rigorous in their training and evaluation than Ó Séaghdha and Copestake (2013) because the former report one value for the C parameter across all folds, whereas Ó Séaghdha and Copestake (2013) optimize that parameter for *each* cross-validation fold.

results, as they have a positive impact on Ó Séaghdha and Copestake's (2007) dataset, but not on their own dataset. To further assess the performance of their model, Tratz and Hovy (2010) evaluate it on a set of 150 noun–noun compounds whose constituents are unseen in their training data. These 150 examples were randomly selected from New York Times articles and used for a final inter-annotator agreement study in Tratz and Hovy (2010). Their model achieves an accuracy of 51% on the inter-annotator agreement data (i.e. the unseen compounds), which is—as Tratz (2011, p. 60) reports—"8% lower than the human agreement figure".

Being the largest dataset to annotate the semantics of noun–noun compounds, Tratz and Hovy's data can be considered one of the comparatively influential datasets for compound interpretation in NLP. Their dataset was used by others to further study compound interpretation (Dima & Hinrichs, 2015; Shwartz & Waterson, 2018) as well as for compound compositionality prediction (Hermann et al., 2012).[26] In Chapter 6, we review and replicate the work by Dima and Hinrichs (2015) and evaluate our models on a version of the Tratz and Hovy (2010) dataset.

2.3.6 Shwartz and Waterson (2018)

Shwartz and Waterson (2018) propose a paraphrasing approach to predict the relations in the Tratz (2011) dataset.[27] They describe three models, called Path-based, Integrated and Integrated-NC. The three models rely on learning so-called path embeddings (cf. Shwartz et al., 2016) for the dependency paths connecting the constituents of a given compound; the dependency paths are extracted from a concatenation of the English Gigaword Corpus (Parker et al., 2011) and a Wikipedia dump. Two of their models (namely Integrated and Integrated-NC) also use word embeddings to represent the constituents of the compound and the Integrated-NC model uses an additional embedding vector to represent the full compound (learned by replacing the constituents of the compounds in the corpus by a single token).

One of the primary motivation of this study is to explore the effect of a

[26]Hermann et al. (2012) use Tratz and Hovy's (2010) dataset in a binary classification task of compositionality, where the compounds are split into two groups, lexicalized and compositional, based on the relation `LEXICALIZED`.

[27]Tratz (2011) introduces a slight variant of the original dataset by Tratz and Hovy (2010); we will return to the distinction between the two datasets in Chapter 3.

phenomenon called lexical memorization.[28] Therefore, they evaluate their models on four different splits of the Tratz (2011) dataset (each consisting of train, development and test sets): (1) random, (2) lexical-full, (3) lexical-mod and (4) lexical-head. The first split includes the full dataset, whereas the last three impose a constraint on the three sets to have distinct vocabulary in the training vs. evaluation subsets (this constraint is only applied on the modifier and head nouns in the case of lexical-mod and lexical-head, respectively). The lexical-full split is similar in spirit to the inter-annotator agreement set by Tratz and Hovy (2010). While their models do not outperform the previously reported results by Dima (2016) on the full dataset (i.e. the random split), they do achieve better results on the other three splits, and hence show that their path-based approach helps mitigate the problem of lexical memorization.

2.4 Conclusion

In this chapter, we have introduced some of the main linguistic definitions and categorizations of noun–noun compounds. However, as evident from our discussion in § 2.1, there has been little agreement on what constitutes a noun compound, not to mention how to interpret it. Early linguistic accounts of compound interpretation took somewhat opposing views on the semantic interpretation of noun–noun compounds, as we explained in § 2.1.2. On the one hand, Levi (1978), inter alios, argued for representing the semantics of a subset of complex nominals using nine abstract relations, and on the other hand, Downing (1977) suggested that there is simply no finite set of compounding relations. We also discussed how compounds, or complex nominals more generally, were categorized in theoretical linguistics according to several criteria, such as lexicalization and novelty (§ 2.1.3), endocentricity and exocentricity (§ 2.1.4) and nominalization (§ 2.1.5). In § 2.1.6, we reflected on the connection between the linguistic and computational studies of noun–noun compounds, arguing that some of the linguistic notions require explicit annotation to be captured in computational models. Moreover, as noted by Spärck Jones (1983) the computational analysis of compounds introduces new problems that are not necessarily part of theoretical studies (e.g. PoS tagging prior to compound identification).

[28] Levy, Remus, et al. (2015) define lexical memorization as the phenomenon in which the classifier learns that a specific word in a specific slot is a strong indicator of the label. We investigate this phenomenon in § 7.6.

In § 2.2, we discussed the five noun–noun compound analysis tasks in NLP, namely: (1) compound identification § 2.2.1, (2) bracketing § 2.2.2, (3) nominal sense disambiguation § 2.2.3, (4) semantic interpretation § 2.2.4 and (5) compositionality prediction § 2.2.5. We do not claim that these tasks cover all possible problems related to compound analysis, but—to the best of our knowledge—past work focused on one or more of the aforementioned tasks. Throughout the discussion, we highlighted that some tasks received more attention than others, such as the compound identification problem which is arguably a foundational step in compound analysis but has been often overlooked in the literature.

We dedicated the final part of this chapter, § 2.3, to an in-depth review of some of the recent NLP studies on inventory-based interpretation of noun–noun compounds. In the following, we summarize the studies we reviewed from two points of views: (1) the taxonomies and datasets and (2) the machine learning models and features they use.

Taxonomies and Datasets: Lauer (1995) approximates the semantics of noun–noun compounds using eight prepositional paraphrases on a dataset of 282 two-word and 244 three-word compounds. Girju et al. (2005) annotated a little over $4,500$ binary compounds and 484 three-word compounds starting with a taxonomy of 35 semantic relations, but they report that only 21 of these relations occur in the dataset. Ó Séaghdha and Copestake (2007) relied on the relations introduced by Levi (1978) to create a coarse-grained inventory of six relations (with two additional levels of granularity), which they then used to annotate $1,443$ two-word compounds extracted from the BNC. Kim and Baldwin (2008) used the set of 20 semantic relations defined by Barker and Szpakowicz (1998) to annotate $2,169$ two-word compounds extracted from the WSJ segment of the PTB. Kim and Baldwin (2013) report bracketing and annotating 1,571 three-word compounds, but only the first dataset, i.e. Kim and Baldwin (2008), is available on-line. Finally, Tratz and Hovy (2010) annotated a relatively large dataset of $17,509$ binary compounds using a taxonomy of 43 semantic relations.

Machine Learning Models and Features: In § 2.3, we showed that noun–noun compound interpretation is, by and large, approached as an automatic classification problem. A wide variety of machine learning models have been already applied to learn compound interpretation, including rule-

based learning algorithms and decision trees like C5.0 (Nastase & Szpakowicz, 2003) nearest neighbor classifiers using semantic similarity based on lexical resources (Kim & Baldwin, 2013), kernel-based methods like SVMs using lexical and relational features (Girju et al., 2005; Ó Séaghdha & Copestake, 2013), maximum entropy with a relatively large selection of lexical and surface form features such as synonyms and affixes (Tratz & Hovy, 2010) and, most recently, neural networks solely relying on word embeddings to represent the compound's head and modifier nouns (Dima & Hinrichs, 2015; Shwartz & Waterson, 2018). It was clear, throughout the literature review, that past work depended heavily on features extracted from lexical resources such as WordNet (Girju et al., 2005; Tratz & Hovy, 2010; Kim & Baldwin, 2013). Interestingly, though, the earlier studies by Lauer (1995); Lapata and Keller (2004) used statistical methods that do not require lexical resources, but the nature of relations they used made this possible; that is, prepositional paraphrases. More recent approaches, however, make use of co-occurrence distributions Ó Séaghdha and Copestake (2013) and distributional semantic models such as word embeddings (Dima & Hinrichs, 2015; Shwartz & Waterson, 2018).

In the following chapter, we take a closer look at some of the existing 'compound-specific' taxonomies and datasets as well as linguistic annotation frameworks that annotate noun–noun compounds as part of a broader, uniform perspective on sentence meaning.

Chapter 3

Annotation of Noun–Noun Compounds and Beyond

The first part of this chapter offers a concise review and analysis of some of the existing noun–noun compound interpretation datasets. In the second part, we introduce more general linguistic annotation frameworks that can be potentially used to derive datasets for compound interpretation. We start with an overview of the existing taxonomies and datasets for compound interpretation in § 3.2. We then turn to appraise three of the publicly available datasets and contrast their annotations on a handful of compounds in § 3.3. In § 3.4, we review several linguistic annotation frameworks that go *beyond* just representing the meaning of compound constructions to phrases and full sentences. In doing so, we explore if and how such resources can be put to use for compound interpretation.

3.1 Introduction

The primary focus of this thesis on noun–noun compound interpretation warrants a dedicated review of the datasets and taxonomies proposed to interpret compounds in NLP. Therefore, in this chapter, we explain how compounds were annotated in past work and, consequently, motivate the need for a new approach and dataset (introduced in Chapter 4) that facilitates the integration of compound interpretation in a broader linguistic and computational perspective. We, thus, distinguish between two types of taxonomies for compound interpretation. We refer to the first type as 'compound-specific'

taxonomies, which are created almost exclusively for noun–noun compound interpretation. The second type is more generic in the sense that it exploits linguistic representation resources concerned with annotating larger constructions, such as full noun phrases and sentences, to derive relations for compound interpretation. Of course, in either case, the taxonomies we consider consist of finite predefined sets of semantic relations, in contrast to the unbounded paraphrase-based approach (cf. § 2.2.4).

Many of the recent NLP studies on compound interpretation rely on taxonomies of relations—and datasets—that are arguably tailor-made for noun–noun compounds. Even though such relations are often theoretically motivated, they ultimately isolate compound interpretation from other work in computational semantics and introduce large variations in terms of specificity and coverage. This variation has long roots in theoretical linguistics, as shown in Chapter 2, but it also means that many of the subsequent studies in NLP often start anew with every new taxonomy of relations. That said, there have been some efforts to map the relations in different taxonomies, e.g. Tratz (2011), but such efforts remain approximative at best for two reasons. First, most studies that use distinct taxonomies also annotate distinct sets of compounds, and hence the relation mapping can only be done by comparing the relations themselves—not the compounds they annotate. Second, highly similar relations might be defined differently across previous studies. In fact, even the same relations can be redefined over time; for example, the organizers of SemEval-2010 Task 8 (Hendrickx et al., 2010) redefine four of the semantic relations introduced in SemEval-2007 Task 4 (Girju et al., 2007) and write that "no complete continuity should be assumed" between the two tasks.[1] All of the above is not a critique of past studies, but it alludes to a—perhaps unavoidable—pattern in the NLP literature on compound interpretation, which leads many scholars to start afresh when studying compound interpretation, and hence making the comparison across their work less straightforward. Therefore, we try to strike a balance between creating yet another taxonomy and enabling cross-framework comparison by deriving taxonomies, and datasets, from established linguistic resources (cf. § 3.4 and Chapter 4).

Although we start the following section with a short survey of the compound interpretation datasets, we do not aim to present a comprehensive

[1] Both SemEval-2007 Task 4 and SemEval-2010 Task 8 are about semantic interpretation of pairs of nominals; we provide more details in § 3.2.

review of past studies in this chapter. Instead, we will only focus on three taxonomies (and their corresponding datasets), chosen in light of the following criteria. First and foremost, the dataset should annotate noun–noun compounds only; that is, in contrast to the datasets that annotate noun–modifier pairs which include adjectival modifiers, such as the dataset by Nastase and Szpakowicz (2003). Second, the dataset should, of course, be available (publicly or otherwise) and contain a 'reasonable' number of compounds (at least 1,000 examples). Third, the dataset cannot be domain-specific to allow comparison with other datasets, and hence we exclude datasets like the one by Rosario and Hearst (2001). Based on these criteria, the three datasets we will review in some detail are the ones of Ó Séaghdha and Copestake (2007), Kim and Baldwin (2008) and Tratz (2011).

3.2 Compound-Specific Datasets

Almost all the studies we reviewed in § 2.3 introduced their own dataset, with the exception of Dima and Hinrichs (2015) and Shwartz and Waterson (2018). Needless to say, however, there are other inventories and datasets in the NLP literature than the ones we mentioned in the previous chapter. Therefore, we complement the review of Chapter 2 by briefly surveying the broader landscape of datasets that were used in past work on compound interpretation. For the sake of completeness, though risking repetition, we will refer to some of the datasets that were already introduced in § 2.3.

Vanderwende (1994) uses a set of 13 *wh*-questions to interpret the relations holding between what she calls noun sequences, e.g. the compound *night attack* is interpreted using the *wh*-question `When`. Vanderwende (1994) defines 'conventional names' for the *wh*-questions that make it easier to compare to other taxonomies of relations; for instance, the conventional names of `When` and `Where` are `Time` and `Locative`, respectively. Lauer (1995) assumes a set of eight prepositional phrases to approximate the relations holding between noun–noun compounds (cf. § 2.3.1). Nastase and Szpakowicz (2003) introduce a dataset of 600 modifier-noun pairs which were manually annotated with a two-level hierarchy of relations, viz. five general relations subdivided into 30 relations (see § A.1 for the full taxonomy with examples). However, this dataset is not limited to noun–noun compounds because the modifier can be a noun, an adjective or an adverb. As explained in § 2.3.2, Girju et al. (2005) use an inventory of 21 relations based on Moldovan et al. (2004) taxonomy of

Dataset	Dataset Size	Taxonomy Size
Vanderwende (1994)	197	13
Lauer (1995)	282	8
Nastase and Szpakowicz (2003)	600	30
Girju et al. (2005)	4,504	21
Ó Séaghdha and Copestake (2007)	1,443	6
Kim and Baldwin (2008)	2,169	20
Tratz and Hovy (2010)	17,509	43
Bos and Nissim (2015)	965	25

Table 3.1: Overview of compound interpretation datasets in NLP. Emphasis is used for the datasets that satisfy the criteria for in-depth review.

35 relations for noun phrases which includes Levi's (1978) complex nominals, genitives and adjective phrases. Ó Séaghdha and Copestake (2007), Kim and Baldwin (2008) and Tratz and Hovy (2010) propose their own taxonomies and annotate different sets of compounds; we will return to reviewing these three datasets in detail below. Bos and Nissim (2015) use a gamification method to annotate a set of 965 noun–noun compounds extracted from the Groningen Meaning Bank (Bos et al., 2017). Following Lauer's (1995) approach, Bos and Nissim (2015) assume that the semantics of noun–noun compounds can be interpreted using prepositions. However, unlike Lauer (1995) who uses just 8 prepositions, Bos and Nissim (2015) start with 26 common English prepositions, 25 of which are attested in the final dataset.

Several SemEval tasks related to the semantic interpretation of "pairs of nominals" have been organized over the past decade.[2] In SemEval-2007 Task 4, Girju et al. (2007) devise a taxonomy of seven relations that occur in previous datasets at the time.[3] Later, in SemEval-2010 Task 8, Hendrickx et al. (2010) define a taxonomy of nine semantic relations, of which four were borrowed from Girju et al. (2007), but the annotation guidelines of these four relations were revised in the 2010 task. According to the annotation guidelines of both tasks, the pair of nominals consist of a common noun and a prenominal modifier which can be a noun or adjective. Hence, these pairs of nominals are not necessarily noun–noun compounds by our definition.

[2]Butnariu et al. (2010) and Hendrickx et al. (2013) organized two SemEval tasks for compound interpretation using paraphrasing verbs and preposition.

[3] The seven relations are: `Cause-Effect`, `Instrument-Agency`, `Product-Producer`, `Origin-Entity`, `Theme-Tool`, `Part-Whole` and `Content-Container`.

Table 3.1 summarizes the datasets we mentioned above. However, it is important to reiterate that not all of the datasets in the table are exclusively concerned with noun–noun compounds. In the following sections, we look more closely at the three of these datasets (highlighted above) that meet the requirements suggested in § 3.1.

3.2.1 Ó Séaghdha and Copestake (2007)

In § 2.3.4, we reviewed the machine learning methods used to learn the dataset by Ó Séaghdha and Copestake (2007). Therefore, we now focus on the dataset itself in terms of its semantic relations as well as the annotation process.

The theoretical framework for the Ó Séaghdha and Copestake dataset is based on the so-called recoverably deletable predicates of Levi (1978) (cf. § 2.1.2). However, Ó Séaghdha and Copestake do not take the relations proposed by Levi as is; instead, they define five desirable criteria according to which they either drop some of the nine relations by Levi (1978), such as `CAUSE` and `MAKE`, or add new ones, such as `INST` and `ACTOR`. The criteria they define assume desirable properties of the relations as well as the taxonomy as a whole, such as coverage and coherence among other criteria which are described by Ó Séaghdha (2008, p. 28–29).[4]

Ó Séaghdha and Copestake (2007) refined their annotation guidelines over the course of six months and finally ended up with six coarse-grained semantic relations (listed in Table 3.2). Five of these six relations are directed, and hence if directionality is taken into account, the total number of relations would be 11; Ó Séaghdha and Copestake refer to these relations as the directed coarse-grained relations. Moreover, each of the six coarse-grained relations has its own set of annotation rules. As part of the annotation process, the annotators specified which of the rules license the relation they selected, and hence the annotation rules themselves can be seen as a more fine-grained set of relations.

To create the dataset, Ó Séaghdha and Copestake (2007) sampled a total of 2,000 binary noun–noun compound types from the BNC, using a simple PoS-based heuristic for compound identification (like Lauer's, cf. § 4.3.1). As will be explained in § 4.3.1, such heuristics can result in false positives, and

[4]The coverage criterion assumes that the relations account for as much data as possible, and coherence requires the relations to describe a coherent concept and the boundaries between such concepts to be clear (Ó Séaghdha, 2008, p. 28).

Relation	%	Example
BE	9.55	monitor screen
HAVE	9.95	mountain summit
IN	15.40	dog box
ACTOR	11.80	class struggle
INST	13.30	hunger strike
ABOUT	12.15	tax exemption
REL	4.05	fashion essentials
LEX	1.75	monkey business
UNKOWN	0.45	simularity crystal
MISTAG	11.00	
NONCOMPOUND	10.60	

Table 3.2: Relations in Ó Séaghdha and Copestake (2007). The relation distributions in this table are based on the numbers reported in Table 3.2 in Ó Séaghdha (2008, p. 40). The percentages in the table are computed over a dataset of 2,000 compounds.

therefore part of the annotation process by Ó Séaghdha and Copestake (2007) is to check whether the compounds are valid.[5] Therefore, Table 3.2 lists the six coarse-grained relations in addition to two classes indicting PoS tagging errors (MISTAG) and invalid compounds (NONCOMPOUND). Furthermore, the table includes three other relations denoting lexicalization (LEX), unknown meaning of the compound (UNKNOWN) and compounds that cannot be interpreted using the six relations (REL).

Parts of the Ó Séaghdha and Copestake (2007) dataset were annotated by two annotators over two stages. First, two batches of 100 compounds were drawn from the full dataset (2,000 compounds) and used in a two-stage annotation training phase to ensure that the two annotators can reach an adequate agreement level. Second, after the training stage, a sample of 500 compounds was annotated by the two annotators leading to an inter-annotator agreement of 66.2% and Cohen's (1960) Kappa score of 0.693. The rest of the dataset was annotated by one annotator only. Although the compounds in the final dataset are distributed out of context, the actual annotation was done in context, i.e. the original sentences from which the compounds were

[5]Indeed, Ó Séaghdha (2008) reports that their heuristic has an accuracy of 78.4% on identifying noun–noun compounds.

extracted were shown to the annotator(s). Moreover, the annotators were not allowed to assign more than one relation per compound, unlike Kim and Baldwin (2008) and Tratz and Hovy (2010) as we will explain below.

As can be seen in Table 3.2, the distribution of the relations in Ó Séaghdha and Copestake's dataset is well balanced. However, the size of the 'classification' dataset—i.e. the compounds assigned one of the six semantic relations—is relatively small (1,443 compounds) compared to other recent datasets.[6] Further, the semantic relations are arguably overly abstract when compared to other taxonomies. For example, the relation `HAVE` is defined by five annotation rules which denote, among other things, possession, group membership and part-whole relations. However, as we will show below, each of these (sub-)definitions or rules is a relation in and by itself in other taxonomies. Similarly, in contrast to other annotation schemes, Ó Séaghdha and Copestake's combines the temporal and locative aspects in one relation, viz. `IN`.

3.2.2 Kim and Baldwin (2008)

Kim and Baldwin (2008) annotated 2,169 binary compounds sampled from the WSJ corpus in the PTB. They also used a PoS-based heuristic to identify noun–noun compounds, which requires manual validation of the identified compounds to exclude false positives. Kim and Baldwin (2008) adopt the taxonomy of 20 relations defined by Barker and Szpakowicz (1998) without modification, which is highly similar to the two-level hierarchical taxonomy by Nastase and Szpakowicz (2003), cf. Table A.1 in Appendix A.

Kim and Baldwin's (2008) annotation scheme differs from Ó Séaghdha and Copestake's in three ways. First, the full dataset by Kim and Baldwin (2008) was annotated by two annotators, in contrast to Ó Séaghdha and Copestake (2007) whose dataset has about one third of the compounds annotated by two annotators. Second, the annotators were not shown extra, context information apart from the compounds themselves. Third, the annotators were allowed to assign more than one relation per compound in case of 'genuine' ambiguity; that is, in the cases where extra contextual information is needed to interpret the compound and assign a semantic relation; Kim and Baldwin (2008) give

[6] Ó Séaghdha (2008) distinguishes between the annotation dataset and the classification dataset. The latter, has a total of 1,443 compounds that are annotated by one of the six semantic relations and it is this dataset Ó Séaghdha and Copestake use in subsequent classification experiments. Whereas the annotation dataset consists of 2,000 compounds but these include lexicalized compounds, tagging errors and false positives.

Relation	%	Example
TOPIC	39.05	music star
PURPOSE	13.78	trade union
OBJECT	7.61	lawsuit settlement
SOURCE	7.33	crop problem
PROPERTY	7.15	prescription drug
CAUSE	5.49	death notice
CONTENT	3.23	debt collection
POSSESSOR	2.44	company car
PRODUCT	2.44	liquor industry
TIME	2.03	lunch break
LOCATION	1.84	hotel lobby
LOCATED	1.15	horse barn
MATERIAL	1.15	paper product
CONTAINER	1.15	committee member
EQUATIVE	1.15	fountain pen
INSTRUMENT	0.83	laser printer
BENEFICIARY	0.69	consumer price
RESULT	0.69	oil spill
AGENT	0.64	court decision
DESTINATION	0.14	space shuttle

Table 3.3: Relations in Kim and Baldwin (2008). The distributions were calculated on the full dataset, i.e. the combination of the train and test splits.

cotton bag as an example of genuine ambiguity, which can be interpreted as a bag made of cotton (MATERIAL) or a bag for cotton (PURPOSE).

Kim and Baldwin (2008) report an annotator agreement of 52.31%, after training the annotators on 200 compounds which are not part of the original dataset. However, they do not report other scores to account for agreement by chance, such as Cohen's Kappa measure. In addition, Kim and Baldwin (2008) state that in the cases where the annotators assign more than one relation per compound, agreeing on at least one relation was considered as an agreement.[7]

Table 3.3 lists the semantic relations attested in Kim and Baldwin's (2008) dataset and their frequencies. As can be seen from the table, almost 39% of the compounds in their dataset are annotated with the relation TOPIC. Further,

[7] 8.2% of the compounds in Kim and Baldwin's dataset are annotated with more than one relation.

the five most frequent relations account for three quarters of the compounds in the dataset, which obviously leaves the least frequent relations with very few examples. Even though the inventory of relations used by Kim and Baldwin (2008) is more fine-grained than the one used by Ó Séaghdha and Copestake (2007), there are still some relations that can be considered somewhat abstract. For example, in experimental work preceding the documentation of the dataset proper, Kim and Baldwin (2006) considered the relation `PROPERTY` too general and decided to exclude it from the dataset in their experiments on compound interpretation using verb semantics.

3.2.3 Tratz and Hovy (2010)

When we introduced Tratz and Hovy's (2010) work in § 2.3.5, we reported that they created a dataset of 17,509 noun–noun compounds annotated with 43 semantic relations organized in ten groups. However, the dataset later published by Tratz (2011) includes 19,158 noun–noun compounds annotated with a revised taxonomy of 37 relations.[8] We distinguish between these two versions of the dataset by referring to the former as the Tratz and Hovy (2010) dataset and the latter as the Tratz (2011) dataset.

Tratz and Hovy (2010) proposed a new taxonomy of relations, which they refined throughout five rounds of annotation—each consisting of 100 examples—to guarantee more consistency in annotation (i.e. to improve inter-annotator agreement). The annotators were shown the compounds in different contexts and asked to select one or two relations per compound. These annotation rounds informed the adjustments that Tratz and Hovy (2010) made incrementally on their taxonomy; for example, they excluded the relation `PURPOSE` because it was found to be ambiguous with other categories and introduced more fine-grained relations that generally express purpose. Tratz and Hovy (2010) relied on the Amazon Mechanical Turk (MTurk) service (a crowd-sourcing 'micro-working' platform) for those rounds of annotation.[9]

[8] Tratz (2011) revised the taxonomy, and annotation, of the compound dataset "to allow for a better mapping between prepositions and noun compound relations" (Tratz, 2011, p. 76). This led to conflating some of the relations into one, and hence the smaller number of relations in the revised dataset.

[9] Tratz and Hovy (2010) highlight the linguistic drawbacks of using such a platform, for example they could not make sure that the so-called 'Turkers' (the annotators in this case) are native speakers of English. More importantly, however, MTurk is not merely a crowd-sourcing platform, it is an unregulated labor market, which raises ethical concerns about its use in academic research and otherwise. Fort et al. (2011) discuss some of the

Tratz and Hovy (2010) performed a final inter-annotator agreement study to evaluate the quality of their taxonomy on 150 compounds that were not part of the original dataset. They also relied on the MTurk service for the agreement study, but they accounted for the fact that MTurk is not ideal for inter-annotator agreement studies by applying several constraints and weighting methods, which are fully explained in Section 6 in Tratz and Hovy (2010). They measured agreement based on their own annotation of the 150 compounds and the MTurk annotations, and report a combined agreement score of 59% on the full taxonomy and 0.57 Kappa score.[10]

In Table 3.4 we compile the relations, and frequencies, from the dataset made available by Tratz (2011), and hence the relations and their distribution are different from the ones reported in Tratz and Hovy (2010).[11] For example, Table 1 in Tratz and Hovy (2010) lists seven sub-relations under `TOPIC`, but we only observe three relations of this type in Table 3.4, including the generic relation `TOPIC`. This suggests that some of the more specific relations were conflated under the more general `TOPIC` relation. The 37 relations are grouped under 12 types of relations, which are indicted as numbers in the left-most column of Table 3.4. The numbers refer to the following groups: (1) Objective, (2) Causal, (3) Purpose, (4) Ownership, Employment and Use, (5) Time, (6) Location and Whole+Part, (7) Composition and Containment, (8) Topical, (9) Attributive and Equative, (10) Other Complements, (11) Personal Name and Title and (12) Other. The relations under the eleventh group (Personal Name and Title) are not described by Tratz (2011) but they are attested in the dataset. We refer to Appendix B in Tratz (2011) for full definitions of the relations listed in Table 3.4.

As can be seen from Table 3.4, the distribution of the relations is comparatively balanced, with the most frequent relation accounting for 17.13% of the data, viz. `OBJECTIVE`.[12] It is noteworthy, however, that the six most frequent relations in Tratz's (2011) dataset seem to be the least specific among the 37 relations in the taxonomy. These relations are `OBJECTIVE`,

issues with MTurk, such as the very low wages and complete lack of labor rights.

[10]The agreement scores Tratz and Hovy (2010) report vary depending on the annotator, as they hired several annotators via MTurk, but here we only consider the combined weighted agreement score of all annotators.

[11]The dataset is distributed as part of the following package `https://www.isi.edu/publications/licensed-sw/fanseparser`. Accessed 5 June 2015.

[12]According to Tratz (2011, p. 197) the relation `OBJECTIVE` describes "the logical grammatical object", which we find to be more of a syntactic definition than a semantic one.

	Relation	%	Example
1	OBJECTIVE	17.13	tax claim
2	SUBJECT	3.54	press report
	CREATOR-PROVIDER-CAUSE_OF	1.52	fire damage
	JUSTIFICATION	0.26	genocide trial
	MEANS	1.50	phone interview
3	PERFORM&ENGAGE_IN	11.48	peace effort
	CREATE-PROVIDE-GENERATE-SELL	4.83	wine shop
	OBTAIN&ACCESS&SEEK	0.86	finance plan
	MITIGATE&OPPOSE	0.78	tax relief
	ORGANIZE&SUPERVISE&AUTHORITY	1.60	election committee
	PURPOSE	1.95	exchange system
4	OWNER-USER	2.12	company car
	EMPLOYER	2.32	state lawyer
	EXPERIENCER-OF-EXPERIENCE	0.53	team spirit
	USER_RECIPIENT	1.03	worker salary
5	TIME-OF1	2.17	night club
	TIME-OF2	0.47	watermelon season
6	LOCATION	5.16	street lawyer
	WHOLE+PART_OR_MEMBER_OF	1.69	staff lawyer
7	CONTAIN	1.21	oil tank
	SUBSTANCE-MATERIAL-INGREDIENT	2.64	plastic pencil
	PART&MEMBER_OF_COLLECTION&CONFIG&SERIES	1.78	customer base
	VARIETY&GENUS_OF	0.10	ant species
	AMOUNT-OF	0.88	tax level
8	TOPIC	7.02	wage talk
	TOPIC_OF_EXPERT	0.68	film scholar
	TOPIC_OF_COGNITION&EMOTION	0.31	market optimism
9	EQUATIVE	5.43	drug charge
	ADJ-LIKE_NOUN	1.33	mass exodus
	MEASURE	4.21	month time
	PARTIAL_ATTRIBUTE_TRANSFER	0.35	chocolate lab
10	RELATIONAL-NOUN-COMPLEMENT	5.62	drug epidemic
	WHOLE+ATTRIBUTE&FEATURE&QUALITY	0.29	monopoly power
11	PERSONAL_NAME	0.52	barack obama
	PERSONAL_TITLE	0.53	mister bond
12	LEXICALIZED	0.78	drag queen
	OTHER	5.36	contact lense

Table 3.4: Relations in Tratz (2011). The relations and their frequencies were extracted from the dataset itself.

PERFORM&ENGAGE_IN, TOPIC, RELATIONAL-NOUN-COMPLEMENT, EQUATIVE and LOCATION. Tratz and Hovy (2010) map their taxonomy of relations to other taxonomies, such as the ones by Barker and Szpakowicz (1998) and Girju et al. (2005), and demonstrate that their relations are similar to the ones defined in past work.

3.3 Contrastive Analysis

One way to compare the three datasets we introduced above is to inspect how they annotate a selection of noun–noun compounds that are common to the three of them. Upon inspecting the three datasets, we find that there are only 33 compounds in common; Table 3.5 lists these 33 compounds and their annotation in the datasets of Ó Séaghdha and Copestake (2007), Kim and Baldwin (2008) and Tratz (2011). Obviously a sample of 33 compounds is not enough to perform a quantitative analysis of the approximate agreement between the three annotation schemes. Moreover, a proper qualitative analysis would mandate a comprehensive review of the annotation guidelines of the three datasets, which is something that not only falls outside the scope of our work, but is also not readily available, at least in the case of Kim and Baldwin's (2008) dataset.

Of the 33 compounds, 15 are of type TOPIC (i.e. 45.45%) in Kim and Baldwin's dataset, which is unsurprising given that 40% of the compounds in their dataset are annotated with that relation. However, this makes us more reluctant to read much into the 'patterns' we observe in Table 3.5. For example, Ó Séaghdha and Copestake's IN, INST and ACTOR relations annotate many of the same compounds as Kim and Baldwin's TOPIC, but this is likely to be a by-product of the latter relation being very frequent in Kim and Baldwin's dataset. Given that the TOPIC relation seems to cover a broad concept, we exclude the compounds annotated with this relation from our analysis.

The compounds annotated with the relation HAVE by Ó Séaghdha and Copestake (i.e. examples 11 – 16 in Table 3.5) receive somewhat similar relations in Kim and Baldwin (2008) and Tratz (2011). Ó Séaghdha (2008) defines five rules that license the relation HAVE which express: possession, part-whole, group-member, condition-experiencer and property-object. These rules encompass most of the relations observed in the other two datasets such as WHOLE+PART_OR_MEMBER_OF, POSSESSOR and CONTAINER. Tratz's relation

	Compound	Ó&C	K&B	T
1	home market	IN	TOPIC	PURPOSE
2	stock market	IN	TOPIC	PURPOSE
3	golf course	IN	TOPIC	PERFORM&ENGAGE_IN
4	office space	IN	LOCATED	LOCATION
5	video game	INST	TOPIC	MEANS
6	computer software	INST	TOPIC	LOCATION
7	computer system	INST	TOPIC	EQUATIVE
8	jet engine	INST	CONTENT	WHOLE+PART_OR_MEMBER_OF
9	aid package	INST	CONTENT	TOPIC
10	printing press	INST	PROPERTY	PERFORM&ENGAGE_IN
11	committee member	HAVE	CONTAINER	WHOLE+PART_OR_MEMBER_OF
12	family member	HAVE	CONTAINER	WHOLE+PART_OR_MEMBER_OF
13	union member	HAVE	POSSESSOR	WHOLE+PART_OR_MEMBER_OF
14	department store	HAVE	TOPIC	CONTAIN
15	world economy	HAVE	TOPIC	LOCATION
16	customer base	HAVE	CONTENT	PART&MEMBER_OF_COLLECTION...†
17	machine tool	BE	MATERIAL	EQUATIVE
18	junk bond	BE	PROPERTY	EQUATIVE
19	parent company	BE	POSSESSOR	EQUATIVE
20	crystal chandelier	BE	CONTENT	SUBSTANCE-MATERIAL-INGREDIENT
21	government report	ACTOR	SOURCE	SUBJECT
22	employment service	ACTOR	TOPIC	EQUATIVE
23	government official	ACTOR	TOPIC	EMPLOYER
24	aerospace division	ACTOR	TOPIC	OTHER
25	engineering group	ACTOR	TOPIC	PERFORM&ENGAGE_IN
26	advertising agency	ACTOR	TOPIC	PERFORM&ENGAGE_IN
27	research team	ACTOR	PURPOSE	PERFORM&ENGAGE_IN
28	state legislature	ACTOR	PURPOSE	EMPLOYER
29	board meeting	ACTOR	CONTENT	SUBJECT
30	property tax	ABOUT	TOPIC	OBJECTIVE
31	interest payment	ABOUT	CAUSE	OBJECTIVE
32	takeover offer	ABOUT	PURPOSE	OBJECTIVE
33	oil price	ABOUT	SOURCE	RELATIONAL-NOUN-COMPLEMENT

Table 3.5: The set of compounds found in Ó&C: Ó Séaghdha and Copestake (2007); K&B: Kim and Baldwin (2008) and T: Tratz (2011). † Shortened form of PART&MEMBER_OF_COLLECTION&CONFIG&SERIES

LOCATION might seem out of place at first glance, but Tratz and Hovy (2010, p. 681) report that the relations "LOCATION and WHOLE + PART/MEMBER OF were commonly disagreed upon by Turkers so they were placed within their own taxonomic subgroup".

Per definition, the relations BE and EQUATIVE in Ó Séaghdha and Copestake (2007) and Tratz (2011), respectively, are highly similar. In both studies, the two relations can express sub-types as well as material. These similarities hold in examples 17 – 19 in Table 3.5, but as we mentioned before our sample is very small to draw any conclusions. Therefore, we cannot judge to what degree this observation indicates consistency across the different annotation schemes. The same applies for the relations ABOUT and OBJECTIVE in Ó Séaghdha and Copestake (2007) and Tratz (2011), respectively. Furthermore, the six compounds annotated as INST (examples 5 – 10) by Ó Séaghdha and Copestake receive six different relations in Tratz's dataset. Due to the small size of the sample at hand—again—we find it difficult to determine whether the mapping of INST to many other relations in Tratz's dataset is a mere coincidence or a pattern.

The discussion above illustrates the problem with comparing existing taxonomies and datasets for noun–noun compound interpretation. That is, we do not have enough data to perform a quantitative analysis, simply because the datasets often annotate largely distinct sets of compounds. To quote Ó Séaghdha (2008, p. 30), a "definitive comparison of multiple schemes would require annotation of a single corpus with every scheme, but in practice this is rarely done." This observation serves as one of our motivations to create a new dataset that makes it possible to compare annotations across frameworks by considering resources that annotate the same corpus (cf. Chapter 4). Therefore, in the following section we explore if and how existing linguistic resources can be used to derive a new dataset for noun–noun compounds that addresses the aforementioned problem, among others.

3.4 Linguistic Annotation Frameworks

The rest of this chapter explores how noun–noun compounds are analyzed within linguistic meaning representations that operate at phrase- or sentence-level. In contrast to the compound-specific datasets, these meaning representation frameworks consider semantic structures across a broader range of phenomena and different syntactic realizations. Therefore, they are often

pushed in different directions, which may lead to broader and stronger generalizations. For the same reasons, however, these frameworks may have less specialized annotations for compounds, compared to the datasets we reviewed in the first part of this chapter.

By introducing some of the frameworks below we set the scene for the following chapter, but we will leave the particularities of deriving a dataset for noun–noun compound analysis from these frameworks to the next chapter.

3.4.1 NomBank

NomBank (Meyers, 2007) is a sister project of the Proposition Bank or PropBank (Palmer et al., 2005) where the former annotates the argument structure for nominal predicates and the latter focuses on the argument structure of verbs. Both resources annotate the WSJ corpus of the PTB (in addition to other corpora for PropBank), and hence provide an additional layer of shallow meaning representation on top of the syntactic representations in the PTB.[13] NomBank partly builds on PropBank to populate its lexical entries and preserves some consistency across the two resources; for example, the default argument structure of verbal nominalizations in NomBank reuses the argument labels of the underlying verb in PropBank (Meyers et al., 2004).

NomBank set out to annotate all 'markable' noun phrases in the WSJ corpus. For a noun phrase (NP) to be markable, it must of course contain an argument-taking common noun (i.e. a propositional noun expressing an event, relation or state) and at least one argument of the head noun, among other conditions (Meyers, 2007, p. 7). According to Meyers et al. (2004), argument-taking nouns need not be verbal nominalizations; in fact, they define 16 classes of argument-taking nouns, in addition to verbal and adjectival nominalizations. The 16 classes include relational nouns, such as *director* and *husband*, and partitive nouns, such as *set* and *cascade*. As mentioned above, to define the nominal argument structures, NomBank draws on the predicate argument structures in PropBank, sometimes even when there is no direct

[13]The need for an explicit representation of argument structures is motivated by the fact that the argument structure of a given predicate often corresponds to multiple syntactic realizations, due to verb alternations or diathesis alternations (Levin, 1993). Hence the mapping between the grammatical subject or object and semantic roles such as agent and patient or theme need not be constant for a lemma; for example, in *The boy broke the window* and *The window broke* we have two grammatical analyses of *the window* (object in the former and subject in the latter) even though *the window* plays the same semantic role in both cases.

morphological connection between the nominal predicate and the verbal one. For example, NomBank assumes that *aggression* takes the same argument structure as the verb *attack* in PropBank.

NomBank views complex proper nouns (or proper noun phrases) as "unanalyzable wholes", and hence they are considered unmarkable (i.e. they were not annotated), cf. Meyers (2007, Section 3.2). In addition, most idioms are not markable in NomBank with the exception of those that can be interpreted literally such as *a grain of salt*. However, NomBank does annotate so-called dead metaphors (i.e. lexicalized metaphors) such as *tapestry* which is often used a synonym for *combination*.

In concrete terms, NomBank defines a frame entry for each nominal predicate that specifies its argument and allowed adjuncts, following the same style used in PropBank. Each frame lists the possible 'rolesets' of the predicate; that is, the possible sets of arguments a predicate can take. In some cases, the different senses of a polysemous noun can give rise to distinct rolesets, but this is far from being the rule in NomBank. Meyers (2007, p. 23) clearly states that "the same sense of a noun seems to participate with different sets of roles." Further, NomBank does not distinguish between "closely related senses of a noun unless they take incompatible sets of argument roles" (Meyers, 2007, p. 23). The arguments and adjuncts in NomBank are also defined following the PropBank style; arguments are numbered (`ARG0`, `ARG1`, `ARG2`, ...), and some of them can be generalized to prototypical semantic roles such as agent and patient. Adjuncts (or modifiers) are expressed using the label `ARGM` and are further specified via so-called function tags, which denote whether a modifier is temporal `ARGM-TMP`, locative `ARGM-LOC` or directional `ARGM-DIR` among other types. Meyers (2007, p. 90) defines a total of twelve function tags, to which we will return in the following chapter.

The actual NomBank annotation of the WSJ corpus consists of a list of propositions which specify the predicate and its arguments (and adjuncts) in a given sentence, as shown in the following example:[14]

(3.1) State officials

ARG2 = state, ARG0 = official, REL = official

Meyers et al. (2004) report that there are approximately 200,000 instances of markable nouns in the WSJ segment of the PTB; according to their

[14]In this example, the predicate *official* is also annotated as the `ARG0` of itself, which is an instance of so-called incorporated arguments in NomBank (Meyers, 2007, p. 8).

estimates, 5,000 of these instances are adjectival nominalizations and the rest are evenly divided between verbal nominalization and the other noun classes they define. Further, Meyers et al. (2004) report an initial estimate of 6,500 lexical entries for argument-taking common nouns in the PTB. The number of instances that were eventually marked in NomBank version 1.0 (released December 17, 2007) is 114,576, the rest of the markable instances do not actually take arguments in the corpus. Similarly, the final number of frames or lexical entries representing argument-taking nouns is 4,704.

Some of the noun phrases annotated in NomBank constitute, partially or fully, noun–noun compounds. Therefore, the same relations (i.e. arguments and adjuncts) used to annotate noun phrases also apply for noun–noun compounds. Assuming a method to identify compounds in the WSJ text, we can extract the annotation of the compounds in NomBank and create a new dataset whose relations are the arguments and adjuncts defined in NomBank. The details of how this can be done in practice are laid out in the following chapter.

3.4.2 PCEDT 2.0

The Prague Czech–English Dependency Treebank 2.0 (PCEDT; Hajič et al., 2012) is a manually annotated parallel treebank of the same WSJ text in PTB and its Czech translation. Unlike NomBank, the PCEDT annotates the full sentences using multiple interlinked layers of representation based on the valency theory of the Functional Generative Description framework in theoretical linguistics (FGD; Sgall et al., 1986). The representation layers in PCEDT start from the surface form with the bottom-most layer, the word layer (w-layer), which contains a tokenized version of the plain text. The morphological layer (m-layer) adds morphological information on the tokens in the w-layer such as PoS tags and lemmas. The analytical layer (a-layer) contains a parsed version of the text in the form of dependency trees. The tectogrammatical layer (t-layer), which is the topmost layer, captures the meaning of the sentence by representing both (deeper) syntactic and semantic relations in the sentence. In addition, the English part of PCEDT includes the original annotation from the PTB, which are represented in an additional layer called the phrase-structure layer (p-layer).

We will only focus on the t-layer, as we are mainly interested in understanding how the semantics of noun–noun compounds are represented

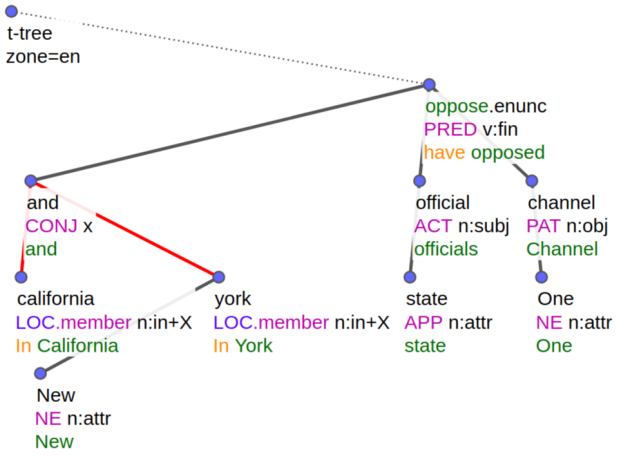


Figure 3.1: The PCEDT tectogrammatical annotation of "In California and New York, state officials have opposed Channel One."

in PCEDT. The tectogrammatical representation is, in practice, a rooted dependency tree structure with different types of nodes, edges and attribute values (within the nodes). Figure 3.1 shows an example of the t-layer annotation in PCEDT.[15] As Hajič et al. (2012) explain, only content words and coordinating conjunctions are represented as nodes in the t-layer (the linguistic contribution of function words is stored as attributes in the inner structure of the nodes). PCEDT defines eight types of nodes in the t-layer that have different functions and inner structures, e.g. technical root, complex and atomic nodes. The edges in the t-tree are unlabeled (i.e. they bear no description), but their types are determined based on the daughter node's type. The inner structure of each node consists of attribute-value pairs that vary depending on the node type; such attributes include information on the morphosyntactic realization of the node, whether the node is a member of coordination, among many other attributes. Of the numerous attributes in the t-layer nodes, we are concerned with the functor attribute which describes the syntactico-semantic relation between the node and its parent.

According to the PCEDT annotation manual, functors represent the semantic values of syntactic dependency relations (Cinková et al., 2006, p. 107). In other words, functors are essentially semantic labels between a lexical unit (predicate) and its valency complementations, which are defined in the

[15] Figure 3.1 is generated using the Treex Web interface, `https://lindat.mff.cuni.cz/services/treex-web`.

PCEDT English Valency Lexicon (EngValLex; Semecký & Cinková, 2006). Functors in PCEDT can be grouped according to several criteria; one such grouping is based the valency criterion which divides functors into arguments and adjuncts. Argument functors include `ACT` (Actor), `PAT` (Patient) and `ADDR` (Addressee) among others. PCEDT defines several types of adjunct functors such as temporal functors (e.g. `TWHEN`, `TTILL`), locative and directional functors (e.g. `LOC`, `DIR1`) and implicational functors (e.g. `AIM`, `CAUS`). Cinková et al. (2009) report that about 70 functor types are used in the PCEDT annotation. However, not all of these functors occur in noun–noun dependency relations, which is the part of the sentence annotation that concerns us. We will return to the functors attested in noun–noun dependency relations in the following chapter (§ 4.5).

Given the whole-sentence meaning representation in PCEDT, it follows that PCEDT annotates noun–noun compounds, and hence we can expect to derive a dataset for noun–noun compounds from PCEDT. In fact, in the annotation manual of PCEDT, Cinková et al. (2006, p. 162) clearly assert that "nouns as modifiers are interpreted according to their semantic relation towards the governing noun." Further, Cinková et al. (2006) define specific annotation rules for deverbal nominalizations, based on a set of suffixes typical of deverbal nouns, which assumes that some forms of nominalizations inherit their argument structure from the underlying verb.

Deriving a noun–noun compound dataset from PCEDT, thus, amounts to identifying the compounds in the WSJ corpus and extracting their functors from the PCEDT's t-layer. For example, considering Figure 3.1 again, we see that it includes a noun–noun compound, viz. *state officials*, and the relation between its constituents is the functor `APP` (which denotes the appurtenance of one person or thing to another).

3.4.3 Other Resources

In the same general spirit of this second part of the chapter, we further review other candidate resources that can be potentially used to study noun–noun compound interpretation. For reasons that will become clear below, however, these resources have not taken a central role in our work, and thus are discussed somewhat more tersely.

```
(h / have-org-role-91
    :ARG1 (p / police~e.26)
    :ARG2 (o / official~e.27))
```

Figure 3.2: AMR analysis of *police official*

Abstract Meaning Representation: The Abstract Meaning Representation (AMR; Banarescu et al., 2014) is a formalism for English whole-sentence meaning representation. In practice, AMRs are directed, acyclic and rooted graphs, whose nodes represent concepts (or constants) and edges represent the relation between the nodes. The AMR concepts are in no small part based on PropBank frames. In theory, one can identify the noun–noun compounds in the AMR copora and extract their corresponding relations from the AMR graphs. However, since one of the basic principles of AMR is to "abstract away from syntactic idiosyncrasies" (Banarescu et al., 2014, p. 178), the AMR graphs are, in practice, not aligned with surface form tokens.[16] Hence, in order to derive an AMR-based dataset for noun–noun interpretation, one would first need to align the tokens to the AMR graph nodes.[17] Even if we assume token-aligned graphs, it is not always clear how to extract the noun–noun compound analysis from AMR. For example, the constituents of the compound *police official* are analyzed as arguments to a non-lexical concept `have-org-role-91` (which expresses organizational membership).[18] The sub-graph in Figure 3.2 shows the analysis of *police official* in AMR.

DeepBank: DeepBank (Flickinger et al., 2012) annotates the WSJ corpus in PTB with the English Resource Grammar (ERG; Flickinger, 2000), which is a broad-coverage, linguistically precise grammar for English written within the linguistic framework of Head-Driven Phrase Structure Grammar (HPSG; Pollard & Sag, 1994). By design, these analyses limit themselves to what is often called sentence meaning, i.e. the constraints on interpretation imposed

[16] AMR 2.0 (released on June 15, 2017) includes *automatically* generated alignments, whereas AMR 1.0 (released on June 16, 2014) does not provide any form of alignments. We evaluated the use of AMR for creating a dataset for compound interpretation before its second release. Needless to say, the issue of string-to-graph alignment remains unresolved in AMR 2.0, because it was generated automatically. Therefore, we argue that AMR 2.0 is still not ideal for creating a high-quality AMR-based compound dataset.

[17] In fact, aligning tokens or strings to AMR graphs has emerged as a separate task in and of itself in NLP (Pourdamghani et al., 2014; Lyu & Titov, 2018).

[18] A non-lexical concept is an abstract concept that does not directly correspond to a token in the sentence.

by grammatical structure. Thus, noun–noun compounds in DeepBank are annotated with a very underspecified relation called `compound`, and hence DeepBank does not commit to any semantic interpretation of noun–noun compounds. That said, DeepBank does annotate the internal structure of multi-word compounds, and we will exploit this information in Chapter 4.

3.5 Conclusion

In this chapter, we focused on the semantic interpretation of noun–noun compounds in terms of the datasets and taxonomies of relations proposed in previous NLP studies. Evidently, there is no lack of compound interpretation datasets, as shown in our brief review of such datasets in § 3.2. However, much like the theoretical studies on compounding, past work in NLP often assumes varying definitions of compounds and how they ought to be interpreted, resulting in a mixed bag of taxonomies and datasets. The variability of the datasets and taxonomies is not a bad thing per se, but it does require extra efforts to align and compare past work. Tratz (2011), for example, presented an *approximate* mapping of the past inventories of relations, but such a mapping cannot be evaluated quantitatively because most of the existing datasets annotate disjoint sets of compounds.

Of the existing datasets, we chose to review three datasets that annotate noun–noun compounds exclusively, among other criteria; viz. the datasets by Ó Séaghdha and Copestake (2007), Kim and Baldwin (2008) and Tratz (2011). We showed, in § 3.2, that these three datasets make varyingly different assumptions starting from the annotation guidelines to the granularity of the relation they adopt. Ó Séaghdha and Copestake's dataset is remarkably motivated by a theoretical framework (i.e. that of Levi, 1978), in contradistinction to Tratz and Hovy (2010) who relied on 'crowd-sourcing' to refine their relatively large taxonomy of relations. The relations in Ó Séaghdha and Copestake (2008) are perhaps the most abstract among the three datasets and the dataset itself is the smallest in size. Kim and Baldwin's (2008) dataset is slightly larger in size in terms of the number of compounds and relations. However, the distribution of the relations is relatively skewed where the five most frequent relations annotate 75% of the compounds in the dataset. The dataset by Tratz (2011) is by far the largest one with respect to the number of relations as well as compounds. As noted in § 3.2.3 though, the more specific a relation is the less frequent it is; for example, the relation `MITIGATE&OPPOSE` has a frequency

of 0.78% compared to an 11.48% frequency for PERFORM&ENGAGE_IN, both relations broadly express purpose. Overall, each of these datasets comes with its advantages and disadvantage; for example, while Ó Séaghdha and Copestake's dataset is framed by a linguistic theory, its size poses a limitation on machine learning approaches. Tratz and Hovy's dataset, on the other hand, includes a sizable number of compounds, but it is perhaps less linguistically motivated, with respect to its inventory of relations as well as annotation scheme (i.e. the use of MTurk for semantic annotation).

To contrast the three datasets, in § 3.3 we analyzed the subset of compounds the three datasets annotate. We found that there are only 33 compounds that are common to the three datasets, which obviously limits our contrastive analysis to mere observations. For example, even though we observed some potential cross-dataset consistency in relations that express possession and group membership, we cannot conclude whether or not this observation is indeed a pattern. This uncertainty, coupled with the fact that none of the datasets (and taxonomies) seems to be advantageous on all fronts, motivates our approach to consider how existing linguistic representation resources deal with noun–noun compounds in § 3.4.

We introduced two well-established meaning representation frameworks, NomBank and PCEDT (Sections 3.4.1 and 3.4.2, respectively), which can be exploited to derive a new dataset for compound interpretation. NomBank annotates the argument structure of nominal predicates in noun phrases. PCEDT, in contrast, is a multi-layered whole-sentence meaning representation framework. We showed that noun–noun compounds are inherently annotated as part of broader constructions in both resources. The fact that PCEDT annotates the same corpus as NomBank (viz. the WSJ corpus in PTB) makes these two resources all the more interesting. We also briefly discussed the potential of using two other meaning representation resources; namely, AMR and DeepBank. We found that extracting compound interpretation from AMR graphs presupposes an alignment between the tokens and AMR concepts, which adds an error-prone step to deriving a compound dataset from AMR. DeepBank does not provide semantic interpretation for noun–noun compounds, but it remains a valuable resource when it comes to compound bracketing (cf. § 4.4).

In the following chapter, we present the actual process of creating a new dataset with dual annotation from NomBank and PCEDT, among other resources.

Chapter 4

Resource Creation

In this chapter, we describe the process of deriving a new noun–noun compound dataset from a selection of well-established general-purpose linguistic resources. We define several versions of our dataset that include syntactic and semantic annotation of compounds. In addition, we present a comprehensive study of compound identification as a prerequisite step towards creating a new compound dataset from annotations over running text. We start the chapter, in § 4.1, by motivating the need for a new dataset and explain the merits of the approach we take to derive it. In § 4.2, we lay out the general technical framework we use throughout the chapter. In § 4.3, we present and discuss several compound identification methods and then use them to identify noun–noun compounds in the WSJ corpus of the PTB. In § 4.4, we extract the bracketing of the multi-word compounds in our dataset from three linguistic resources and evaluate the bracketing agreement between them. Lastly, in § 4.5, we add semantic relations to the compounds in our dataset from two meaning representation frameworks and discuss how these annotation relate to each other.

Parts of this chapter build on previously published work in Fares et al. (2015) and Fares (2016).

4.1 Introduction and Motivation

In Chapters 2 and 3, we showed that there is little agreement on how to interpret or represent the semantics of noun–noun compounds, whether in theoretical linguistics or NLP. Consequently, many taxonomies and datasets

emerged throughout the decades of research on the topic, each with its own merits and shortcomings (cf. § 3.2). However, which taxonomy (and dataset) to use remains—by and large—an open research question and is highly dependent on downstream applications. In an attempt to answer this question, we take a new perspective on compound interpretation that seeks to 'integrate' this task within broader established meaning representation frameworks. Therefore, we dedicate this chapter to introducing our new noun–noun compound dataset and the process of deriving it from existing linguistic resources.[1]

The need for a new dataset is motivated by several key reasons, some of which were already explained in the previous chapter. First, the existing compound datasets assume distinct taxonomies of relations and annotate distinct sets of compounds—which makes it difficult to compare them, as we highlighted above. Our new dataset aims to allow cross-framework comparison by using resources that annotate the same text (and hence the same compounds), such as NomBank and PCEDT. The second reason pertains to the usability and integration of compound interpretation in other NLP tasks or applications. As Copestake and Briscoe (2005, p. 130) appositely note, "without consideration of the integration issues and the purposes of compound processing, any definition of a target for noun compound analysis is somewhat arbitrary." Given that almost all of the existing taxonomies are compound-specific (i.e. tailor-made for compounds), it is unclear how they can be utilized beyond the task of compound interpretation itself. Therefore, with this consideration in mind, we believe that our new dataset opens up new possibilities for integrating compound interpretation within other NLP tasks, because it relies on annotation frameworks that are not specific to noun–noun compounds. Likewise, in order to build a compound processing pipeline that handles both bracketing and semantic interpretation of noun–noun compounds, we need a dataset that includes both types of information. Moreover, on a theoretical level, creating a new dataset based on existing meaning representations can help further our understanding of noun–noun compounds as a linguistic construction inherent to some meaning representation 'schemes'. Lastly, on a more practical level, the parallel annotations in our dataset facilitates experimentation with machine learning methods such as multi-task learning as we will show in Chapter 7.

[1]We use 'linguistic resources' as a general term to refer to meaning representation formalisms, treebanks and computational lexica.

4.2 Overview

This section gives a brief overview of our approach to automatically construct a bracketed and semantically annotated dataset of noun–noun compounds from several linguistic resources. The construction process consists of three main steps that mirror three of the tasks introduced in § 2.2; namely, (1) compound identification, (2) compound bracketing and (3) compound interpretation. For each of these steps, we will use several linguistic resources that provide lexical, syntactic and semantic annotations for (parts of) the WSJ corpus of the PTB, viz. NomBank, the English part of PCEDT 2.0, DeepBank and, of course, the PTB itself. However, since the original PTB leaves the internal structure of noun phrases unannotated, we will also use the noun phrase annotations by Vadas and Curran (2007). More specifically, we will first identify noun–noun compounds in the WSJ corpus using a selection of compound identification methods detailed in § 4.3. Secondly, in § 4.4, we will extract the bracketing of the identified compounds from three different resources: the PTB noun phrase annotation by Vadas and Curran (2007), DeepBank and PCEDT. Lastly, in § 4.5, we will extract the semantic relations of two-word compounds as well as bracketed multi-word compounds from two resources: PCEDT and NomBank.

On a more technical level, we will use the PCEDT files to identify noun–noun compounds, because the so-called phrase-structure layer (*p*-layer) in PCEDT includes the noun phrase annotation by Vadas and Curran (2007), which is required to apply our compound identification methods. For bracketing, we will also use the PCEDT's *p*-layer and *t*-layer as well as DeepBank. More concretely, though, we will use the dataset prepared by Oepen et al. (2014) which includes DeepBank and the PCEDT's tectogrammatical layer. We rely on the dataset by Oepen et al. (2014) because it converts the tectogrammatical annotation in PCEDT to dependency representation in which the "set of graph nodes is equivalent to the set of surface tokens". To extract the semantic relations, we will also use the dataset by Oepen et al. (2014) for PCEDT relations and the original NomBank files for NomBank relations.

After each step throughout the whole process, we store the data in a relational database with a schema that represents the different types of information (e.g. bracketing and semantic relations), and the different resources from which they are derived (e.g. PCEDT and NomBank). In addition, the database also stores the WSJ sentences of the noun–noun compounds in our

dataset (i.e. the context). This setup allows us to easily combine information in different ways, and therefore instantiate several versions of the dataset.

4.3 Compound Identification

This section presents careful analysis and experimentation directed at the identification of noun–noun compounds in running English text. We first motivate the need to address this seemingly simple problem. We then move on to explaining how previous work has approached compound identification. Finally, we propose a new method to identify noun–noun compounds based on syntactic information, and compare it with the more traditional PoS-based identification strategy.

4.3.1 Motivation and Past Work

> "Processing compound nouns thus implies not only interpreting compounds to provide explicit meaning representation for them ... but also *recognising their occurrences*, on the very many occasions that they occur in English text" — Spärck Jones (1983, p. 5)[2]

The very first step of creating a noun–noun compound dataset is to identify sequences of nouns in running text and determine whether a given sequence actually forms a compound. Noun–noun compound identification is, thus, an enabling task to create a new compound dataset, and—by extension—a prerequisite to studying noun–noun compound interpretation. However, as we explained in § 2.2, this essential task has been mostly overlooked in the literature, as the majority of past studies focus on semantic interpretation of compounds and do not pay much attention to the question of identifying them. Such a perception can be justified, because compound identification might seem like a simple task at first glance. However, as we will show below, this basic task is in fact not trivial.

Like for many other NLP tasks, the main approaches to compound identification can be broadly categorized as either symbolic or statistical. Those approaches can be further grouped depending on the linguistic information and strategies they use (cf. § 4.3.2). One of the most widely used compound identification heuristics was proposed by Lauer (1995). To identify binary (i.e.

[2]Emphasis added.

two-word) compounds, Lauer's heuristic simply looks for consecutive pairs of so-called 'sure' nouns—nouns that are unambiguous with respect to their PoS tags—that are neither preceded nor followed by other nouns. Hence, words like *bear* and *book* are excluded by Lauer's heuristic because they can be used as verbs as well as nouns. Lauer (1995) reports a high identification precision of 97.9% on a set of 1,068 candidate compounds from the Grolier Multimedia Encyclopedia, where an important factor presumably is his limitation of candidate compound constituents to unambiguous nouns.[3] Importantly, Lauer (1995, p. 130) clearly points out that "there is no guarantee that two consecutive nouns form a compound." For example, the direct and indirect nominal objects of a transitive verb can occur consecutively without forming a noun–noun compound. The following sentence (from the WSJ corpus) illustrates the problem with relying on PoS tags only to identify compounds, wherein the first and second objects of the verb *call* might be mistakenly taken to form a compound, viz. *program part*.

(4.1) Chairman Theodore Cooper called the *program part* of the company's two-year strategy to implement budget constraints.

Variations of the heuristic proposed by Lauer (1995) comprise some of the most widely used symbolic approaches to noun–noun compound identification in more recent NLP studies (Lapata, 2002; Girju et al., 2005; Ó Séaghdha & Copestake, 2007; Tratz & Hovy, 2010). Some of these studies, however, do not mention Lauer's restriction to unambiguous nouns, e.g. Tratz and Hovy (2010), which could lead to identifying more false positives because of PoS tagging errors. Furthermore, some of the studies that use Lauer's heuristic resort to manual inspection of the extracted candidate compounds to exclude false positives (Girju et al., 2007; Ó Séaghdha & Copestake, 2007; Kim & Baldwin, 2008). Others use a frequency-based remedy together with the PoS-based identification method; for example, Farahmand et al. (2015) rely on a PoS-based method to extract noun–noun pairs from Wikipedia and then filter out the pairs whose frequency count is smaller than ten.

[3]Note that the sure noun constraint introduces bias in identifying compounds because it eliminates an entire class of nouns that can be verbed. In addition, if we assume a set 'sure' nouns, there is no guarantee that these nouns will not be verbed. In fact, Lauer (1995, p. 207) lists *information sources* as one of the compounds in his study which means that both constituents are assumed to be 'sure' nouns, but the word *source* can be used as a verb according to the Oxford English Dictionary.

Lapata and Lascarides (2003) evaluate Lauer's heuristic on the BNC by inspecting a sample of 800 noun sequences classified as valid compounds and report an accuracy of 71%, which is substantially lower than the original result by Lauer (1995). However, this lower accuracy score is not unexpected because Lapata and Lascarides (2003) do not implement Lauer's criterion of unambiguous nouns and they use an automatically PoS tagged version of the BNC, which contains some PoS tagging errors. In the same article, Lapata and Lascarides (2003), also introduce statistical models (based on C4.5 decision tree and naïve Bayes learners) to identify noun–noun compounds. They train and test the models on $1,000$ noun sequences that occur only once in the BNC, and experiment with different combinations of features and learners. Their best model achieves an accuracy of 72.3%. In addition to surface form statistics, Lapata and Lascarides (2003) use PoS tag information in their statistical models, which is the same type of linguistic information used in Lauer's heuristic (i.e. PoS tags).

In the following, we propose a new noun–noun compound identification method that makes use of syntactic information to overcome some of the limitations of Lauer's heuristic.

4.3.2 Compound Identification Strategies

In order to explain the problem and our approach more precisely, we define three dimensions of noun–noun compound identification strategies. The first dimension is the type of linguistic information used to detect compounds, namely PoS tags (PoS-based) or syntax trees (syntax-based). The second dimension concerns the treatment of proper nouns (NNPs), and here we can define three options: (a) Simply treat proper nouns like common nouns (i.e. no special treatment), (b) exclude all noun sequences that contain proper nouns or (c) exclude noun sequences that are headed by a proper noun (assuming that the head is always the right-most word in the sequence). We refer to those three strategies, respectively, as NNP^*, NNP^0 and NNP^h. The third dimension concerns the number of constituents (i.e. nouns) within the compound, which partly depends on the type of linguistic information we use to identify noun–noun compounds. In the PoS-based approach, we distinguish between binary and n-ary strategies for compound identification, where the former identifies two-word compounds and the latter identifies compounds that have $n >= 2$ constituents. In the syntax-based approach, we

also distinguish between binary and n-ary compounds. However, taking into consideration that the internal structure (i.e. bracketing) of n-ary compounds is available, we can also decompose n-ary compounds (where $n > 2$) into 'sub-compounds' including binary ones.

We apply the strategies outlined above on the following example (from the WSJ) to illustrate the difference between them:

(4.2) $\text{Nasdaq}_{\texttt{NNP}}$ $\text{bank}_{\texttt{NN}}$ $\text{index}_{\texttt{NN}}$, $\text{which}_{\texttt{WDT}}$ $\text{tracks}_{\texttt{VBZ}}$ $\text{thrift}_{\texttt{NN}}$ $\text{issues}_{\texttt{NNS}}$

First, under a PoS-based binary strategy we will extract *thrift issues*, while an n-ary strategy will extract both *thrift issues* and *Nasdaq bank index*. As for the proper noun treatment, an NNP^0 strategy would exclude *Nasdaq bank index* but NNP^h would not because the proper noun *Nasdaq* is not in the head position. In the syntax-based approach, the same rule for NNP treatment would apply, but we will extract one more binary compound, namely *bank index*, as syntax gives access to the internal structure of the compound *Nasdaq bank index*.

In the following, we will empirically compare the PoS-based and syntax-based approaches for both binary and n-ary compounds (using NNP^0 and NNP^h for proper noun treatment). Since our ultimate goal is to create an annotated dataset of compounds, we will not consider compounds that consist of proper nouns only, because NomBank does not annotate complex proper nouns (cf. § 3.4.1).

4.3.3 Syntax-Based Identification

The PoS-based approach for compound identification simply looks for a consecutive sequence of nouns, and if the goal is to identify binary compounds, then the noun pair cannot be preceded nor followed by other nouns. With richer linguistic representations, viz. syntactic trees, the definition of noun–noun compounds goes one step further; the sequence of nouns is also a sequence of leaf nodes in the parse tree. Therefore, the definition of a noun–noun compound becomes a sequence of noun leaf nodes that are dominated by the same (noun phrase) parent node.[4] The requirement of a single parent node stems from the fact that noun–noun compounds act as one nominal, and hence their constituents cannot belong to two different phrases. In the

[4]We will amend this definition when we introduce the actual syntactic representation used in our experiments.

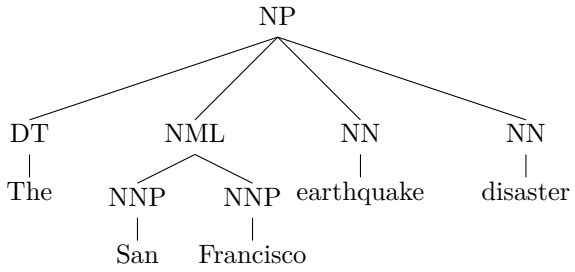


Figure 4.1: Internal noun phrase structure of *The San Francisco earthquake disaster.*

syntax-based approach, unlike the PoS-based approach, it does not matter if a compound is followed or preceded by other nouns, because the surrounding nouns can be excluded if they are dominated by a different parent node.

In order to compare the PoS- and syntax-based approaches, we use the English part of the PCEDT which contains the WSJ corpus of the PTB. As mentioned in § 4.2, we choose to use the PCEDT for compound identification because its *p*-layer includes the internal noun phrase annotations introduced by Vadas and Curran (2007). Recall, further, that the original PTB does not annotate the internal structure of noun phrases, which is why we need the annotations by Vadas and Curran (2007).

Figure 4.1 shows an example of the internal annotation of noun phrases in the PCEDT's *p*-layer. The NML node stands for nominal modifier left-branching, and it is one of the nodes added by Vadas and Curran (2007) to the PTB annotation.[5] Vadas and Curran (2007) left the right-branching noun phrases unaltered; in other words, if a noun phrase does not contain an NML node, then it is right-branching.

Our definition of noun–noun compounds above requires leaf nodes to have a joint parent node, but in Figure 4.1 we see that *San Francisco* has a different parent node than *earthquake disaster*, even though they are part of the same compound (and noun phrase). Therefore, in the implementation of syntax-based compound identification we make an exception for the identical-parent condition when the parent node is of type NML. In concrete terms, this means that we can extract the following three compounds from the structure in Figure 4.1 (assuming we allow compounds consisting of proper nouns only):

[5]The other node Vadas and Curran (2007) add is JJP, and it is used with adjectival phrases.

[[San Francisco] [earthquake disaster]]

Note that even though we make an exception for the identical-parent condition for NMLs, we still preserve their (left) bracketing constraints, and hence, a 'compound' like *Francisco earthquake* will not be extracted from the example phrase above.

4.3.4 Results and Discussion

In order to compare the PoS- and syntax-based approaches we experiment with detecting noun–noun compounds in the full WSJ of the PTB with eight different configurations as shown in Table 4.1. The table provides total counts of compound instances (tokens) and the numbers of distinct strings (i.e. compound types).[6]

In all configurations, the syntax-based approach identifies more compounds than the PoS-based one, and that is because the former has access to the internal structure of the compounds and can therefore extract sub-compounds out of n-ary ones where $n > 2$. Furthermore, in the binary setup, the PoS-based approach is limited to strictly two consecutive nouns. For example, the noun sequence $board_{NN}$ $meeting_{NN}$ $yesterday_{NN}$ is not considered by the binary PoS-based approach because it contains three consecutive nouns, whereas the syntax-based approach extracts the binary sub-compound *board meeting*. Apart from this, the mere numbers do not tell us much in the absence of gold-standard data—to the best of our knowledge there is no gold-standard dataset for noun–noun compound identification. Therefore, we manually inspect a total of 100 random binary NNP^h compounds; 50 of which are only identified by the PoS-based approach and the other 50 are only identified by the syntax-based approach. In other words, we inspect two disjoint sets of compounds whose members are identified by only one of the two approaches.

Of the first set, 28 instances include a percent sign which is tagged as a noun (NN) in PTB, e.g. *% drop* in "...and a 4% drop in car loadings". In fact, sequences like *% stake* and *% increase* are among the top ten most frequent 'compounds' identified by the PoS-based approach, which is unsurprising given the WSJ domain. Such cases are excluded in the syntax-based approach because the percent sign and the following noun belong to two different phrase constituents. The PoS-based set includes five cases that were wrongly

[6]Note that no linguistic pre-processing (e.g. down-casing or stemming) was applied when calculating the type counts reported in Table 4.1.

	PoS-based				Syntax-based			
	Binary		N-Ary		Binary		N-Ary	
	NNP^0	NNP^h	NNP^0	NNP^h	NNP^0	NNP^h	NNP^0	NNP^h
Tokens	27,677	33,167	30,296	39,429	29,535	36,441	34,151	42,835
Types	15,128	18,766	17,167	23,704	15,853	20,018	19,469	25,021

Table 4.1: Total number of noun–noun compounds identified in the WSJ corpus in the PTB. NNP^0: Excludes all sequences that contain proper nouns. NNP^0: Excludes sequences headed by a proper noun.

identified as compounds because of annotation errors on the PoS tag level in PTB, but not on the syntax level. For example, the PoS-based method extracts *store data* from the sentence in Example 4.3, because the word *store* is tagged as NN (singular common noun) in PTB. On the syntactic level, however, the phrase "to store data" receives the right analysis (i.e. verb phrase) and hence it is excluded by the syntax-based method.

(4.3) Conner dominates the market for hard-disk drives used to *store data* in laptop computers

We also find subtler annotation errors like treating the adjective *in vitro* as the combination of a preposition (IN) and a noun (NN), which led the PoS-based approach to extract *vitro cycles* as a compound in "...after only two in vitro cycles". The remaining instances involve nouns that are not dominated by the same parent node (and therefore not identified by the syntax-based method). There are several linguistic constructions that may lead to such errors, such as the objects of a ditransitive verb and temporal modifiers like *today* and *yesterday* (tagged as nouns rather than adverbs in the PTB).[7] In sum, the 50 compounds detected by only the PoS-based approach are invalid noun–noun compounds, which suggests that the syntax-based approach succeeds in excluding some of the false positives referred to by Lauer (1995).

Of the 50 compounds detected by the syntax-based approach only, there are 38 compounds that were extracted from other compounds with more than two constituents—cases which could not have been identified by the binary

[7]According to the Part-of-Speech Tagging Guidelines of the PTB (Santorini, 1990, p. 19): "The temporal expressions yesterday, today and tomorrow should be tagged as nouns (NN) rather than as adverbs (RB). Note that you can (marginally) pluralize them and that they allow a possessive form, both of which true adverbs do not."

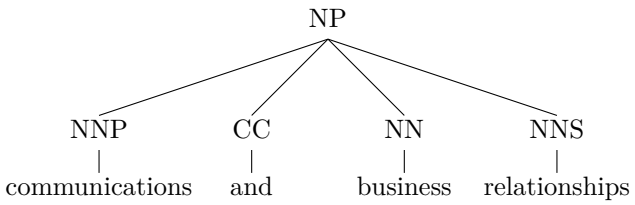


Figure 4.2: Coordination structure

PoS-based approach. Furthermore, we see seven compounds that are either followed or preceded by other nouns. Such cases are also unidentifiable by the PoS-based approach because it requires pairs of nouns not surrounded by other nouns. We also find four annotation errors where left-branching noun phrases were annotated as right-branching. For example, the phrase *San Diego home* is considered right-branching in the Vadas and Curran (2007) annotation of noun phrases in PTB, which leads to extracting *Diego home* as a 'compound'.

The result analysis further reveals that the syntax-based approach includes arguably incorrect compounds when a noun is preceded by a coordinated phrase with noun conjuncts, such as *communications and business relationships* in Figure 4.2. The syntax-based approach extracts *business relationships*, but this can be either incorrect or incomplete extraction given the nature of coordination structures as we will discuss in the following sub-section.

The result analysis also revealed that our implementation of the identical-parent condition was not fine-grained enough to preserve the left bracketing information in some NML constituents. For example, in Figure 4.3 our implementation wrongly extracted the compound *development expenses*. In the following section we report the number of compounds extracted with a finer-grained implementation of the syntax-based approach that handles such errors.

4.3.5 Refined Identification Method

As discussed in the previous section, extracting noun–noun compounds that are partially contained in nominal coordinate structures calls for careful treatment. In order to handle coordinate constructions properly, we need to distinguish between distributive and non-distributive (or collective) coordinate structures. Consider the following coordinate constructions:

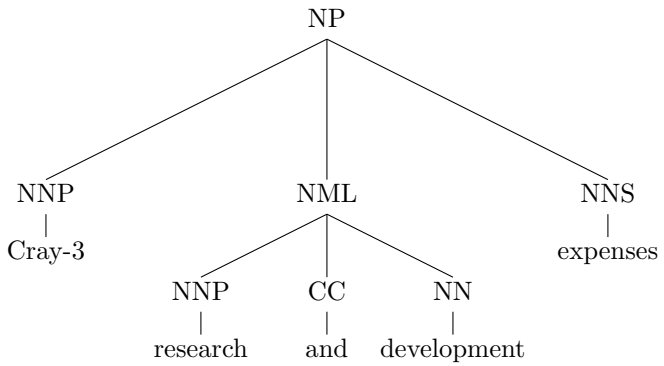


Figure 4.3: Coordination structure: Left-branching

(4.4) Business and nursing programs

(4.5) Research and development expenses

The first construction can be considered distributive and could be paraphrased as *business programs* and *nursing programs*. The second construction, however, is arguably non-distributive, which means that the two nominal conjuncts *research* and *development* 'jointly' modify the noun *expenses*—though it is also possible that the construction is referring to *research expenses* and *development expenses*, but we assume it is non-distributive for the sake of argument. Given this distinction between distributive and non-distributive coordinate structures, it would in principle be possible to extract compounds from distributive coordinate structures; e.g. *nursing programs* from Example 4.4. In practice, however, the PTB annotation does not distinguish between distributive and non-distributive coordinate structures, and therefore we decide conservatively to exclude all noun–noun compounds that are part of coordinate structures.

We further refine our implementation of the syntax-based identification approach to ensure that left-branching noun phrases are handled correctly. Consider the phrase *regional wastewater system improvement revenue bonds* in Figure 4.4, which includes an adjectival modifier as part of the initial compound. According to our definition of noun–noun compounds (as strictly nominal sequences), the only compound that can be extracted from this phrase is *revenue bonds*. Given the underspecified bracketing information within the first NML constituent, extracting *wastewater system* might be incorrect because, arguably, *wastewater* in this construction may be modified by *regional*, as shown in the following bracketing. We refine our implementation of the

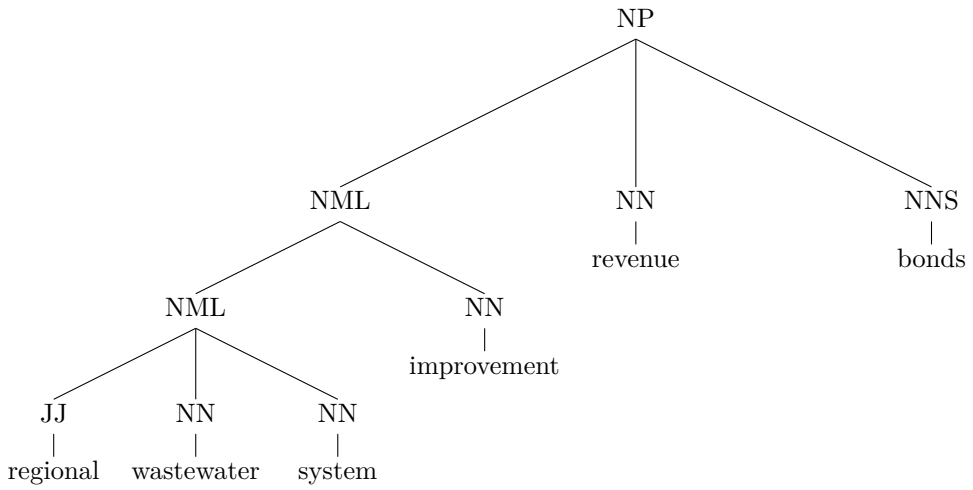


Figure 4.4: Analysis of *regional wastewater system improvement revenue bonds*

syntax-based approach to correctly handle such cases as well, i.e. identify *revenue bonds* as only compound in this example.

[[[[regional wastewater] system] improvement] [revenue bonds]]

Table 4.2 shows the number of compound instances and types identified using the refined syntax-based approach, which also excludes all noun–noun compounds that are part of a coordinate structure. Our refined implementation of the syntax-based heuristic identifies $33,092$ binary NNP^h compounds and $38,917$ n-ary NNP^h compounds. Expectedly, we now identify fewer compounds than in the less restrictive syntax-based method (cf. Table 4.1), but still comparable in number to the PoS-based method (which would extract some compounds from both the conjoined modifier and adjectival modification structures of Figures 4.3 and 4.4). However, the trends regarding false positives and false negatives observed in the results analysis of § 4.3.4 apply with equal force to this more conservative parameterization of our syntax-based method.

We adopt this final set of noun–noun compounds as the basis for the following sections where we automatically construct a dataset of bracketed noun–noun compounds annotated with semantic relations from PCEDT and NomBank. Note, however, that the number of compounds in our dataset will vary in the following sections depending on the resources we use.

	Binary		N-Ary	
	NNP^0	NNP^h	NNP^0	NNP^h
Compound instances	27,418	33,092	29,666	38,917
Compound types	14,498	18,200	16,249	22,765

Table 4.2: Number of noun–noun compounds in the WSJ corpus of the PTB identified using the refined syntax-based method.

4.4 Noun–Noun Compound Bracketing

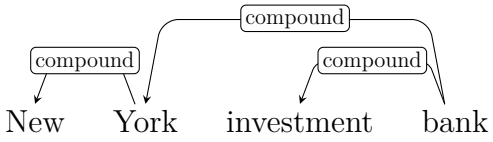
We now turn to the bracketing of multi-word compounds in our dataset, which already came up in the context of compound identification in § 4.3.5. Given that the set of compounds we identified in the previous section contains n-ary compounds, we need to establish the bracketing of such compounds. As explained in § 2.2.2, noun–noun compound bracketing can be defined as the disambiguation of the internal structure of compounds with three nouns or more. For example, we can bracket the compound *noon fashion show* in one of the following ways:

(4.6) Left-branching: [[noon fashion] show]

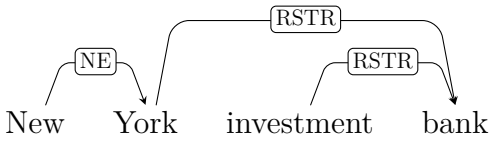
(4.7) Right-branching: [noon [fashion show]]

In this example, the right-branching interpretation refers to a fashion show happening at noon, whereas the left-branching one expresses a show of 'noon fashion', for example in an interpretation of mid-day attire. One can argue that the right-branching bracketing is the more likely one, but the correct bracketing need not always be as obvious—some compounds can be subtler to bracket, e.g. *car radio equipment* (Girju et al., 2005).

As explained in § 4.2, we extract the bracketing of the n-ary compounds in our dataset from three resources: Vadas and Curran's (2007) annotation of noun phrases in PTB (VC-PTB, henceforth), DeepBank and PCEDT. Vadas and Curran (2007) manually annotated the internal structure of noun phrases (NPs) in the PTB which were originally left unannotated (i.e. all NPs in the PTB have a flat structure). However, as is the case with other resources, Vadas and Curran's annotation is not completely error-free. For example, we find at least 57 right-branching compounds that contain *New York*, like the compound *New York lawyer* which is bracketed as right-branching by Vadas and Curran (2007), but it is obviously left-branching, i.e. [[New York]



(a) DeepBank



(b) PCEDT

Figure 4.5: The DeepBank and PCEDT annotation of the compound *New York investment bank*.

lawyer]. We, therefore, crosscheck the bracketings by Vadas and Curran (2007) with those of DeepBank and PCEDT. The latter two, however, do not contain explicit annotation of noun–noun compound bracketing, but we can 'reconstruct' the bracketing based on the dependency relations assigned in both resources, i.e. the logical form meaning representation in DeepBank and the tectogrammatical layer (*t*-layer) in PCEDT. More specifically, we rely on the dependency edges in the graph representations by Oepen et al. (2014) to reconstruct the compound bracketing in DeepBank and PCEDT. For example, Figures 4.5a and 4.5b show (parts of) the bi-lexical dependency graphs of the compound *New York investment bank* in DeepBank and PCEDT. Based on the dependency edges in both graphs, we extract the following bracketing for the compound: [[New York] [investment bank]].

4.4.1 Data and Results

In the following, we present the results of mapping and contrasting the PCEDT, DeepBank and VC-PTB bracketing of the n-ary compounds (where $n > 2$) identified using the refined syntax-based approach in § 4.3.5.

Even though we can identify 38,917 noun–noun compounds in the full WSJ corpus (cf. Table 4.2), the set of compounds that constitutes the basis for bracketing analysis (i.e. the set of compounds that occur in all three aforementioned resources) is smaller for two reasons. First, DeepBank only annotates the first 22 sections of the WSJ corpus. Second, not all the

	DeepBank PCEDT	DeepBank VC-PTB	PCEDT VC-PTB	DeepBank PCEDT VC-PTB
NNP^h	80%	79%	88%	75%
NNP^0	78%	75%	90%	74%
NNP^h excl. sub	82%	82%	86%	75%
NNP^0 excl. sub	81%	77%	90%	74%

Table 4.3: Pairwise and three-way bracketing agreement. NNP^0: excluding proper nouns; NNP^h: proper nouns are only allowed in the modifier position; excl. sub: excluding sub-compounds

noun sequences identified as compounds in VC-PTB are treated as such in DeepBank and PCEDT. Hence, the number of noun–noun compounds that occur in all the three resources is 26,500. Furthermore, almost three-quarters (76%) of these compounds consist of two nouns only, meaning that they do not require bracketing, which leaves us a subset of 6,244 compounds—we will refer to this subset as the bracketing subset. Unless otherwise specified, all numbers and percentages reported in the rest of this sub-section are based on the bracketing subset.

After mapping the bracketings from the three resources we find that they agree on the bracketing of almost 75% of the compounds in the bracketing subset (cf. the right-most column in Table 4.3). Such an agreement score is relatively good compared to previously reported agreement levels on much smaller datasets, e.g. Girju et al. (2005) report a bracketing agreement of 87% on a small set of 362 three-word compounds. Inspecting the disagreement among the three resources reveals two things. First, noun–noun compounds which contain proper nouns (NNP) constitute 45% of the compounds that are bracketed differently. Second, 41% of the differently bracketed compounds are actually sub-compounds of larger compounds. For example, the compound *deputy editorial features editor* is bracketed in three different ways in VC-PTB, PCEDT and DeepBank which results in different sub-compounds in these resources, namely *deputy editorial features* in PCEDT and *editorial features editor* in VC-PTB and DeepBank. The sub-compounds themselves also receive different bracketing analyses, leading to extracting the binary sub-compounds *editorial features* from PCEDT and DeepBank and *features editor* from VC-PTB.

VC-PTB	[New [York [exchange [board meeting]]]]
PCEDT	[[New York] [exchange [board meeting]]]
DeepBank	[[[New York] [exchange board]] meeting]
Ours	[[[[New York] exchange] board] meeting]

Table 4.4: Bracketing of *New York exchange board meeting* in VC-PTB, PCEDT and DeepBank. The last row shows our intuition of what should be the correct bracketing.

It is noteworthy that those two observations do not reflect the properties of compounds containing NNPs or sub-compounds; they only tell us their percentages in the set of differently bracketed compounds. To get the full picture, we need to look at the number of sub-compounds and compounds containing NNPs in the set of compounds where the three resources agree. As it turns out, 72% of the compounds containing NNPs and 76% of the sub-compounds are bracketed similarly across the three resources. Therefore, when we exclude them from the bracketing subset we do not see a significant change in bracketing agreement among the three resources (compare the top two cells in the right-most column in Table 4.3 with the bottom two cells in the same column).

We find eleven cases only where the three resources provide three different bracketings, and in some cases the three bracketings are arguably wrong. For example, Table 4.4 shows the bracketing of the compound *New York exchange board meeting* according to the three resources, in addition to what we believe to be the correct bracketing.[8]

In Table 4.3 we report pairwise bracketing agreement levels among the three resources. The relatively high agreement level between PCEDT and VC-PTB should not come as a surprise; the so-called phrase-structure layer (*p*-layer) in PCEDT uses the VC-PTB annotation, and therefore the PCEDT bracketing extracted from the *t*-layer is expected to be similar to that in the *p*-layer (i.e. VC-PTB bracketing). Further, PCEDT and VC-PTB seem to disagree more on the bracketing of noun–noun compounds containing NNPs, because when proper nouns are excluded the agreement level between PCEDT

[8]Our bracketing is based on the context in which the compound occurs in the PTB: "At another point during the hearing, Rep. Markey asked Mr. Phelan what would be discussed at a *New York exchange board meeting* today." Knowing that John J. Phelan Jr was a chairperson of the New York Stock Exchange, we believe that our analysis is more plausible than the other three bracketings in Table 4.4.

and VC-PTB increases, but it decreases for the other two pairs.

As we look closer at the compound instances where at least two of the three resources disagree, we find that some instances are easy to classify as annotation errors. For example, the compound *New York streets* is bracketed as right-branching in VC-PTB, but we can confidently say that this a left-bracketing compound. Not all bracketing disagreements are that easy to resolve though; one example where left- and right-bracketing can be accepted is *European Common Market approach*, which is bracketed as follows in DeepBank (1) and PCEDT and VC-PTB (2):

1. [[European [Common Market]] approach]
2. [European [[Common Market] approach]]

Even though we do not aim to study compound bracketing in this work, we believe that a part of the bracketing dataset can be used for future experimentation. More specifically, we can redefine the bracketing subset to be the set of compounds that are bracketed similarly in the three resources, which arguably ensures a high-quality annotation.[9] Lastly, using the bracketing annotation, we can extract binary sub-compounds from n-ary ones and include them in our compound interpretation dataset, as detailed in the following section.

4.5 Semantic Relations

Now that we have a set of noun–noun compounds with bracketing annotation for n-ary compounds, we move to the last step of adding semantic relations to the compounds in our dataset.

As explained in § 4.2, we rely on PCEDT and NomBank to define the semantic relations in our dataset, which includes the bracketed compounds from the previous section as well as two-word or binary compounds. However, unlike in § 4.4, the set of noun–noun compounds in this section consists of the compounds that are (1) bracketed similarly in PCEDT and VC-PTB, (2) occur

[9]Resolving the bracketing discrepancies in the three resources falls outside the scope of our work, but we have shared our findings with the original resource creators and provided them with a web-based 'tool' that allows them to inspect the bracketing disagreement, and possibly correct them. The tool can be found on `https://ltgoslo.github.io/fun-nom/bracketing`

in both resources and (3) are annotated in NomBank. This set consists of 26,709 compounds and 15,647 compound types, which can be either binary or n-ary and include proper nouns in the modifier position only.[10] We do not use the intersection of the three resources (as in § 4.4), because DeepBank does not contribute to the semantic relations of noun–noun compounds (cf. § 3.4.3) and it limits the size of our dataset (cf. § 4.4). Nonetheless, given our technical setup (i.e. the database) we can readily produce the set of compounds that occur in the three resources and are bracketed similarly, and then extract their semantic relations from PCEDT and NomBank.

We introduced some of the theoretical aspects behind NomBank and PCEDT in § 3.4.1 and § 3.4.2, respectively; for convenience, we briefly repeat some of the relevant points here. PCEDT assigns syntactico-semantic labels, so-called functors, to all the syntactic dependency relations in the tectogrammatical layer (a deep syntactic structure). Drawing on the valency theory of the Functional Generative Description, PCEDT defines 70 functors for verbs as well as nouns and adjectives (Cinková et al., 2009).[11] NomBank, on the other hand, annotates the argument structure of nominal predicates only; in other words, it assigns role labels (arguments and adjuncts) to common nouns in the PTB. In general, NomBank distinguishes between predicate arguments and adjuncts (modifiers), which correspond to those defined in PropBank.[12] We take both types of NomBank relations (i.e. arguments and adjuncts) as well as the PCEDT functors to be the 'semantic' relations of noun–noun compounds in our dataset.

4.5.1 Data, Results and Reflections

Given 26,709 noun–noun compounds, we construct a dataset with two relations per compound: a PCEDT functor and a NomBank argument role or modifier. The resulting dataset is relatively large compared to the datasets we reviewed in the previous chapter (cf. Table 3.1). However, the largest dataset by Tratz and Hovy (2010) is type-based and includes a small number

[10] The number of compound types becomes 14,405 if we count based on the lemmatized form of the constituents.

[11] The functors attested in our dataset are fully defined in § B.2. The full inventory of PCEDT functors is available on `https://ufal.mff.cuni.cz/pcedt2.0/en/functors.html`. Accessed: 13 April 2019.

[12] We define the NomBank arguments and adjuncts attested in our dataset in § B.1. The full list of NomBank adjuncts can be found in Table 2 in Meyers (2007, p. 90).

Relation	Count	%	Example
ARG1	15811	59.20	interest rate
ARG2	3779	14.15	appeals court
ARG0	2701	10.11	bank financing
ARG3	1767	6.62	vice president
ARGM-LOC	1131	4.23	market price
ARGM-MNR	563	2.11	program trading
ARGM-TMP	510	1.91	pretax profit
ARGM-PNC	149	0.56	takeover bid
Support	142	0.53	parent company
ARG4	76	0.28	earthquake insurance
ARGM-ADV	18	0.07	paper losses
ARG1-H0	15	0.06	sticker-shock reaction
ARG0-H0	12	0.04	House-Senate agreement
ARG8	9	0.03	breakfast meeting
ARGM-EXT	8	0.03	percentage gainer
ARGM-CAU	4	0.01	cancer deaths
ARG9	4	0.01	Geneva meeting
ARG1-H1	4	0.01	debt-reduction requirement
ARG3-H0	3	0.01	Chicago-Paris flight
ARG5	3	0.01	landslide win

Table 4.5: Distribution of NomBank relations over the set of 26,709 compounds. The relations are defined in § B.1.

of compounds with proper nouns. The size of our dataset becomes 11,761 if we exclude the compounds containing proper nouns and only count the compound types.[13] In addition to being the second largest dataset for compound interpretation, our dataset offers several important advantages, such as dual semantic annotation and bracketing of multi-word compounds, inter alia.

Tables 4.5 and 4.6 show the NomBank and PCEDT relations attested in our dataset and their distribution over the 26,709 compounds.[14] Overall, we observe a total of 34 PCEDT functors and 20 NomBank roles and adjuncts.[15]

[13]Counting the number of compound types based on their lemmas, the size of our dataset becomes 10,596.

[14]In Table 4.6, and throughout this thesis, we shorten the names of some the PCEDT functors. Specifically, we drop the suffix '-arg' from the names of the following relations: PAT-arg, ACT-arg, ORIG-arg, EFF-arg and ADDR-arg.

[15]We also use a special relation called NONE with 37 binary compounds that do not actually form compounds according to the PCEDT *t*-layer analysis. For example, in our

Relation	Count	%	Example
RSTR	12992	48.64	vice president
PAT	3867	14.48	auto maker
APP	3543	13.27	company spokesman
REG	2176	8.15	money manager
ACT	1286	4.81	earnings growth
LOC	979	3.67	floor trader
TWHEN	367	1.37	pretax gain
AIM	284	1.06	pension fund
ID	256	0.96	Bush administration
MAT	136	0.51	interest rates
NE	132	0.49	Bay area
ORIG	114	0.43	interest income
MANN	83	0.31	program trading
MEANS	56	0.21	loan losses
EFF	55	0.21	stock prices
AUTH	49	0.18	Hemingway book
BEN	40	0.15	employee manuals
ADDR	39	0.15	consumer loans
CAUS	38	0.14	cancer deaths
THL	38	0.14	quarter loss
CRIT	25	0.09	bankruptcy-law protection
DIR1	21	0.08	college dropouts
DIR3	19	0.07	research investment
TFRWH	18	0.07	December contract
EXT	8	0.03	part time
TFHL	6	0.02	lifetime job
CPR	5	0.02	zombie approach
DIFF	5	0.02	media witch hunt†
ACMP	4	0.01	discount-coupon books
DIR2	1	0.00	highway travelers
DPHR	1	0.00	ambulance chasers
RESL	1	0.00	controlling interest
THO	1	0.00	quarterly dividend
TPAR	1	0.00	session losses

Table 4.6: Distribution of PCEDT functors over the set of 26,709 compounds. The relations are defined in § B.2. † The compound *media witch hunt* is right-branching, which means the relation is between *media* and *hunt*.

Before we discuss the distribution of these relations, we define some of the frequent ones in the following. In PCEDT, the most frequent functor, RSTR, is an underspecified relation that expresses a restrictive attribute; PAT and ACT describe, respectively, patient and agent (or actor) arguments; APP describes possession or ownership (appurtenance); REG expresses a circumstance the predicate takes into account; TWHEN and LOC refer to temporal and locative modifiers, respectively. The interpretation of the numbered arguments in NomBank often depends on the predicate itself, but—conventionally—ARG0 and ARG1 correspond to proto-agent (or actor) and proto-patient (or theme) roles, respectively. ARG2 typically expresses recipient or beneficiary, but this need not always hold. Adjuncts in NomBank are denoted by the prefix ARGM- followed by a so-called function label such as LOC (locative), TMP (temporal), MNR (manner), CAU (cause), PNC (purpose) and EXT (extent). We provide the definition of all the NomBank and PCEDT relations in our dataset in § B.1 and § B.2.

As can be seen from Tables 4.5 and 4.6, the distribution of the NomBank and PCEDT relations is rather imbalanced; only twelve functors and nine NomBank relations occur more than 100 times in the dataset. Further, the most frequent NomBank relation (ARG1) accounts for 59% of the data, and the five most frequent relations account for about 95% of the data. We see a similar pattern in the distribution of PCEDT functors, where 49% of the compounds are annotated with the underspecified functor RSTR, and the five most frequent functors account for 89% of the data (cf. Table 4.6).

Such a distribution of relations is not unexpected in both PCEDT and NomBank. Almost 61% of the 114,576 noun instances annotated in NomBank include an ARG1 argument, and hence the high frequency of this relation in our dataset is just representative of the original NomBank annotation. Moreover, we have already seen a somewhat similar pattern in other datasets; for example, Kim and Baldwin (2008) report that 42% of the compounds in their dataset are annotated as TOPIC (cf. § 3.2.2), which is more or less the equivalent of ARG1 in NomBank. According to Cinková et al. (2006), the arguments or relations that cannot be expressed by "semantically expressive" functors in PCEDT usually get assigned the functor PAT, which is the second

dataset, *oil man* is extracted as a binary compound from the phrase "... decide how much oil man William Herbert Hunt will owe ...", but the PCEDT's *t*-layer analyzes *oil* as part of the WH-phrase, i.e. [how much oil] [man William Herbert Hunt will owe], whereas the PTB analysis of this phrase looks as follow: [how much][oil man][William Herbert Hunt will owe].

most frequent functor in our dataset.

Of course, such a skewed distribution of relations is far from ideal for training machine learning models. We can potentially modify our dataset to achieve a slightly more balanced distribution, for example, by mapping some of the specific temporal functors in PCEDT (`THL`, `THFL` and `THO`; cf. § B.2) to just one general relation (e.g. `TWHEN`). However, given our goal to study how noun–noun compounds are interpreted in meaning representation frameworks, we choose to use the dataset as-is, and instead try to employ various machine learning strategies for overcoming the challenge of an imbalanced data distribution (cf. Chapter 7).

Lastly, some of the examples we provide in Tables 4.5 and 4.6 reveal that our dataset still contains some errors. For example, *quarterly dividend* in Table 4.6 is not a noun–noun compound, but it was extracted from the WSJ text because the PTB tags *quarterly* as a noun in this instance.[16] In addition, some of the constituents in Tables 4.5 and 4.6 are hyphenated words, but they still form a single unit in our dataset; for example, *discount-coupon books* is considered a binary compound in our dataset. NomBank tends to be more specific when annotating hyphenated words through using so-called hyphen tags (`-H0` and `-H1`) to denote which constituent of the hyphenated string is actually the argument; e.g. the relation `ARG1-H1` indicates that *reduction* in *debt-reduction requirement* is the `ARG1` of *requirement*. We will specify how such compounds will be treated in practice in Chapter 6.

4.5.2 Correspondence between PCEDT and NomBank

One of the motivations behind creating our new dataset is to compare how different meaning representation frameworks annotate the same set of compounds. In this section, we present a quantitative analysis of how the PCEDT functors map to NomBank arguments and modifiers, and vice versa.

In theory, some of the PCEDT functors and NomBank relations express the same type of relations; for example, both the PCEDT functor `LOC` and NomBank adjunct `ARGM-LOC` describe a locative modifier.[17] Therefore, in order to confirm whether such 'theoretical' similarities actually hold in practice,

[16] The 'compound' is extracted from the following sentence: "CMS Energy Corp. said it would begin paying a 10-cent-a-share quarterly dividend, the company's first since 1984".

[17] Bonial et al. (2014) also find lexical resource like PropBank and EngValLex (the PCEDT's English valency lexicon) "surprisingly compatible" even though they were created "independently and with different goals".

	ARG1	ARG2	ARG0	ARG3	ARGM-LOC	ARGM-MNR	ARGM-TMP	ARGM-PNC
RSTR	0.60	0.12	0.08	0.10	0.03	0.03	0.01	0.01
PAT	**0.89**	0.05	0.01	0.03	0.01	0.01		0.01
APP	0.42	0.37	0.17	0.01	0.03	0.00	0.00	0.00
REG	0.75	0.09	0.07	0.07	0.00	0.01	0.00	0.00
ACT	0.46	0.03	**0.48**	0.01	0.01	0.00		
LOC	0.16	0.20	0.09	0.01	**0.54**			
TWHEN	0.12	0.04	0.00		0.01		**0.81**	
AIM	0.65	0.12	0.06	0.08	0.00	0.00		**0.05**
ID	0.39	0.30	0.27	0.04	0.00			
MAT	0.86	0.09	0.01	0.02				
NE	0.32	0.46	0.13	0.02	0.06			
ORIG	0.20	0.19	0.13	0.37	0.06	0.01		0.01
MANN	0.23	0.07	0.01		0.04	**0.65**		
MEANS	0.45	0.09	0.04	0.12	0.14	0.11		
EFF	0.60	0.18	0.11	0.04				0.04
AUTH			**1.00**					
BEN	0.45	0.35	0.03	0.17				
ADDR	0.18	0.64	0.10	0.08				
CAUS	0.21	0.18	0.18	0.32				0.08
THL		0.03		0.03			**0.95**	

Table 4.7: Correlation between PCEDT functors and NomBank arguments and modifiers. The table maps PCEDT to NomBank relations. The numbers in boldface indicate the PCEDT functors and NomBank relations that we a priori assume are semantically comparable. The empty cells denote zero, whereas 0.00 is a very small non-zero number. The correlations are to be read row-wise.

	ARG1	ARG2	ARG0	ARG3	ARGM-LOC	ARGM-MNR	ARGM-TMP	ARGM-PNC
RSTR	0.50	0.40	0.38	0.76	0.37	0.79	0.27	0.66
PAT	**0.22**	0.05	0.02	0.06	0.02	0.07		0.13
APP	0.09	0.34	0.22	0.02	0.09	0.00	0.01	0.01
REG	0.10	0.05	0.05	0.08	0.01	0.02	0.01	0.07
ACT	0.04	0.01	**0.23**	0.00	0.02	0.01		
LOC	0.01	0.05	0.03	0.00	**0.47**			
TWHEN	0.00	0.00	0.00		0.00		**0.58**	
AIM	0.01	0.01	0.01	0.01	0.00	0.00		**0.09**
ID	0.01	0.02	0.03	0.01	0.00			
MAT	0.01	0.00	0.00	0.00				
NE	0.00	0.02	0.01	0.00	0.01			
ORIG	0.00	0.01	0.01	0.02	0.01	0.00		0.01
MANN	0.00	0.00	0.00		0.00	**0.10**		
MEANS	0.00	0.00	0.00	0.00	0.01	0.01		
EFF	0.00	0.00	0.00	0.00				0.01
AUTH			**0.02**					
BEN	0.00	0.00	0.00	0.00				
ADDR	0.00	0.01	0.00	0.00				
CAUS	0.00	0.00	0.00	0.01				0.02
THL		0.00		0.00			**0.07**	

Table 4.8: Correlation between NomBank arguments and modifiers and PCEDT functors. The table maps NomBank to PCEDT relations. The numbers in boldface indicate the PCEDT functors and NomBank relations that we a priori assume are semantically comparable. The empty cells denote zero, whereas 0.00 is a very small non-zero number. The correlations are to be read column-wise.

we show the correlation between the PCEDT and NomBank relations in Tables 4.7 and 4.8. The first table shows the mapping of PCEDT functors to NomBank relations (and Table 4.8 is the other way around—from NomBank to PCEDT); for instance, the second cell from the top under the second column in Table 4.7, shows that 89% of the compounds annotated as PAT in PCEDT receive the relation ARG1 in NomBank. Note that since some of the relations in both resources annotate only a handful of compounds, we limit the comparison to the 20 most frequent functors in PCEDT and the 8 most frequent relations in NomBank.

The numbers in boldface in Tables 4.7 and 4.8 indicate the PCEDT functors and NomBank relations that we a priori assume are semantically comparable. For example, the temporal functors in PCEDT (viz. TWHEN and THL) intuitively correspond to the temporal modifier in NomBank (ARGM-TMP). Indeed, looking at the figures in Tables 4.7 and 4.8, it is evident that such similarities largely hold in practice. More specifically, 81% of the TWHEN compounds and 95% of the THL compounds in PCEDT are annotated as ARGM-TMP in NomBank (note, however, that there are only 38 THL compounds; cf. Table 4.6). From Table 4.8, 58% of the ARGM-TMP compounds in NomBank map to THWEN in PCEDT. On the surface, 58% might seem comparatively low (considering the reverse mapping), however the number of ARGM-TMP compounds (510) is higher than the TWHEN compounds (367), and hence in the best case scenario the correspondence would be at 72%. In other words, the total number of TWHEN compounds constitutes about 72% of the ARGM-TMP ones.

The locative modifiers in NomBank and PCEDT, which were mentioned above, are also expected to annotate many of the same compounds. The number of compounds annotated by the locative relation in PCEDT is comparable to those annotated as such in NomBank (979 and 1,131, respectively; cf. Tables 4.5 and 4.6). However, the overlap between the sets of compounds annotated by LOC and ARGM-LOC is 54% from PCEDT to NomBank and 47% the other way around. From Table 4.8, we see that 37% of the ARGM-LOC compounds (that is, 415 compounds) are annotated as RSTR in PCEDT. Looking at some examples in this set of compounds, we find some potential inconsistencies in the PCEDT annotation; for example, the compound *floor trader* occurs 21 times in our dataset and is annotated with three different functors in PCEDT (17 RSTR, 3 LOC and 1 ACT). In NomBank, however, all the instances of this compound are annotated with ARGM-LOC. We further probe

the issue of potential annotation inconsistency in PCEDT, in the following subsection.

Similarly, the functors ACT (actor) and AUTH (authorship) show relatively high correspondence with the NomBank argument ARG0 (agent or actor), but the reverse correspondence is not as obvious. The same applies to the mapping from the functor PAT (patient) to the NomBank argument ARG1. In both cases, however, the (high) number of ARG1 and ARG0 in NomBank affects the percentages in Table 4.8 when mapping these two relations to their PCEDT counterparts. Perhaps the most obvious example of where our assumed theoretical similarity does not hold is the relations AIM in PCEDT and ARGM-PNC; both relations express purpose, but their correspondence is very low, cf. Tables 4.7 and 4.8.

It is evident from the discussion above that not all 'theoretical similarities' are necessarily reflected in practice. NomBank and PCEDT are two different resources that were created with different annotation guidelines and by different annotators, and therefore we cannot expect perfect correspondence between PCEDT functors and NomBank arguments and adjuncts. That said, in Chapter 7, we will use transfer and multi-task learning as an empirical step to determine the usefulness of the (partial) correspondence between the NomBank and PCEDT relations.

4.5.3 Type vs. Token Semantics

Thus far, our analysis has focused on the instance-based (or token-based) version of our dataset, where the same compound type can occur more than once. However, as we have repeatedly highlighted, most of the existing compound datasets operate on the type level. It is, therefore, often assumed that noun–noun compounds have the same interpretation regardless of their context.

In the previous section, we observed that PCEDT assigns more than one functor to different instances of the compound *floor trader*. In fact, some of the most frequent compound types are annotated with several functors in PCEDT; for example, the compound *Bush administration* is the most frequent compound under the functor APP (69 times), but it is also the second most frequent compound annotated with the functor ID (10 times). Likewise, *interest rate* is the most frequent compound annotated with the functor MAT (12 times) and the second most frequent one annotated as APP (64 times). The

compound *takeover bid*, which occurs 28 times in our dataset, is annotated with four different functors in PCEDT, including `AIM` and `RSTR`, whereas in NomBank it is always annotated as `ARGM-PNC`. Overall, around 13% of the compound types are annotated with more than one functor in PCEDT, whereas only 1.3% of the compound types are annotated with more than one argument in NomBank.[18]

This observation raises the question of whether or not the semantics of noun–noun compounds varies depending on their context, i.e. token-based vs. type-based semantics. While we identify a few examples that support the token-based perspective in NomBank, most of the examples in PCEDT seem to be a result of inconsistency in annotation. Indeed, the PCEDT documentation clearly cautions that "[t]he annotators tried to interpret complex noun phrases with semantically expressive functors as much as they could. This annotation is, of course, very inconsistent."[19] We suspect that the annotation inconsistency is partly due to the fact that the PCEDT English valency lexicon is primarily focused on verbal lexical units, and thus there are no restrictions on the valency structure for nouns.[20] Therefore, in the following we focus on NomBank only.

We identify at least two reasons why NomBank sometimes assigns different relations for different occurrences (or instances) of the same compound type. First, if one of the compound's constituents has more than one sense, this can lead to different interpretations depending on the context. We already gave an example of such cases in § 2.2.3, in which the compound *earthquake coverage* is annotated with two relations depending on the meaning of *coverage*. In Example 4.8, the word *coverage* is used to refer to insurance protection, whereas in Example 4.9 *coverage* is used in the context of media reporting.

(4.8) Industry officials say the Bay Bridge – unlike some bridges – has no *earthquake coverage*$_{\texttt{ARG4}}$, either, so the cost of repairing it probably

[18] In this subsection, when we write that a compound is annotated with more than one relation, we mean that different instances or occurrences of the same compound type are annotated with distinct relations.

[19] The PCEDT online documentation `https://ufal.mff.cuni.cz/pcedt2.0/en/valency.html`. Accessed: 21 March 2019.

[20] In § 3.4.2, we wrote that the PCEDT defines "specific annotation rules for deverbal nominalizations, based on a set of suffixes typical of deverbal nouns, which assumes that some forms of nominalizations inherit their argument structure from the underlying verb." However, this does not immediately entail that nominalizations have their own entries in the PCEDT English Valency Lexicon.

would have to be paid out of state general operating funds.

(4.9) The Associated Press's *earthquake coverage*$_{\texttt{ARG1}}$ drew attention to a phenomenon that deserves some thought by public officials and other policy makers.

Overall, there are only 60 compound types in our dataset that are annotated with more than one sense from NomBank. However, some of these are clearly due to annotation errors; for example, the compound *budget cut* occurs five times in our dataset, four of which interpret *cut* using the NomBank sense 'cut.02' (meaning reduce) and one time as 'cut.01' (slice). There are also examples of subtle distinctions where it is difficult to tell if the distinction is a deliberate choice or annotation error; e.g. *product development* occurs with two senses of *development*, one refers to the act of creating and the other refers to when things or events come about.

Second, some cases can likely be blamed on annotation inconsistency on the argument level in NomBank. For example, 14 of the 15 instances of *pilot union* in our dataset are annotated as `ARG1` and only one is annotated as `ARG2`, with no obvious reason why this instance is different from the rest. Likewise, instances of the following compounds are annotated with either `ARG1` or `ARG2`: *labor problem*, *health problem*, *housing problem* and *financing problem*. Upon examining these instances, we cannot immediately tell if there is, in fact, a genuine motivation behind the annotation decisions, cf. Examples 4.10 and 4.11. However, we suspect that the definition of `ARG1` as "theme" and `ARG2` as "value" in the NomBank frame of *problem* could have been somewhat ambiguous for the annotators.

(4.10) In Houston, we have seen how bad the *housing problem*$_{\texttt{ARG1}}$ can become.

(4.11) Nearly 36% ranked *housing problems*$_{\texttt{ARG2}}$ as their most serious unmet legal need.

Based on the discussion above, we believe that the variation in the instance-based annotation of compounds in our dataset is largely due to annotation inconsistencies. Therefore, in the following section, we will define a type-based version of our dataset which we will use to train machine learning models for compound interpretation. Nonetheless, the question of token-based vs.

type-based interpretation remains an open one, with a few examples from our dataset that support this view.[21]

4.5.4 Compound Interpretation Dataset

In this section, we introduce the version of the dataset we will use in our machine learning experiments for compound interpretation in the following chapters; we will refer to this version as the interpretation dataset.

To define our compound interpretation dataset, we impose three conditions or criteria on the compounds to be included (the conditions are, by and large, motivated by the findings of this chapter):

1. We require all noun–noun compounds in the interpretation dataset to be binary or two-word compounds.

2. We exclude all compounds that contain proper nouns whether in the head or modifier position.

3. Compounds can occur only once in the dataset, i.e. our interpretation dataset is type-based.

The last condition calls for choosing one relation per compound type in the cases where instances of the same compound type are annotated with multiple relations. We choose the majority relation for each compound type that is annotated with more than one relation in PCEDT and NomBank. For example, the compound *money manager* occurs 61 times in our instance-based dataset and is annotated with three PCEDT functors `PAT` (27 times), `REG` (24 times) and `RSTR` (10 times); we select its most frequent functor (`PAT`) for the compound type. Whenever there is a tie (i.e. there is no majority relation), we favor the non-`RSTR` functors; e.g. if a compound occurs ten times, five of which are annotated with `TWHEN` and the other five are annotated with `RSTR`, we choose the functor `TWHEN` as the relation of the compound type.

[21] For example, we observe in our dataset that the compound *government bill* refers to a government law proposal in the following sentence: "It was the first time in 20 years that such government bills were defeated." However, in another context, the same compound refers to government treasury securities: "Results of the Tuesday, October 10, 1989, auction of short-term U.S. government bills, sold at a discount ... ". The constituent *bill* in these two instances is annotated with two different senses in NomBank, 'bill.03' in the former and 'bill.02' in the latter.

Given the three criteria above, the size of the compound interpretation dataset becomes 10,596, counting the compound types based on the surface form. The dataset size becomes 9,611 if we count the compound types based on their lemmas. Furthermore, the number of NomBank relations goes down to 18 (in contrast to 20 in the instance-based dataset). The two relations that do not occur in the interpretation dataset are `ARG3-H0` and `ARG0-H0`, because they are only observed in the set of compounds that contain proper nouns. The distribution of the relations remains skewed with both the NomBank and PCEDT relations. In fact, many of the relations likely are too infrequent to be learned by machine learning models; while we do not exclude these relations from the dataset, we define a minimum frequency threshold that determines whether a relation is taken into account when computing the macro-averaged F_1 scores of our models in Chapters 6 and 7. We refer to this subset of relations as the 'learnable' relations, and it consists of the NomBank and PCEDT relations that annotate at least 50 compounds in our dataset. In total, there are nine 'learnable' PCEDT functors and eight 'learnable' NomBank relations.[22]

Finally, we randomly split the dataset into three parts: training (70% of the dataset), development (10%) and test (20%).[23] We make sure that the distribution of relations reflect the proportions of the three splits, but some of the extremely infrequent relations can only occur in the training split, such as `AUTH` in PCEDT and `ARG5` in NomBank (both relations occur only two times in our dataset). We will use the training and development splits to train and fine-tune our classification models throughout Chapters 5, 6 and 7. Towards the end of this thesis, in Chapter 7, we will also use the test split to evaluate our final models.

4.6 Conclusion

One of the main contributions of this work is the creation of a new dataset of noun–noun compounds couched in existing meaning representation frameworks. In this chapter, we presented the dataset and detailed the process of deriving it. Our new dataset is the largest dataset that includes *both*

[22]The 'learnable' relations in PCEDT: `RSTR`, `PAT`, `REG`, `APP`, `ACT`, `AIM`, `TWHEN`, `LOC` and `MAT`; and in NomBank: `ARG0`, `ARG1`, `ARG2`, `ARG3`, `ARGM-LOC`, `ARGM-TMP`, `ARGM-MNR` and `ARGM-PNC`.

[23]The interpretation dataset is publicly available at `https://github.com/ltgoslo/fun-nom`.

compound bracketing and semantic relations, and the second largest dataset in terms of the number of compound types and excluding compounds that contain proper nouns.

A significant part of this chapter (§ 4.3) focused on the task of noun–noun compound identification, which many of the recent studies have not dealt with in much detail. We reviewed the most commonly used compound identification heuristic by Lauer (1995) and discussed its limitations in § 4.3.1. We laid out three dimensions on which compound identification strategies vary, namely (1) type of linguistic information used to identify compounds (PoS tags vs. syntax trees), (2) treatment of proper nouns (NNP^0, NNP^h and NNP^*) and (3) length of compounds (binary vs. n-ary). In § 4.3.3, we defined a syntax-based compound identification method that makes use of the syntactic annotation in the PTB as well as the noun phrase annotation by Vadas and Curran (2007). We then evaluated contrastively variations of the syntax-based and PoS-based approaches to identify compounds in the WSJ corpus of the PTB (§ 4.3.4). One of the challenges for quantifying the accuracy of the different identification strategies is the lack of gold-standard evaluation data. We therefore opted for manual inspection of a subset of the identified compounds, which in turn led to gradual improvement in our implementation of the syntax-based method. Overall, our results and analysis showed that achieving high-quality compound identification requires linguistic representations at least at the level of syntactic structure. We also showed, however, that complex cases that include coordinate structures may call for even richer linguistic annotations. Given that our method could still identify false positives (e.g. compounds that are part of coordinate structures), we chose the most restrictive parameterization of our syntax-based method to construct our dataset.

In § 4.4, we established the bracketing of n-ary compounds in our dataset based on the syntactic and semantic annotation of three linguistic resources, PCEDT, DeepBank and the noun phrase annotation by Vadas and Curran (2007). We showed that the three resources agree on the bracketing of 75% of the compounds in our bracketing subset (which consists of $6,244$ compounds). Possibly reflecting their genealogy, the pairwise agreement levels tend to be higher between PCEDT and Vadas and Curran's annotation than the other pairs. Furthermore, we observed that agreement between two of the resources (or even all three of them) does not always mean that these bracketing analyses are correct. That said, we believe that our bracketing

subset is an important addition to the resources on compound bracketing, not least because it facilitates future work on the integration of the compound bracketing and interpretation tasks.

In § 4.5, we constructed a variant of the dataset, whereby each compound is assigned two semantic relations, a PCEDT functor and NomBank argument or adjunct. We discussed the imbalanced distribution of the PCEDT and NomBank relations where relations like `ARG1` and `RSTR` annotate more than 50% of the compounds. We argued, however, that our goal to study compound interpretation as part of broader meaning representations mandates using the dataset as-is, instead of trying to somehow 'improve' the distribution of the relations. We also performed a quantitative analysis of the correspondence between PCEDT and NomBank relations based on our understanding of which relations are semantically comparable to some degree (§ 4.5.2). Our analysis was geared towards gauging cross-framework annotation similarity with no preconceptions of how such similarity should—or could—manifest itself. Therefore, even though we only found partial correspondence between the theoretically similar relations in NomBank and PCEDT, we defer our judgment on the utility of such a correspondence to Chapter 7, where we empirically evaluate them in transfer and multi-task learning experiments. The question of token vs. type semantics surfaced in our analysis of the PCEDT relations (§ 4.5.3). While we found some examples that support token-based interpretation in NomBank, there were clear indicators that the variation in PCEDT is in no small part due to annotation inconsistency. Lastly, in § 4.5.4, we defined the version of the dataset that will be used in our machine learning experiments in the following chapters. We will often refer to the PCEDT annotation of the compounds in our dataset as the PCEDT dataset and the NomBank annotation as the NomBank dataset. In addition to these two datasets, we will also use Tratz's (2011) dataset in our experiments in Chapters 5 and 6.

All in all, this chapter demonstrated how existing resources can be exploited to construct a dataset for compound bracketing as well as interpretation. While we chose to remain as faithful as possible to the original annotations, we identify a few interesting possibilities to improve the expressiveness of our dataset. For example, we might exploit the fact that PCEDT sometimes assigns multiple functors per compound type to reduce the number of compounds annotated with the underspecified functor `RSTR`. More concretely, one could replace all `RSTR`s with the most frequent non-`RSTR`

functor for the compounds that are annotated with both RSTR and non-RSTR functors. Even though this method would not change the annotation of the compounds that are annotated with RSTR only, it would still have an effect of 11% of the compound types overall. For NomBank, we could augment the arguments and modifiers with other relations from lexical resources like PropBank and VerbNet (Schuler, 2005). More specifically, about 60% of the senses in NomBank are annotated with a source field which specifies the source of the predicate in PropBank (i.e. the PropBank verbal predicate on which the NomBank predicate is based). One has to keep in mind, however, that the process of nominalization can affect the underlying verb's argument structure; that is, obligatory arguments of the original verb can become optional in the nominalization. Therefore, the mapping between NomBank frames and their source frames in PropBank may not be straightforward. Lastly, the arguments of some NomBank frames are also annotated with so-called thematic roles in VerbNet such as Agent, Instrument and Theme. Extracting such roles from NomBank would add another type of relations to almost 28% of the compound types in our dataset. The opportunities described above show how our approach to compound interpretation opens up for further integration with other existing and widely-used linguistic resources.

Chapter 5

Embeddings and Similarity-Based Classification

Word embeddings have proved successful in many NLP tasks to the extent that they have become one of the de facto standard input representations to neural networks. Indeed, in Chapters 6 and 7 we will use word embeddings to represent the constituents of noun–noun compounds in our neural network classification models. Therefore in this chapter, we focus on word embeddings themselves, introduce the process of training our embedding models and present baseline results of using them for interpretation in a similarity-based classification setup. In § 5.1, we detail the motivation behind using embeddings as our input representation and the experiments we conduct in this chapter. Further, in § 5.2, we give a brief background on distributional semantic models and word embeddings. We then, in § 5.3, introduce the embeddings models we train for all the upcoming experiments as well as the text pre-processing pipeline and (hyper-)parameters used to train the models. In § 5.4, we review two studies that contributed to popularizing the use of embedding models to recover relational similarity via simple vector arithmetic operations. We experiment with using the same vector arithmetic methods to predict the semantic relations of noun–noun compounds in § 5.5 using simple similarity-based techniques, such as the k-nearest neighbors algorithm in § 5.5.2.

Section 5.3 of this chapter builds on our joint work published in Fares et al. (2017).

5.1 Introduction and Motivation

In the past few years, word embeddings have become near-ubiquitous in NLP at large and proven to be more effective than the so-called count-based representations for many NLP tasks (Baroni et al., 2014). Perhaps one of the key reasons behind such widespread use of distributional semantic models in general is the fact that they can be automatically derived from large amounts of unlabeled text, which eliminates the need for manual annotation. As we will show in the following section, word embeddings often improve over the more traditional distributional semantic models while maintaining most of their desirable properties (except interpretability). In addition, several studies have demonstrated that word embeddings not only capture word similarities (or attributional similarity) but also some types of relational similarities between pairs of words (Mikolov, Yih, & Zweig, 2013; Levy & Goldberg, 2014).[1] Furthermore, a number of recent studies have also used word embeddings as input representation to their neural models for noun–noun compound interpretation in particular (Dima & Hinrichs, 2015; Shwartz & Waterson, 2018; Ponkiya et al., 2018). All of the above, in one way or another, motivate the choice of word embeddings to represent the constituents of noun–noun compounds in our upcoming experiments (cf. Chapters 6 and 7).

The goal of this chapter is twofold; first, we lay the foundation of the models we use to represent the constituents of noun–noun compounds; that is, we explain the basic ideas underlying word embeddings and present our methodological process to train word embeddings, including the specification of the linguistic pre-processing steps applied to the embedding training corpora and the model hyperparameters and properties. Second, we seek to determine to what extent word embeddings can be used to predict the semantic relations of compounds. Since it was shown that word embeddings capture some types of relational similarities (e.g. analogy relations), we ask whether word embeddings capture (some of) the semantic relations holding between the constituents of the compounds in the Tratz, NomBank and PCEDT datasets. To answer this question, we apply the vector arithmetic methods typically used to solve the semantic analogy task in a straightforward exemplar-based classification algorithm for noun–noun compound interpretation.

[1] Turney (2006) defines *attributional similarity* as the similarity between (the attributes) of two words, and *relational similarity* as the similarity between the relation holding between two pairs of words, such as analogy.

The similarity-based approach to compound interpretation can be seen as a simple baseline for the following experiments in Chapter 6. However, for our purpose, we use these experiments as a proxy to confirm whether or not the embedding models capture some level of semantic similarity (either relational or attributional) between the compounds. In other words, in part we use the experiments in § 5.5 to confirm the validity of our methodological decision to rely on word embeddings to represent noun–noun compounds.

5.2 Background

In this section, we introduce some of the concepts and assumptions underlying the distributional semantic models. In addition to establishing terminology, our goal does not go beyond a brief review of the long history of the distributional hypothesis and its applications in computational linguistics.[2]

The idea behind the distributional hypothesis can be summarized as follows: words that have similar meaning tend to occur in similar contexts (or similar linguistic environments). The *distributional hypothesis*—as a linguistic approach to represent meaning or difference therein—has deep roots in theoretical linguistics as well as philosophy, most notably in the works of Wittgenstein (1953) Harris (1954) and Firth (1957). The aforementioned three assumed varied definitions of 'distributional' and had different goals; for example, Harris (1954) proposed that language can be *structured* (i.e. its units can be grouped under different classes) based on a distributional analysis of the language. Harris spoke of difference in meaning in that words (or linguistic units in general) that have different meanings tend to have different co-occurrents. Firth (1957), on the other hand, focused on the notion of collocation and word sense disambiguation where the different senses of a polysemous word can be disambiguated based on the context it occurs in. Pulman (2013) traces the early interpretations, and applications, of the distributional hypothesis in computational linguistics to the late fifties and early sixties of the last century. For example, Pulman (2013) refers to the doctoral thesis of Karen Spärck Jones (1964) who was one of the pioneers of statistical natural language processing and information retrieval (IR), and it is in the field of IR that vector space models (VSMs) have become particularly

[2]For a comprehensive review, we refer to Sahlgren (2008), Turney and Pantel (2010), Pulman (2013) and Clark (2015), to name a few.

ubiquitous. VSMs, in the context of IR, are simply algebraic models that represent textual documents as vectors in an n-dimensional space whose dimensions correspond to the word types in the collection of documents at hand (i.e. the corpus). In an IR system, the documents as well as the user query are represented as vectors in the same space, and the distance between (or the directionality of) the query vector and document vectors correspond to their 'similarity' (or more accurately, the relevance of a given document to the user query).

Distributional semantic models (DSMs) make use of VSMs and the distributional hypothesis for lexical semantics; however, instead of creating one vector per document, DSMs work on the word type level.[3] That is, DSMs have one vector per word type representing the meaning of a given word based on its co-occurrents within some context (leading to a so-called term-term matrix in contrast to the term-document matrix in IR). The (semantic) similarity between two words is then computed in terms of the distance or angle between the vectors representing these words.[4] The word vectors are constructed by *counting* the number of times a given word co-occurs with others and then, optionally, applying some information-theoretic measures such as (positive) pointwise mutual information. The process of deriving or populating DSMs requires only large amounts of text, and of course computational power. With the availability of the latter, creating such models became much easier. However, sparsity (or linguistic creativity) and the curse of dimensionality remain among the major bottlenecks in creating and using these models, and this is where word embeddings are sometimes considered superior to the count-based DSMs.

Before we turn to word embeddings, it is important to highlight that long before the advent of word embeddings, matrix factorization methods (i.e. rank reduction methods) have been applied to count-based VSMs and DSMs to generate low-dimensional word representations from large co-occurrences

[3]We use "distributional semantic models" to refer to models that represent word meanings and "vector space models" to refer to the more general application of the distributional hypothesis in vector spaces, such as the term–document representation in IR. The distinction we make here is only for clarity, but in practice these terms are often used interchangeably in the literature, e.g. Turney and Pantel (2010); Kiela and Clark (2014).

[4]The terms *context* and *similarity* can be defined in multiple ways which have direct impact on what kind of relations can be uncovered from DSMs (e.g. syntagmatic vs. paradigmatic relations). However, here we assume familiarity with such concepts and definitions and refer to the work by Padó and Lapata (2003) and Sahlgren (2008), inter alios, for more details.

matrices. For example, latent semantic analysis (Deerwester et al., 1990, LSA) for term–document matrices and Hyperspace Analogue to Language (Lund & Burgess, 1996, HAL) for term–term matrices.

5.2.1 Word Embeddings

Like DSMs, word embeddings can be thought of as a mapping from V (a vocabulary or set of words) to $\mathbb{R}^d$ (d-dimensional space of real numbers), where each word w in V is represented by a real-valued vector $\vec{w}$ in the d-dimensional embedding space. However, unlike count-based DSMs, the vectors in word embeddings are dense and of much lower dimensionality (often between 50 and 600 dimensions). Moreover, the embedding dimensions do not correspond to context words and the vectors' values on a given dimension do not explicitly encode co-occurrence counts; which is one of the reasons word embeddings are considered obscure and harder to interpret than the count-based methods. More importantly, how such dimensions and values are learned is somewhat dependent on the embedding algorithm itself. Several algorithms have been proposed to learn neural language models—as part of the neural networks—and word embeddings (Collobert & Weston, 2008; Collobert, Weston, et al., 2011; Mikolov, Sutskever, et al., 2013; Pennington et al., 2014; Lebret & Collobert, 2014; Bojanowski et al., 2017). Among the most notable and widely used algorithms are the continuous bag-of-word (Mikolov, Chen, et al., 2013, CBOW) and skip-grams with negative sampling (Mikolov, Sutskever, et al., 2013, SGNS) algorithms which were popularized via the *word2vec* toolkit, and the Global Vectors algorithm (Pennington et al., 2014, GloVe). We use the latter algorithm to train our word embeddings in § 5.3, but we will very briefly describe the former two to contrast them with GloVe.

The objective in the SGNS algorithm is to learn a representation that is good at predicting a word's context given the word itself. Whereas the objective in CBOW, conversely, is to learn a representation to predict a word given its context. In either case, the ultimate goal is not the models themselves for those two types of prediction tasks, but rather the weights learned by the models, which make up the actual word representations. Word embeddings are, therefore, often referred to as prediction-based models, in contrast to the traditional count-based ones.

Pennington et al. (2014) describe both SGNS and CBOW as window-

based methods because they only rely on the 'local context' to learn word representations; i.e. they scan the corpus one window at a time (for some definition of window, e.g. five surrounding words) without considering global co-occurrence information throughout the corpus. GloVe, in contrast, is a matrix factorization method that uses global statistics of word co-occurrence in the corpus (hence the name Global Vectors). Indeed, the actual training of a GloVe model starts from a word co-occurrence matrix (i.e. the input to the training algorithm is a count-based term–term matrix) and the goal of the algorithm is to learn a low-rank matrix that captures as much information as possible from the original co-occurrence matrix.

The key insight in constructing a GloVe model is that the ratio of word co-occurrence probabilities—in comparison to raw probabilities—is "better able to distinguish relevant words [...] from irrelevant words" (Pennington et al., 2014, p. 1534). Based on this assumption, Pennington et al. (2014) define the learning algorithm as a weighted least squares problem, as follows:

$$J(\theta) = \sum_{i,j=1}^{V} f\Big(X_{ij}\Big)\Big(w_i^T \tilde{w}_j - log\ X_{ij}\Big)^2 \tag{5.1}$$

X is the word-word co-occurrence matrix, V is the vocabulary and f is a weighting function defined by Pennington et al. (2014) to downplay the effect of high co-occurrence counts.[5] The objective function J in Equation 5.1 aims to learn the set of parameters θ that minimizes the difference between the inner product of the word w_i and the context word $\tilde{w}_j$, on the one hand, and the log of the co-occurrence count X_{ij}, on the other, for all pairs of words (i, j) that co-occur in the corpus.[6] The details of arriving at Equation 5.1 can be found in Pennington et al. (2014).

From the above, it becomes clear that GloVe tries to combine the best of the count-based and prediction-based worlds by utilizing the non-zero entries in the *global* word-word co-occurrence matrix while requiring the word vectors to encode meaning by way of the ratio of co-occurrence probabilities. Further,

[5]The weighting function introduces two parameters, namely $x_{\max}$ and α, but we do not discuss them here since we use their default values—which were empirically defined by Pennington et al. (2014).

[6]We do not include the bias terms in Equation 5.1 for simplicity. However, they are actually important in order to redefine the log of the probability ratio in terms of the co-occurrence counts, i.e.: the bias terms account for $log(X_i)$ in $log(P_{ij}) = log(\frac{X_{ij}}{X_i}) = log(X_{ij}) - log(X_i)$. See Equations 6 through 8 in Pennington et al. (2014).

GloVe shows that the distinction between count-based and prediction-based method is in fact not clear-cut. In the following section, we present the collection of GloVe models that we have trained on a variety of training corpora with different choices of pre-processing steps and dimensionalities.

5.3 Word Embedding Models

In this section, we outline the process of training a total of eight GloVe models on different combinations of word form choices (full form vs. lemma), training corpora and a variety of dimensionalities.[7] Our decision to train several systematically varied GloVe models is motivated by the findings of Dima and Hinrichs (2015); Schnabel et al. (2015); Levy, Goldberg, and Dagan (2015) that certain embedding models seem to be more suitable for certain tasks and that some properties of the models do have an impact on performance of downstream tasks—such as compound interpretation in the case of Dima and Hinrichs (2015).

Levy, Goldberg, and Dagan (2015) report that some of the gains of prediction-based methods over count-based methods are in fact due to certain hyperparameter choices and design decisions. They hence stress the need for full transparency to facilitate fair comparison and reproducibility of embedding models. In a similar spirit, we argue that it is equally important to make explicit the design choices with regard to text pre-processing prior to training the actual embedding models (Fares et al., 2017).[8] This entails making explicit the pre-processing steps as well as tools used to prepare the training corpus. Therefore, in the following, we start by introducing our text pre-processing 'pipeline' and then move to training the GloVe models.

Pre-Processing Pipeline: The very first step in training word embedding models is processing the training data; that is, to extract and clean the textual content—if needed—and then sentence-split and tokenize the text. Sentence-splitting and tokenization are arguably the minimum pre-processing steps

[7]Some of the embedding models presented in this section are part of a larger initiative by the Nordic Language Processing Laboratory (NLPL). The full repository of word vectors is described in Fares et al. (2017) and can be accessed at `http://vectors.nlpl.eu`.

[8]In Fares et al. (2017), we show that lemmatization does have a potentially large effect on the embedding model's performance on so-called 'intrinsic' evaluation tasks such as word similarity and analogy. See Table 1 in the aforementioned work.

required before training an embedding model. Other steps include, inter alia: stop-word removal, number normalization (i.e. replacing digits with a specific numerical value such as zero), downcasing, phrase identification, named-entity recognition and morphosyntactic annotations such as part-of-speech tagging.

We use two corpora and a concatenation thereof to train the GloVe models: the English Wikipedia dump from September 2016 (about 2 billion word tokens) and English Gigaword Fifth Edition (about 4.8 billion word tokens) (Parker et al., 2011). Wikipedia dumps consist of 'MediaWiki' and HTML markup, and hence we use a tool called *WikiExtractor* to extract and clean the textual content from the Wikipedia dump.[9] In terms of linguistic pre-processing, we sentence-split, tokenize, and lemmatize the text in Wikipedia and Gigaword using the Stanford CoreNLP Toolkit, version 3.6.0 (Manning et al., 2014). Furthermore, we also remove all stop-words using the stop-word list defined in the Natural Language Toolkit (Bird et al., 2009, NLTK).

Those pre-processing steps above lead to two types of training corpora in terms of their word form, viz. lemmatized and full form, which is the first aspect in which our GloVe models vary. That is, the models can be trained on lemmatized or non-lemmatized versions of the corpora. Second, the models differ in the training text: Wikipedia, Gigaword and the concatenation of both (Wiki+Giga henceforth). Third, the model can be of different dimensionalities, viz. 50, 100, 300, 600 or 1000 dimensions. We train eight GloVe models that allow us to study the effect of these three properties independently, i.e. surface form, training data and dimensionality. The models are listed in Table 5.1.

All the GloVe models are trained using the reference implementation published by Pennington et al. (2014). Training a GloVe model consists of four steps: (1) collecting unigram vocabulary counts, (2) constructing word–word co-occurrence matrix, (3) shuffling the co-occurrence matrix (row-wise) and (4) training the GloVe model on the co-occurrence matrix. To ensure replicability—inasmuch as the embedding algorithm allows it—we 'pre-shuffle' the training corpora in the same way for all the models.[10] Hence, we skip the third step of shuffling the co-occurrence matrix. Further, in step (1), we define minimum frequency thresholds for the three corpora, and thus we ignore words that occur less than 80 times in Wikipedia, 100 times in Gigaword and

[9]The tool can be found on: `https://github.com/attardi/wikiextractor`. Accessed: 16 January 2019.

[10]Pre-shuffling was also important to align—as much as possible—the word2vec and GloVe models we compare in Fares et al. (2017).

Training Data	Word Form	Dimensions	Vocab Size	Freq. Cutoff
Wikipedia	Full form	300	302815	80
Gigaword	Full form	300	292967	100
Wiki+Giga	Full form	300	291392	200
Wiki+Giga	Lemma	50	260073	200
Wiki+Giga	Lemma	100	260073	200
Wiki+Giga	Lemma	300	260073	200
Wiki+Giga	Lemma	600	260073	200
Wiki+Giga	Lemma	1000	260073	200

Table 5.1: List of GloVe models. Number of training iterations: 50 if *dimensions* $<$ 300; 100 iterations otherwise. Symmetric context window size for all the models: 5.

200 times in Wiki+Giga. In step (2), we use a symmetric context window of size five to construct the co-occurrence matrix for all the models. For the training step, we use the default values for the parameters of the weighting function ($x_{\max} = 100$ and $\alpha = 0.75$). We set the number of training iterations to 50 for models with less than 300 dimensions and 100 iterations otherwise, as recommended by Pennington et al. (2014).

All the resulting models will be used in § 6.3, but for the experiments in this chapter we will report on using the model trained on the full forms of Wiki+Giga (i.e. the third model from the top in Table 5.1).

5.4 Linguistic Regularities in Embeddings

It is perhaps no surprise that word embeddings capture the attributional similarity between words, such as synonymy; less expectedly though, word embeddings also capture certain types of relational similarity as shown by Mikolov, Yih, and Zweig (2013); Levy and Goldberg (2014), among others. In the following, we briefly review the findings of these two studies and explore if and how a similar approach can be used to predict the semantic relations of noun–noun compounds.

Mikolov, Yih, and Zweig (2013) report that word embedding models capture syntactic and semantic regularities (i.e. relational similarity) that can be recovered by applying simple vector arithmetic operations. The most famous example of such regularities is the observation that subtracting the

vector of the word *man* from the vector of *king* and adding the vector of *woman* results in a vector whose nearest neighbor is the vector of the word *queen* (excluding the vectors of the words *king*, *man* and *woman*); that is, if w_i is the vector for the word i, the observation translates to:

$$w_{\text{king}} - w_{\text{man}} + w_{\text{woman}} \approx w_{\text{queen}}$$

Mikolov, Yih, and Zweig (2013) formulate the task of identifying syntactic and semantic regularities in word embeddings as an analogy task and introduce a dataset of 800 analogy questions that capture morpho-syntactic relations (the MSR dataset henceforth). Each of these questions consists of two pairs of words, say a:b and c:d, that share a particular relation and the question is then: given the pair a:b and the word c, find the word d that satisfies the analogy "a is to b as c is to d". In addition, Mikolov, Chen, et al. (2013) design another analogy dataset that contains $19,544$ questions covering seven semantic relations and seven morpho-syntactic ones; this dataset is referred to as the Google Analogies Dataset.

To solve the analogy questions in both datasets, Mikolov, Yih, and Zweig (2013) compute the vector $y = w_b - w_a + w_c$ and search for its nearest neighbor in the embedding space in terms of cosine similarity. They refer to this method as the vector offset method, and it can expressed in terms of word vectors as shown in Equation 5.2.

$$\underset{d \in V \setminus \{a,b,c\}}{\arg\max} \left(cos(w_d, w_b - w_a + w_c)\right) \tag{5.2}$$

Levy and Goldberg (2014) redefine the vector offset method by Mikolov, Yih, and Zweig (2013) to recover relational similarity as a combination of three pairwise word similarities, i.e. they express relational similarity in terms of attributional similarity. Hence, Equation 5.2 can be rewritten as follows:

$$\underset{d \in V \setminus \{a,b,c\}}{\arg\max} \left(cos(w_d, w_b) - cos(w_d, w_a) + cos(w_d, w_c)\right) \tag{5.3}$$

Levy and Goldberg (2014) refer to the method described in Equation 5.3 as 3CosAdd; we will follow the same terminology here as well. The second vector arithmetic method Levy and Goldberg (2014) introduce is called

PairDirection, and it requires the two pairs of words to share the same direction, as shown in Equation 5.4.[11]

$$\arg\max_{d \in V \setminus \{a,b,c\}} (cos(w_d - w_c, w_b - w_a)) \tag{5.4}$$

Levy and Goldberg (2014) state that even though Mikolov, Yih, and Zweig (2013) do not mention PairDirection in their paper, they do use it to solve the semantic analogy questions, while they use 3CosAdd to solve the syntactic ones.

The vector arithmetic methods by Mikolov, Yih, and Zweig (2013) and Levy and Goldberg (2014) outperform the previous state-of-the-art on the SemEval-2012 Task 2: Measuring Relation Similarity (Jurgens et al., 2012).[12] Word embedding models have been often compared in light of their performance on the analogies datasets mentioned above and other word similarity datasets such as SimLex-999 (Hill et al., 2015)—these evaluations are sometimes referred to as 'intrinsic' evaluation of word embeddings. However, those comparisons were not always fair; for example, the competing embedding models are sometimes trained on different corpora as highlighted by Levy, Goldberg, and Dagan (2015). Further, the very approach of evaluating embeddings on word similarity (and analogy) has been criticized for—among other reasons—its weak correlation with extrinsic evaluation measures (Faruqui et al., 2016). For our part, we will evaluate word embeddings with respect to their impact on the performance on our neural classifier in the context of noun–noun compound interpretation (cf. § 6.3). That said, we refer to Fares et al. (2017) for a carefully designed comparison between some of the GloVe models presented in this chapter and SGNS word2vec models, based on their performance on the Google Analogies Dataset and SimLex-999.

Having discussed how vector arithmetic operations are used to solve semantic and syntactic analogy tasks, the following section addresses ways of using the same methods to predict the semantic relations of noun–noun

[11] Levy and Goldberg (2014) define a third vector arithmetic method based on a multiplicative combination of the vectors called 3CosMul. We do not include this method in our review because we do not use it in our experiments.

[12] Interestingly, Levy and Goldberg (2014, p. 172) report that the traditional count-based models also capture some form of relational similarity, and hence they claim that word embeddings are not "discovering novel patterns, but rather is doing a remarkable job at preserving the patterns inherent in the word-context co-occurrence matrix."

compounds.

5.5 Vector Arithmetic for Compounds

The task of noun–noun compound interpretation can be construed as an analogy problem wherein the compound semantic relations correspond to the analogy relations. Therefore, we utilize the same methods proposed for solving the analogy problem in word embeddings, with the aim of establishing a very simple baseline and gauging the 'representational' ability of our embedding models in the context of noun–noun compound interpretation. Compound interpretation, as discussed in Chapter 2, is a multiclass classification problem: given an unseen compound, our goal is to predict its semantic relation based on the examples seen before (i.e. in the training data).

To cast compound interpretation as an analogy problem, we assume that the two noun constituents in each compound make up the word pairs in the analogy problem. In concrete terms, in the analogy problem we have two pairs of words (a:b and c:d) which exhibit a shared relation (e.g. the gender relation in *man*:*woman* and *king*:*queen*). In the compound interpretation problem, we have two compounds $a\ b$ and $c\ d$ that are annotated with the same relation, and we take this relation to play the role of the 'analogy' relation. The difference, however, is that in the analogy problem we are looking for d in V so that "a is to b as c is to d" given $a : b$ and c, whereas in compound interpretation, we are looking for the compound $c\ d$ in the training data with the highest relational similarity to the compound $a\ b$. To find the compound that yields the highest relational similarity, we apply the vector arithmetic methods 3CosAdd and PairDirection. Said differently, in the analogy problem we aim to find the word in the embedding vocabulary that satisfies the analogy relation, whereas in compound interpretation we do not start from the embedding vocabulary itself, but rather from the training split and then use the embedding model to compute the relational similarities. Hence, the search in compound interpretation is more 'restricted' than in the analogy problem. In addition, what matters in compound interpretation is the relation of the most similar compound, not the compound itself.

Hence, assuming C is the set of compounds in the training split where each instance is a pair of nouns and c, d is the test compound, we can rewrite the vector arithmetic methods 3CosAdd and PairDirection as shown in Equations 5.5 and 5.6, respectively:

$$\arg\max_{a,b \in C}(cos(w_d, w_b) - cos(w_d, w_a) + cos(w_d, w_c)) \tag{5.5}$$

$$\arg\max_{a,b \in C}(cos(w_d - w_c, w_b - w_a)) \tag{5.6}$$

Based on the above, predicting the relation of a given compound amounts to finding its most 'relationally' similar compound in the training split, using Equations 5.5 and 5.6, and then returning the *relation* of that compound.

As mentioned in the beginning of this section, compound interpretation can be seen as an analogy problem. To verify this assumption, we define an additional similarity measure that only relies on attributional similarity (i.e. word similarity) between the compounds' constituents, instead of relational similarity. Given two compounds, *a b* and *c d*, we define compound similarity as the sum of the pairwise similarities between their left- and right-constituents, as shown in Equation 5.7. We will refer to this measure as SimCom.

$$\textsc{SimCom}(a\ b, c\ d) = cos(w_a, w_c) + cos(w_b, w_d) \tag{5.7}$$

Classification using SimCom, like in the previous methods, looks for the compound in the training split that maximizes SimCom given an input compound *c d*, i.e.:

$$\arg\max_{a,b \in C} \textsc{SimCom}(a\ b, c\ d)$$

Now that we have three similarity-based methods to interpret noun–noun compounds, the remaining part of this chapter will focus the experimental results of using these methods on the Tratz, NomBank and PCEDT datasets.

5.5.1 Experimental Results

Before presenting the results of the three similarity-based methods to predict compound relations, we briefly outline our experimental setup. The similarity-based classification algorithm is memory-based, which means there is no training or learning prior to the actual classification step. We conduct our

Measure	Tratz	NomBank	PCEDT
3CosAdd	55.42	70.76	43.26
PairDirection	55.57	70.65	46.96
SimCom	**59.38**	**71.85**	46.41
Majority baseline	17.08	67.61	**52.28**

Table 5.2: Accuracy of the three similarity-based methods on the development splits of the three datasets. The GloVe model is trained on the full forms of Wiki+Giga and of 300 dimensions.

experiments on the training and development splits of the Tratz, NomBank and PCEDT datasets. The training splits of the three datasets (cf. § 4.5.4) make up the training samples or instances, and the compounds in the development splits are used for evaluation.[13]

For all the experiments in this section and the following one, we use the 300-dimensional GloVe model trained on the non-lemmatized concatenation of Wikipedia and Gigaword (Wiki+Giga), i.e. the third model in Table 5.1. Further, all the word vectors are normalized to unit length (aka ℓ_2-norm) which transforms the cosine similarity to a simple dot product of the normalized vectors.

Table 5.2 shows the accuracy results of using the three similarity measures 3CosAdd, PairDirection and SimCom to predict the relations of the compounds in the development splits of Tratz, NomBank and PCEDT.

The first thing to observe in Table 5.2 is that the SimCom method leads to remarkably better results on Tratz compared to the other two relational similarity methods. SimCom also leads to an improved accuracy on NomBank but the difference between SimCom and the other two methods is about one point (in contrast to 3.8 points on Tratz). Second, the relational similarity methods lead to comparable results on both Tratz and PCEDT.[14] Third, on PCEDT, it is the 3CosAdd method that leads to the worst accuracy, while the other two methods yield more or less the same accuracy.

[13]For the Tratz (2011) dataset, we use the training and development splits as defined by Dima and Hinrichs (2015), cf. § 6.1.

[14]Levy and Goldberg (2014) report that the PairDirection method outperforms 3CosAdd on restricted vocabulary, but we do not see any major differences between the two. That said, our task and datasets are different from theirs, and hence it is not necessarily surprising to see a different pattern here.

To better understand these results, however, one needs to take into account the accuracy of the majority class baseline shown in the last row of Table 5.2. The accuracy of similarity-based methods on PCEDT is lower than that of the majority baseline classifier. Furthermore, the similarity-based accuracy on NomBank is a few points higher than the majority-class baseline. The high accuracy of majority baseline on NomBank and PCEDT is not surprising given what we know about the relation distributions in these two datasets (cf. Tables 4.5 and 4.6). However, the fact that the similarity-based methods perform worse than the majority baseline on PCEDT—in contrast to NomBank whose majority relation is even more frequent—can be interpreted as an indicator of the difficulty of learning the PCEDT relations. The distribution of the Tratz relations is more balanced than the other two datasets and the performance of the similarity-based methods is well above the majority baseline.

While we do not report per-relation F_1 scores in this chapter, it is important to highlight here that the similarity-based methods achieve relatively high scores on the relations with the least diverse set of constituents in Tratz such as `MEASURE`. We observe a similar pattern in PCEDT and NomBank where the relations of high constituent diversity have a low per-relation F_1 score whereas the ones with low constituent diversity have a high score (we detail the constituent diversity per relation in Figure 7.3 in Chapter 7). These observations indicate that the similarity-based methods rely on attributional similarity rather than relational similarity to predict the relations in the three datasets. The fact that the SimCom method leads to better accuracy on the Tratz dataset further supports the aforementioned conclusion. This conclusion, however, is not unexpected given the somewhat 'generic' nature of some of the semantic relations in the three datasets; for example, the relations denoting actor (`ARG0` in NomBank and `ACT` in PCEDT) are likely to be too abstract to be encoded in word embeddings. Furthermore, Levy and Goldberg (2014) in fact report that semantic analogy relations are more difficult to recover than others, for example the currency relations in the Google Analogies Dataset.

5.5.2 K-Nearest Neighbors

To complement our similarity-based perspective, in this section we present the results of using the k-nearest neighbors (k-NN) algorithm together with

k	Tratz	NomBank	PCEDT
1	55.57	70.65	46.96
3	60.68	**76.41**	49.89
5	61.77	75.11	54.13
7	62.40	75.54	54.46
9	**63.28**	75.33	**55.54**
11	62.60	75.33	55.43

Table 5.3: Accuracy results of PairDirection-based similarity for different values of k. The GloVe embedding model is trained on the full forms of Wiki+Giga and of 300 dimensions.

one of the relational similarity methods (viz. PairDirection) to predict the compound semantic relations in Tratz, NomBank and PCEDT. We experiment with multiple odd values of k, however since our classification problem is not binary, using odd values of k would not break the ties in finding the majority class among the k-nearest neighbors. Therefore, we implement a distance-weighted version of k-NN, where the neighbors are weighted according to their similarity to the compound that has to be classified.

Table 5.3 shows the accuracy results when considering an increasing number of k-nearest neighbors. The similarity of the nearest neighbor is computed using the PairDirection measure in the embedding space of the 300-dimensional GloVe model trained on the non-lemmatized form of Wiki+Giga (the same model used in the previous section).

As can be seen from Table 5.3, the accuracy on the three datasets increases by considering more than just the one nearest compound in the training data. For NomBank, the ideal number of nearest neighbors seems to be 3; whereas the highest accuracy on Tratz and PCEDT is achieved when $k = 9$. However, given the distribution of the relations in NomBank and PCEDT, one has to be careful interpreting the accuracy results in isolation. More concretely, while the improvement in accuracy on NomBank is due to better predictions of all the relations when $k = 3$, as soon as k exceeds 7 the improvement becomes more and more limited to correct predictions of the most frequent relations (`ARG1`); for example, when $k = 9$ we see 98% recall in predicting `ARG1` and 67% precision. Seeing `ARG1` among the nearest neighbors is, of course, bound to happen with increasingly higher values of k, because `ARG1` amounts to 67.61% of the training examples. The same holds for PCEDT where increasing the

number of nearest neighbors does improve the predictions of almost all the relations but after $k = 9$ we start to see some harmful effect on some relations, while RSTR (the most frequent relation) continues to improve.[15] The Tratz dataset is more balanced and hence increasing k does not lead to a similar effect.

Contrasting the results in this section and the previous one, it is clear that if one is to use a similarity-based approach for compound interpretation, it is important to rely on more than the one nearest neighbor.

5.6 Conclusion

In this chapter, we gave a high-level overview of distributional semantic models, both in the traditional sense (i.e. count-based models) and the more recent prediction-based models. In the latter, we focused on the Global Vectors (GloVe) algorithm by Pennington et al. (2014) since we use it to train our own embedding models. In § 5.3, we introduced eight GloVe models and detailed how we trained them on a variety of training corpora, with different dimensionalities and text pre-processing steps. These models will be put to use in the following chapter, where we try to quantify their impact—as input representation—on the final performance of a neural classifier for noun–noun compound relations.

In § 5.4, we reviewed two of the relevant studies on recovering relational similarities (or linguistic regularities) from word embeddings using vector arithmetic methods. And in § 5.5, we proposed a way to use the same vector arithmetic methods to interpret noun–noun compounds and defined an additional method based on the attributional similarity of the compound's constituents. In § 5.5.1, we presented the results of using one of our embedding models in a similarity-based approach to predict the semantic relations of noun–noun compounds in the Tratz, NomBank and PCEDT datasets. The results indicated that the embedding models have a limited ability to encode the 'abstract' semantic relations in the three datasets, and that by simply relying on the similarity between the compound's constituents we could achieve better accuracy results (at least in the case of Tratz and NomBank). Although we have only reported on the use of one embedding model in our

[15] As mentioned before, the imbalanced distribution of relations calls for another evaluation metric, such as macro-averaged F_1 score, but since the experiments in this chapter are exploratory we will leave the detailed evaluation to the following chapters.

experiments in this chapter, we did run pilot experiments using almost all the embedding models with our similarity-based approach. One important observation that surfaced from these experiments is that the results vary depending on the embedding model, and therefore in the following chapter (in § 6.3) we systematically experiment with using all the GloVe models as input to our neural classifier. In § 5.3, we experimented with using the k-NN algorithm with the PairDirection method to try and reach a final conclusion regarding the embeddings' ability to capture relational similarities in terms of semantic relations of nominal compounds. While our k-NN experiments showed some improvement in accuracy across the three datasets as we increased the number of nearest neighbors, these improvements were mainly due to 'better' predictions of the most frequent relations in NomBank and PCEDT.

Finally, the goal of our experiments in this chapter was to gauge the potential of using the GloVe models as input representation for our neural classification model. As mentioned above, the models seem unable to capture the notions of relational similarities as defined by the compound relations in Tratz, NomBank and PCEDT. That said, the GloVe models were able to capture attributional similarities between the compound constitutes (as evidenced by the high F_1 score on the least lexically diverse relations). In Chapter 6, we aim to learn these relational similarities using neural networks with word embeddings as input representation.

Chapter 6

Neural Classification

In this chapter we present a comprehensive series of experiments on using neural network models to learn the semantic interpretation of noun–noun compounds. The experiments in this chapter are geared towards determining which aspects or properties of the neural classification model affect the overall accuracy, among other metrics, across three compound datasets, viz. Tratz, NomBank and PCEDT. Moreover, throughout the chapter we take special care to reflect on the methodological aspects of both the experiments and evaluation measures. We start the chapter by reviewing the highly relevant work by Dima and Hinrichs (2015), the first study on neural compound interpretation, which we then replicate as a first step towards designing a neural network model for our dataset. In § 6.1.1, we present the results of replicating the experiments conducted by Dima and Hinrichs (2015) and compare our results to theirs. In § 6.3, we systematically study the interplay between the performance of the neural classifier and a selection of properties of word embeddings (when used as input representation to the neural classifier). In § 6.4, we experiment with different neural architectures to gauge the role of the head and modifier nouns in predicting the compound relations. We conduct a series of experiments to validate our choice of hyperparameters and their impact on classification accuracy across three noun–noun compound datasets in § 6.5. Thereafter, we explore the effect of using additional lexical semantic features as input to the neural classification model in § 6.6. Finally, in 6.7, we investigate the learning curves of the neural classification model on the three compound datasets and quantify the effect of random initialization on the final performance of the classifier.

6.1 Background: Dima and Hinrichs (2015)

Dima and Hinrichs (2015) use a feedforward neural classifier to predict the semantic relations of the compounds defined in a revised version of the dataset by Tratz and Hovy (2010). In addition, they use pre-trained word embeddings to represent the constituents of the compounds. In the following, we will unpack the details around (1) the compound dataset, (2) their use of word embeddings and (3) the neural classification model used by Dima and Hinrichs (2015).

First, Dima and Hinrichs (2015) use the dataset introduced by Tratz (2011), which is a revised and publicly available version of the original dataset by Tratz and Hovy (2010) (cf. § 3.2.3). The major differences between these two versions of the dataset are: first, the revised dataset contains more compound examples (19,158 compounds) than the original one (17,509 compounds); second, the relation inventory in the revised dataset consists of 37 semantic relations whereas the original one consists of 43 relations. As Dima and Hinrichs (2015) report, Tratz (2011) did not (re)evaluate the model described by Tratz and Hovy (2010) on the revised dataset, which makes the comparison between the results reported by Dima and Hinrichs (2015) and Tratz and Hovy (2010) less straightforward. To distinguish between the two versions of the dataset, we follow the same naming convention used by Dima and Hinrichs (2015) where the revised version is referred to as the "Tratz dataset" and the original version of the dataset is referred to as the "Tratz and Hovy dataset".

Dima and Hinrichs (2015) pre-process the Tratz dataset before using it in their experiments. These pre-processing steps include lower-casing all words (the Tratz dataset has some names which are capitalized) and normalizing the spelling of some complex words like *health care* (which occurs in different forms in the dataset, viz. *health care* and *healthcare*). After applying all the pre-processing steps, Dima and Hinrichs (2015) end up with a constituent dictionary of size $V = 5,242$. They split the Tratz dataset into three parts: train (13,409 compounds), development (1,920 compounds) and test (3,829 compounds).

Dima and Hinrichs (2015) use four pre-trained, publicly available word embedding models that differ in their vector dimensionality, vocabulary size and training data size and genre. The embedding models were also trained using different embedding algorithms proposed by Collobert, Kavukcuoglu,

and Farabet (2011); Pennington et al. (2014); Lebret and Collobert (2014); Mikolov, Sutskever, et al. (2013)—some of these algorithms and models were briefly discussed in § 5.2.1. The properties of the embedding models used by Dima and Hinrichs (2015) are presented in Table 6.1 (which is adapted from Table 2 in the aforementioned article). Dima and Hinrichs (2015, p. 177) report that in the case of embedding models that come with multiple options for vector dimensionality (e.g. the GloVe models) "it was always the highest dimensional embedding that gave the best results in the cross-validation setup".

Name	Dimensions	Dict. size	Data size	Support corpora
CW	50	130,000	0.85bn	enWikipedia + Reuters RCV1
GloVe	300	400,000	42bn	Common Crawl (42bn)
HPCA	200	178,080	1.65bn	enWikipedia + Reuters + WSJ
word2vec	300	3,000,000	100bn	Google News dataset

Table 6.1: Overview of the word embedding models used by Dima and Hinrichs (2015). CW: Word embeddings by Collobert, Kavukcuoglu, and Farabet (2011); GloVe models by Pennington et al. (2014); HPCA: Embeddings obtained by applying Hellinger PCA to the word co-occurence matrix (Lebret & Collobert, 2014); word2vec: Continuous skip-gram model by Mikolov, Sutskever, et al. (2013). Support corpora: the data used to train the embeddings.

The neural classifier implemented by Dima and Hinrichs (2015) consists of four layers: (1) input layer, (2) lookup table (aka embedding layer), (3) hidden layer and (4) output layer. The architecture of the model is shown in Figure 6.1. The input layer is simply to specify the index (in the lookup table) of the word embedding vector representing one of the constituents of a given compound. Hence, the size of the input layer is two (one index per constituent). The second layer (the lookup table) is where the word embedding vectors are actually stored—merely a $V \times d_e$ matrix of real-valued numbers, where d_e is the dimensionality of the word embeddings. As detailed later, it is essential to have the embedding vectors as part of the neural network model to be able to modify them via backpropagation as the model is being trained. The vectors selected from the lookup table (by the input indexes) are concatenated and fed to the hidden layer, which is fully connected with the lookup layer. Dima and Hinrichs (2015) empirically determined the size of the hidden layer

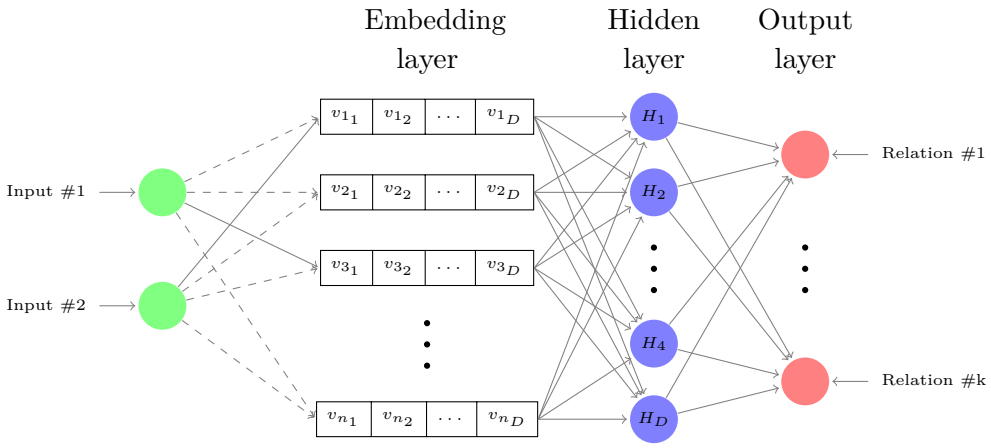


Figure 6.1: Architecture of the neural classifier by Dima and Hinrichs (2015)

to be the same as the size of the embedding vectors (i.e. same number of dimensions). Finally, the output of the hidden layer is passed to the output layer whose size equals the number of relations in the Tratz dataset. The *Softmax* function in the output layer chooses the most likely relation based on the scores assigned by the network. Dima and Hinrichs (2015) determined the model hyperparameters empirically based on the results of their experiments on the development set. We return to the values of these hyperparameters in § 6.1.1 as we present our replication of their model and experiments.

One of the interesting questions Dima and Hinrichs (2015) pose in their work is whether or not updating (fine-tuning) the embedding weights (as part of training the neural classifier) contributes to boosting the accuracy of the neural classifier. Consequently, they experiment with two types of architecture: one with fine-tuning (where the embedding weights are updated) and another one without fine-tuning (where the embedding weights are frozen, i.e. left unchanged). We will often refer to the former as the fine-tuned model and to the latter as the static model.

In total, Dima and Hinrichs (2015) experiment with four embedding models to represent the compounds and two neural architectures. The results of these experiments are shown in Table 6.2 (which is based on Table 3 in their paper); all the numbers reported in the table represent the micro-averaged F_1 scores of the models.[1] Under the columns DEV(NO-F) and DEV(F), the

[1] As Dima and Hinrichs (2015, p. 179) explain: "The classification task at hand is an instance of *one-of* or *multinomial* classification, therefore micro-averaged F_1 is the same as the accuracy."

Embedding model	DEV(NO-F)	DEV(F)	TEST(NO-F)	TEST(F)
CW-50	65.83	78.39	66.62	76.03
GloVe-300	74.17	77.81	76.05	75.87
HPCA-200	61.45	77.14	64.56	76.00
word2vec-300	71.46	73.54	71.38	71.59
Random embedding	-	74.17	-	71.43
CW-50+GloVe-300+HPCA-200	79.01	79.48	78.14	77.12

Table 6.2: Accuracy results of the experiments reported by Dima and Hinrichs (2015). NO-F: No fine-tuning of word embedding. F: Word embeddings were fine-tuned during training.

models were trained on the training split and evaluated on the development split. Under the columns TEST(NO-F) and TEST(F) the models were trained on both the training and development splits and evaluated on the test split. It is important to highlight here that their decision to (re)train the model on both the training and development splits not only render the results on development incomparable to test, but also makes it difficult to gauge potential overfitting effects—we will elaborate on this point in the following section.

The first four rows in Table 6.2 correspond to the results of using the four embedding models introduced earlier; the fifth row represents the results of using random embeddings as input to the neural classifier (in this case, it only makes sense to report the results of using the fine-tuned random embeddings). The last row shows the result of using the concatenation of the embedding vectors from three models.

The first thing to observe from Table 6.2, and which is also reported by Dima and Hinrichs (2015), is that the model that uses the concatenation of the three word embedding vectors achieves the best accuracy with and without fine-tuning on both the development and test splits. However, as Dima and Hinrichs (2015, p. 180) explain, in this setup the size of the network grows considerably since the size of the hidden layer is assumed to be the same as the dimensionality of the input vector (550 in this case), and this leads to "a much more difficult training task when compared to more compact representations". Dima and Hinrichs (2015) do not elaborate on the meaning of "difficult training task" in this context, but it is obvious that the computational cost of training the neural classifier increases as the

dimensionality of word embedding vectors, and consequently the size of the hidden layer, increase. In addition, large neural networks (i.e. models with a large number of parameters) are more prone to overfitting than smaller models.

Second, as evident in Table 6.2 and as reported by Dima and Hinrichs (2015), fine-tuning of word embeddings helps improve the prediction accuracy of the model in most cases. This effect is especially remarkable in the case of lower dimensional embeddings like *CW-50* where fine-tuning leads to a solid ten-point increase in accuracy on the test set. However, the improvements become less pronounced as the dimensionality of the word embedding model increases. That said, as Dima and Hinrichs (2015) point out, vector dimensionality is not the only difference between the four models; in fact, the size of the word embedding training data (i.e. support corpora in Table 6.1) varies drastically between *CW* and *HPCA*, on the one hand, and *GloVe* and *word2vec*, on the other hand. Dima and Hinrichs (2015, p. 180) conclude that the embedding models trained on large amounts of data are "better suited for direct use in new tasks and have less to gain from fine-tuning". Interestingly, however, using randomly initialized embeddings leads to an accuracy on par with the accuracy achieved using the *word2vec* model, which is trained on large amounts of data. We expect major differences in accuracies on (partly) unseen compounds; i.e., compounds whose constituents do not occur in the training data, because they would not have a meaningful representation in the randomly initialized embeddings.

Dima and Hinrichs (2015) report that their model outperforms that of Tratz and Hovy (2010) on unseen compounds by a wide margin (i.e. compounds that contain constituents only seen in the test set, cf. § 3.2.3). As explained earlier, however, the test datasets are not directly comparable due to the difference in the number of compound instances and semantic relations. Nonetheless, given that Dima and Hinrichs (2015) report an accuracy of 77.12% on unseen compounds while Tratz and Hovy (2010) report 51% accuracy on unseen compounds in a similar—though not identical—test set, one can stipulate that the model by Dima and Hinrichs (2015) is more robust in classifying unseen compounds. The use of pre-trained word embeddings partly explains the superior results on unseen compounds by Dima and Hinrichs (2015) as they encode 'features' about the semantic similarity of thousands of words even if they are unseen in the training set.

6.1.1 Replicating Dima and Hinrichs (2015)

As reported in the previous section, the neural model introduced by Dima and Hinrichs (2015) leads to robust results when tested on unseen compounds, which makes their choice of model architecture and hyperparameters a good starting point to design a neural classification model for our datasets.

Even though we do not necessarily aim to reproduce the exact results reported by Dima and Hinrichs (2015), in this section we try to replicate their setup as closely as possible to see if we arrive at comparable results, and hence validate our implementation. To do so, first, we implement the exact same network architecture described in the previous section, i.e. a feedforward neural network of four layers: an input layer, a lookup table (an embedding layer), a hidden layer and an output layer. Second, we reuse the exact same version of the Tratz dataset as well as the same data splits used in their experiments.[2] Third, whenever possible, we use the same hyperparameters specified in Dima and Hinrichs (2015); however, as explained below, this is not always possible due to using different neural network implementation frameworks. Fourth, we use two of the word embedding models they use to represent the constituents of the noun–noun compounds.

Table 6.3 summarizes the similarities and differences between the hyperparameters and other properties of the model by Dima and Hinrichs (2015) and ours. The parameters in Table 6.3 are self-explanatory, however, it is worthwhile commenting on two points. First, in terms of hyperparameters the only difference between the two models in Table 6.3 is the optimization function (or the optimizer); Dima and Hinrichs (2015) use Averaged Stochastic Gradient Descent (ASGD), whereas we experiment with an optimization function called Adaptive Moment Estimation (Adam; Kingma & Ba, 2015). The reason we have a different optimization function is simply because we use different libraries for neural networks. Dima and Hinrichs (2015) implement their model using *Torch7*, an earlier scientific computing framework (Collobert, Kavukcuoglu, & Farabet, 2011), whereas we implement ours using *Keras*, a Python deep learning library with *TensorFlow* as backend.[3] The optimization function used by Dima and Hinrichs (2015), ASGD, is not implemented in Keras (at the time of conducting our experiments). Second, Dima and Hinrichs (2015) use the early stopping criterion proposed by Prechelt (2012) to

[2]We are grateful to Corina Dima for sharing the pre-processed dataset with us.

[3]Keras: The Python Deep Learning library `www.keras.io`

	Dima & Hinrichs	Ours
Hidden layer activation	Sigmoid	Sigmoid
Hidden layer size	300	300
Output layer activation	Softmax	Softmax
Optimizer	Averaged SGD, $\eta = 0.9$	Adam $\eta = default$
Loss function	Negative log likelihood	Negative log likelihood
Batch size	5	5
Epochs	Early stopping criterion	Early stopping criterion

Table 6.3: Comparison of hyperparameters between our implementation of Dima and Hinrichs (2015) and their original model.

Embedding model	DEV(NO-F)	DEV(F)	TEST(NO-F)	TEST(F)
GloVe-300	74.43	77.50	73.31	75.32
word2vec-300	75.99	77.34	73.94	75.16

Table 6.4: Results of evaluating our replication of the Dima and Hinrichs (2015) model on the Tratz dataset. See Table 6.1 for details about the embedding models used here.

avoid overfitting, in which the training simply stops when the "generalization error (i.e. the error on the *development* set) has increased in s successive epochs"; they set s to five. While this stopping criterion is straightforward to define when we train the model on the training split and evaluate it on the development split, it is less clear how such a stopping criterion can be applied when the model is trained on the combination of the training and development splits and evaluated on the test split, as Dima and Hinrichs (2015) did. Upon correspondence with the first author of the aforementioned article, we confirmed that Dima and Hinrichs (2015) in fact used the test split itself in the early stopping criterion to monitor the model's performance. This unfortunate design decision ultimately favors the model that performs best on the test split, and hence violates the constraint on the test split being completely held-out. When evaluating on the test split, we train our model on the training split only and monitor its performance on the development split only (to apply early stopping). Therefore, our results on the test split are not directly comparable with those by Dima and Hinrichs (2015).

With the exception of the optimization function, our model can be considered a 'near-replica' of the model by Dima and Hinrichs (2015). Indeed, this is confirmed by looking at the results in Table 6.4 which are comparable to—and sometimes better than—the results reported by Dima and Hinrichs (2015). They report an accuracy of 77.81% on the development split when using GloVe-300 (cf. Table 6.2), and our model lands at 77.34% accuracy on the same data split and using the same word embedding model. Using the word2vec model, we achieve remarkably better results than what is reported by Dima and Hinrichs (2015). We do not speculate about the reasons behind the improved performance we see, but it is apt to recall here that we use different optimization function from what Dima and Hinrichs (2015) use.[4] All in all, the results in Table 6.4 confirm the validity of our implementation and set the scene for the next round of experiments in which we take a closer look at the model architecture and hyperparameters and word embeddings in light of the classifier's performance on three noun–noun compound datasets: NomBank, PCEDT and Tratz.

6.2 Neural Classification Experiments

We have thus far focused on replicating the model described in Dima and Hinrichs (2015) for noun–noun compound interpretation in the realm of the compounds and semantic relations defined by Tratz (2011). We now turn to studying the use of neural classifiers for our compound datasets (NomBank and PCEDT) as well as the Tratz dataset. More concretely, in the rest of this chapter, we evaluate the effect of different word embedding models and neural network architectures and hyperparameters on the classification performance (in terms of accuracy and macro-averaged F_1 score) on the three compound datasets.

The space of hyperparameters in neural networks is normally quite large; but this space becomes even larger when we consider the hyperparameters related to the input representation (i.e. word embeddings). Therefore, in order to make the results more tractable, and isolate the effect of specific hyperparameters and properties of the neural as well as embedding models, we first conduct a series of experiments on word embeddings in which we

[4] In fact, even if we were using the exact same libraries and optimization functions, seeing the exact same results would have been also unlikely given the non-deterministic nature of parameter initialization in neural networks.

vary only one (hyper)parameter at a time. We follow the same approach when experimenting with new model architectures (§ 6.4) or adding new input features (§ 6.6). This approach implicitly assumes that the hyperparameters and properties we are experimenting with are independent; this, of course, need not be the case, but we accept this simplifying assumption for now to prune the problem space, and eventually be able to control the effect of the different hyperparameters.

Notes on Evaluation: In the following, we will present a rather comprehensive set of experiments which are evaluated and analyzed in terms of accuracy and macro-averaged F_1 score. The accuracy is computed over the full set of evaluation instances (whether in the development or test split in the following chapter). The macro-averaged F_1 scores for PCEDT and NomBank, however, are computed on the subset of 'learnable' relations which are deemed frequent enough in these datasets (the frequency threshold and the relations themselves were defined in § 4.5.4). Furthermore, to ensure that the differences in performance are not a side-effect of the random initialization of some of the neural network weights or the shuffling of training data before each learning epoch, we fix the random seed for the pseudo-random number generators modules in Python and NumPy.

6.3 Effect of Word Embeddings

Before delving into studying the different architectures and hyperparameters of the neural classifier, in the following, we try to determine which types and properties of word embeddings lead to strong end-to-end classification accuracy as well as macro-averaged F_1 score. Dima and Hinrichs (2015) already provide evidence that the word embedding models (used as input representation to the neural classifier) have direct impact on the classification accuracy of the semantic relations in Tratz's dataset. For example, they observe that classification accuracy increases with the dimensionality of the word embeddings and the size of support corpora (i.e. the copora used to learn the word embedding models). The following set of experiments aims to help us further understand the effect of the different properties of word embeddings on the accuracy of neural classifiers, and look for similar—and perhaps new—trends to what Dima and Hinrichs (2015) find, but on other datasets and using a broader and systematically varied range of word embeddings.

In some cases, word embeddings themselves are trained using neural networks (such as word2vec models), which means that the space of hyperparameters is also wide for word embeddings. Therefore, we limit the number of potential experiments by selecting a few parameters (viz. text pre-processing steps, vector dimensionality and training corpora), and change only one of them at a time while fixing all other parameters. For example, when comparing the effect of embedding dimensionality on compound classification we use models that only differ in their dimensionality. Most of the word embedding models used in this section are described in § 5.3.

Unless otherwise stated, the neural architecture we use in the following embedding-related experiments is the same as the one described in the previous section, i.e. our replica of the neural classification model proposed by Dima and Hinrichs (2015).

6.3.1 Text Pre-Processing

We start our experiments by looking at the effect of text pre-processing. Fares et al. (2017) show that simple text pre-processing steps can improve the results when evaluating word embeddings on standard semantic evaluation datasets, such as SimLex-999 (Hill et al., 2015) and the Google Analogies Dataset (Mikolov, Chen, et al., 2013). Indeed, contrary to the common assumption that more data leads to better results, Fares et al. (2017) find that the models that were trained on pre-processed data achieve better results than the models that were trained on larger amounts of *unprocessed* data. Given this conclusion, we use some of the word embedding models introduced in § 5.3 to study the effect of lemmatizing the training corpora of word embeddings on noun–noun compound interpretation. However, using lemma-based word embeddings as input representations to our neural classifier assumes that the noun–noun compound datasets are also lemmatized. Therefore, whenever the lemmatized word embeddings are used, lemmatized versions of the compound datasets are also used. The lemmatized compound datasets are obtained by applying the same lemmatization strategy described in § 5.3 on the original noun–noun compound datasets (i.e. using the *Stanford CoreNLP Toolkit* version 3.6.0).

More concretely, to measure the effect of pre-processing on compound interpretation, we compare the results of using two word embedding models as input to our neural classifier to predict the semantic relations of noun–noun

	Embedding model	Tratz	NomBank	PCEDT
Acc	Lemmas	70.41	78.37	58.04
	Full-forms	72.92	76.63	58.15
F_1	Lemmas	65.58	61.61	39.89
	Full-forms	67.59	54.32	34.50

Table 6.5: Accuracy (top two rows) and macro-averaged F_1 (bottom two rows) on the development split using lemma-based and fullform-based 300-dimensional GloVe word embeddings trained on Wikipedia and Gigaword. The embedding layer was *not* updated during training.

compounds in the NomBank, PCEDT and Tratz datasets. The two word embedding models we use are identical in every aspect except in whether they were trained on lemmas or full-forms. These two models are fully described in Chapter 5, but for the sake of clarity we repeat some of their important properties here. The two models are 300-dimensional and were trained on either lemmas or full-forms from Wiki+Giga (i.e. Wikipedia dump combined with Gigaword Fifth Edition) using GloVe.

The top two rows in Table 6.5 show the accuracy of the neural classifier on the development split of the Tratz dataset as well as our datasets, NomBank and PCEDT, using a lemmatized word embedding model (referred to as 'Lemmas' in the table) and non-lemmatized model (referred to as 'Full-forms' in the table). The last two rows, in the same table, show the macro-averaged F_1 score on the development splits of the three datasets.

As mentioned before, we use lemmatized versions of the datasets in conjunction with the lemmatized word embeddings. In practice this means that we have two versions of each dataset (lemmatized and non-lemmatized), and hence two constituent dictionaries per dataset that are different in size. For example, the size of constituent dictionary in the lemmatized version of the Tratz dataset is $5,070$ compared to $5,242$ in the (original) full-form version. Nonetheless, we can still make general observations about the results in Table 6.5. First, we see a marked drop in accuracy as well as F_1 score on the Tratz dataset when we use lemmas instead of full-forms. One minor, but plausible, explanation for this drop in accuracy is that the way lemmatization was applied on the dataset led to losing significant inflectional information that could help distinguish between some relations in the Tratz dataset. Let's

take for example the compounds *shrine burning*, *flag burning* and *chemical burn*, where the semantic relation of the first two compounds is `OBJECTIVE` and semantic relation of the third one is `SUBJECT`. The lemmatized version of the dataset changes the constituent 'burning' to 'burn', and hence we lose the inflectional suffix which could help distinguish between the first two compounds, on the one hand, and the third one on the other hand.[5] We find that the model operating on lemmas misclassifies the compound *flag burning* (which is lemmatized to 'flag burn') whereas the model operating on full-forms does not. This misclassification error could be blamed on the loss of inflectional information caused by lemmatization which lumps the constituents 'burning' and 'burn' together. Even though a new type of errors is introduced when using lemma-based embeddings and lemmatized compounds, we find that at least 75% of the classification errors made using the full-form embeddings also occur when using the lemma embeddings. Note that our approach to error analysis here is mainly qualitative, as we seek to complement the empirical observations with a linguistically informed explanation of whether or not using lemmas makes sense in the context of noun–noun compound analysis. When it comes to out-of-vocabulary words, we find that even though the lemma-based embeddings (and dataset) have fewer out-of-vocabulary words (viz. 163 words), in contrast to 200 with the full-form embeddings, this does not seem to compensate for the overall drop in accuracy. Another potential explanation for the decreased performance in the lemma-based model has to do with the loss of distinction between different word classes in the lemmatized embedding models. Such loss manifests on the word embedding level when we consider words like 'burn' and 'burning' or 'fly' and 'flying' which end up having one identical semantic representation (i.e. one embedding vector) rather than two separate representations that could have been similar but certainly not identical.

Looking at the NomBank results, we see that using lemmas instead of full-forms does help improve the classification accuracy by 1.74 percentage points. We also see a similar trend in macro-averaged F_1 scores. Nonetheless, the biggest part of the errors made in both setups remains the same; that is, 80% of the errors observed in the lemma-based setup are also seen in the

[5]Whether or not the word 'burning' should be lemmatized to 'burn' depends on the context and its PoS tag. We do not believe 'burning', as a gerund noun, should be lemmatized to 'burn', and therefore the example above seems to be an artifact of lemmatization out of context.

full-form setup. To verify whether or not lemmatization introduces new type of errors (as reported above in the case of the Tratz dataset), we inspect the misclassification errors of the lemma-based setup and find that most of the errors are not directly related to lemmatizing the dataset itself. However, one cannot rule out the effect of lemmatization altogether because the word representations (i.e. the values in the embedding vectors themselves) are affected by lemmatization, as we explained above.

Lastly, the accuracy on the PCEDT dataset remains more or less unchanged, regardless of whether we use lemmas or full forms. However, the macro-averaged F_1 score of the lemma-based model is clearly higher than the full-form one. This 'discrepancy' between the accuracy and macro-averaged F_1 scores stems from the fact that the lemma model outperforms the full-form model on many infrequent relations, sometimes with a large margin such as on the temporal relation `TWHEN`. However, the full-form model moderately outperforms the lemma model in predicting the most frequent relation in PCEDT (`RSTR`). These two observations combined explain why the two models have comparable accuracy scores, but different F_1 scores. Furthermore, we inspect the misclassification errors made by the model on the lemmatized and non-lemmatized versions of the PCEDT dataset. We find that about 76% in either case are the same; from this, one can speculate that the challenge of learning how to predict 76% of the errors lies somewhere else (likewise also for the NomBank and Tratz datasets).

The discussion above indicates that using full-forms instead of lemmas better fits our setup, primarily because lemmatization can only be applied out of context in the case of the Tratz dataset, which arguably leads to some lemmatization errors. Further, full-form tokens preserve the affixes and other forms of inflection that might constitute important features for the classifier as well as the word embedding models (such as the distinction between 'burn' and 'burning' which is lost in the lemmatized tokens). Empirically, however, the results vary depending on the dataset, but it is important to emphasize once again that lemmatizing the noun–noun compound dataset itself introduces changes on the dataset that 'complicate' the comparison with other works (especially so in the case of the Tratz dataset).

D	Tratz		NomBank		PCEDT	
	Static	Fine-tuned	Static	Fine-tuned	Static	Fine-tuned
50	58.19	74.01	75.98	78.15	58.26	**59.67**
100	63.57	75.94	74.89	78.59	57.72	59.24
300	70.41	**76.72**	78.37	**79.35**	58.04	59.24
600	**73.75**	76.25	77.39	79.13	58.59	58.59
1000	72.86	76.30	**78.80**	78.59	**58.91**	58.91

(a) Vector dimensionality versus classification accuracy.

D	Tratz		NomBank		PCEDT	
	Static	Fine-tuned	Static	Fine-tuned	Static	Fine-tuned
50	49.82	65.23	45.60	57.07	35.83	40.23
100	56.18	66.63	46.97	58.15	38.55	39.87
300	65.58	**70.24**	**61.61**	59.45	39.89	42.82
600	**68.84**	68.60	57.22	**60.08**	**40.94**	42.78
1000	66.16	68.99	61.00	53.91	38.81	**42.95**

(b) Vector dimensionality versus macro-averaged F_1 score.

Table 6.6: Vector dimensionality versus (a) accuracy and (b) macro-averaged F_1 on the development split of the three noun–noun compound datasets. Note that all the GloVe embedding models as well as the datasets operate on *lemmas* in the tables above.

6.3.2 Vector Dimensionality and Fine-Tuning

The goal of the next set of experiments is two-fold: First, to study the effect of vector dimensionality on the performance of our noun–noun compound classifier. Second, to examine the benefit of fine-tuning word embedding models (while training the neural classifier) versus the embedding model's dimensionality. Dima and Hinrichs (2015) report that fine-tuning word embeddings leads to more significant improvement in accuracy when the embedding models have relatively low dimensionality. In the following, we use five word embedding models that only differ in their vector dimensionality, but are otherwise identical. The five GloVe models were trained on the lemmatized version of Wikipedia and Gigaword (cf. § 5.3), and hence all the noun–noun compound datasets were also lemmatized.[6]

[6]Our decision to use lemmatized word embeddings stems from partly practical reasons—that is, the availability of lemmatized word embedding models in different dimensionalities.

Table 6.6a shows the classification accuracy of our neural classifier on the development splits of the Tratz, NomBank and PCEDT datasets. Likewise, Table 6.6b shows the macro-averaged F_1 scores. In addition to variable levels of embedding dimensionality, Tables 6.6a and 6.6b show two modes: the *static* mode in which the embedding vectors are not updated during training, and the *fine-tuned* mode in which the embedding vectors are updated (fine-tuned) during training.

Looking at the results in Table 6.6a, we see different trends depending on which dataset and mode one considers. First, in the static mode, it is apparent that higher embedding dimensionality leads to higher accuracy on the Tratz dataset except with the 1000-dimensional embedding, which is somewhat consistent with what Dima and Hinrichs (2015) report.[7] These gains in accuracy, however, become less pronounced when we look at the results on the same dataset in the fine-tuned mode, which is also consistent with the findings of Dima and Hinrichs (2015); that is, fine-tuning helps most for embedding models with comparatively lower dimensions. The macro-averaged F_1 scores on the Tratz dataset are largely in line with the accuracy scores.

We see a somewhat similar trend on the NomBank dataset in both the static and fine-tuned modes. The classification accuracy increases—though less remarkably—with the embedding dimensionality in the static mode; for example, moving from 50-dimensional embedding model to 300-dimensional model leads to more than two percentage points improvement in classification accuracy. However, the improvement becomes much smaller with models of over 300 dimensions. Higher dimensionality also helps in the fine-tuned mode, but moving from one dimensionality level to the next, the effect seems even smaller than in the static mode. The trend observed on the NomBank dataset is similar to what we observe on Tratz above, which is again in line with what Dima and Hinrichs (2015) report. The macro-averaged F_1 scores largely correspond to the accuracy scores in the sense that the models with lower dimensionality (50 and 100) have lower macro-averaged F_1 scores in comparison to the models with higher dimensionality. The accuracy of the 300-dimensional model is slightly lower than that of the 1000-dimensional model but the macro-averaged F_1 score is in favor of the former with a small

[7] Dima and Hinrichs (2015, p. 177) report that "it was always the highest dimensional embedding that gave the best results in the cross-validation setup". We consider our results "somewhat consistent" with theirs because even though the accuracy improves with higher embedding dimensionality, the improvement stops at 300 or 600 dimensions (depending on the dataset).

margin—which is not unexpected given the small difference between the accuracies of these models. In the fine-tuned mode, the 100-dimensional and 1000-dimensional models achieve the same accuracy, but their macro-averaged F_1 scores are not the same or even close. The main reason we see this divergence between accuracy and F_1 score is because the former model (100-dimensional) achieves a much higher F_1 score in predicting the locative relation `ARGM-LOC` in comparison to the latter model (48.00% vs. 12.50%, respectively).

Finally, when it comes to the static mode on PCEDT, we see a different, and somewhat inconsistent, pattern from NomBank and Tratz; using the 50-dimensional model leads to slightly higher accuracy than the 100- and 300-dimensional models. Overall, the highest improvement in accuracy amounts to about one percentage point difference (between the 100-dimensional embedding model and the 1000-dimensional model). In the fine-tuned mode, the difference in accuracy between the best and worst performing models is a little over one percentage point (1.08). Furthermore, we also do not see a clear correlation between the increased embedding dimensionality and improvement in accuracy. The PCEDT macro-averaged F_1 scores in Table 6.6b are also less consistent with their corresponding accuracy scores. For example, the 100- and 300-dimensional embedding models lead to the same accuracy in the fine-tuned mode, but two different macro-averaged F_1 scores, with the latter outperforming the former by 2.95 points, most notably because the 100-dimensional model does not predict (or learn) the functor relation `AIM` at all whereas the 300-dimensional model does.

Given the somewhat inconsistent pattern we see in the PCEDT results, we find it difficult to interpret these results without a closer inspection of the errors made by the neural classifier using the different embedding models. In the fine-tuned mode, around 65.96% of the errors are the same regardless of which embedding dimensionality is in use.[8] Which means that for at least the larger part of the errors, we need to look somewhere else other than the dimensionality of the embeddings, but this still does not explain the pattern (or lack thereof) we see here. Since we have several configurations and many error sets, we inspect the classification errors made by all the fine-tuned models

[8]We write "around 65.96%" because the percentage of common errors per model depends on the number of errors made by each model. We obtain the number 65.96% by taking the average of the percentages on the five error sets; the highest percentage is 66.85% and the lowest is 65.09%.

as well as a few randomly selected examples. One problem we immediately identify is potential annotation inconsistency of some noun–noun compounds in PCEDT (cf. § 4.5.3). All the models mistakenly assign the relation RSTR to the compound *potato farmer* instead of the gold-standard relation REG. However, if we look at the compound instances headed by the noun 'farmer' in the training split, we find that the compounds *banana farmer*, *rice farmer* and *horse farmer* are annotated as RSTR, while the compounds *wheat farmer* and *catfish farmer* are annotated as REG.[9] We do not immediately see why the compounds *rice farmer* and *wheat farmer* were annotated with two distinct relations, and thus suspect that the 'distinction' is a result of annotation mistake (or inconsistency). The potential inconsistency in these training examples makes the learnability of some PCEDT relations difficult, if not impossible in some cases. We present more examples of potential annotation inconsistency in Table 6.7, where we list all the compound examples that are headed by the noun 'talks' and their annotation in the training split. Further, in Table 6.8 we present the compound examples that are headed by the noun 'talks' in the development split, their gold standard annotations and the predictions of the different classifiers. We do not aim to discuss the annotation decisions here, rather point out the reason why some compound examples in the PCEDT dataset are misclassified (or at the very least hard to classify). In Table 6.8, there are two examples that are predicted correctly by all the models and these are annotated with the functor RSTR. Otherwise, we see that all the models confuse the functor relations REG, RSTR and PAT, which is arguably a result of how the training data annotate highly similar compounds with different semantic relations (e.g. *pay talks*, *wage talks* and *price talks* are annotated with three different relations).

While we cannot establish, with full certainly, a direct link between the pattern we see in Table 6.6a and the potential annotation inconsistency, the latter alone leads to put slightly less weight on the PCEDT classifier's performance when deciding on which word embedding model to use for the three datasets. Hence, unless otherwise stated, for the majority of the experiments presented in this chapter and the following one, we will use an embedding model of 300 dimensions because it leads to the best combination of accuracy and macro-averaged F_1 score on the NomBank and Tratz datasets in the fine-tuned mode.

[9]The PCEDT manual defines REG as an "adjunct expressing a circumstance that the main predication takes into account", cf. § B.2.

Compound	Functor	Compound	Functor
arms-control talks	RSTR	takeover talks	REG
bank talks	RSTR	peace talks	REG
strategy talks	RSTR	steel talks	PAT
bid talks	RSTR	quota talks	PAT
contract talks	RSTR	studio talks	ACT
pay talks	RSTR		

Table 6.7: Example compounds headed by the noun *talks* in the training split of the PCEDT dataset.

Compound	Gold	50	100	300	600	1000
labor talks	REG	RSTR	RSTR	RSTR	REG	REG
wage talks	REG	RSTR	RSTR	REG	RSTR	REG
price talk	PAT	REG	RSTR	RSTR	RSTR	REG
merger talks	PAT	RSTR	RSTR	RSTR	RSTR	RSTR
disarmament talks	RSTR	RSTR	RSTR	RSTR	RSTR	RSTR
emergency talks	RSTR	RSTR	RSTR	RSTR	RSTR	RSTR
settlement talks	RSTR	PAT	RSTR	RSTR	RSTR	RSTR

Table 6.8: Example compounds headed by the noun *talks* in the development split of the PCEDT dataset and the predicted functor using the five embedding models.

6.3.3 Size of Training Data and Fine-Tuning

In the following set of experiments, we study the effect of increasing the size of the corpora used to train the word embedding models on the performance of our neural classifier. Here, we also experiment with two modes, 'static' and 'fine-tuned', to better understand the relationship between fine-tuning word embeddings (as part of training the neural classifier) and the size of their training data. We compare four word embedding models by using them as input to our neural classifier, and hence our comparison is again based on the final accuracy and macro-averaged F_1 scores of the neural model on the three compound datasets we have used so far: Tratz, NomBank and PCEDT. All the word embedding models are trained on full-form tokens (i.e. non-lemmatized data) using GloVe and have a vector dimensionality of 300. The first three models in Table 6.9a were introduced in Chapter 5 and the fourth model, Common Crawl (42B), was trained by Pennington et al. (2014).[10]

The results of training the neural classifier on the training split and evaluating it on the development split of each of the three datasets are presented in Table 6.9a in terms of accuracies and Table 6.9b in terms of macro-averaged F_1 scores.

In the static mode, the classification accuracy on the Tratz dataset does not always improve when using embedding models trained on larger amounts of training text, e.g. the Wikipedia model, on the one hand, vs. Gigaword and Wiki+Giga models, on the other hand. Furthermore, the difference in accuracy between the model trained on the smallest corpus (Wikipedia) and the one trained on the largest corpus (Common Crawl) is less than one percentage point (0.89). However, the 'coverage' of the word embedding models certainly improve with more training text; in the Wikipedia model, for example, there are 230 out-of-vocabulary constituents in the Tratz dataset, but the number goes down to 3 with the Common Crawl model. Upon inspecting the errors, however, we find that only six compounds (about 1%), of all the errors made by the Wikipedia model, include out-of-vocabulary words in either the head or modifier positions. Five of these six compounds are also part of the errors made by the Common Crawl model even though—

[10] In fact, the GloVe model trained on Common Crawl differs from the other three models in other aspects and not just the size of the training data, including the initial learning rate and the window size where we used a window of size five whereas Pennington et al. (2014) used a window of size ten.

Corpora	Tratz		NomBank		PCEDT	
	Static	Fine-tuned	Static	Fine-tuned	Static	Fine-tuned
Wikipedia (2B)	73.54	77.08	**77.50**	78.26	57.72	59.24
Gigaword (4.8B)	73.13	76.46	76.09	77.28	58.26	**59.46**
Wiki+Giga (6.8B)	72.92	77.45	76.63	78.04	58.15	58.80
Common Crawl (42B)	**74.43**	**77.50**	75.43	**78.37**	**58.91**	58.59

(a) Embedding corpora size versus classification accuracy.

Corpora	Tratz		NomBank		PCEDT	
	Static	Fine-tuned	Static	Fine-tuned	Static	Fine-tuned
Wikipedia (2B)	**68.39**	69.21	**58.33**	59.81	39.19	**43.60**
Gigaword (4.8B)	67.63	68.11	50.95	49.11	32.73	36.93
Wiki+Giga (6.8B)	66.92	**69.89**	57.07	58.92	34.50	37.10
Common Crawl (42B)	67.80	69.26	53.32	**59.81**	**43.57**	37.46

(b) Embedding corpora size versus macro-averaged F_1 score.

Table 6.9: Embedding corpora size versus (a) accuracy and (b) macro-averaged F_1 score on the development split of the three compound datasets. Note that all the embedding models and datasets are not lemmatized in the tables above.

except for one—their constituents are no longer out-of-vocabulary. Since the small improvement in classification accuracy is not directly related to the coverage of the embedding model, it might be the case that with significantly more training data (Wikipedia vs. Common Crawl) we achieve more reliable semantic representation, at least for predicting the semantic relations defined by Tratz (2011). As reported by Dima and Hinrichs (2015), the size of the embedding training data plays a less important role when the word embeddings are fine-tuned as can be seen in Table 6.9a for the three datasets, not just Tratz. Interestingly, we find that only 50% of the misclassification errors in the static mode are the same across the different models. This percentage of common errors, however, grows to 75% in the fine-tuned mode. Considering the macro-averaged F_1 scores, we find that the Wikipedia model achieves better performance than the Common Crawl model (in the static mode); even though there are many differences in the per-relation F_1 scores of the two models, the most noteworthy one is the relation `JUSTIFICATION` which the Common Crawl model never predicts but the Wikipedia model does. This difference is not clearly reflected in the accuracy scores because the aforementioned relation ranks among the least frequent relations in the

Tratz dataset.

When it comes to NomBank in the static mode, we see that the Wikipedia model yields better accuracy (and macro-averaged F_1 score), in comparison to the Common Crawl model which was trained on 20 times more data. In fact, the Common Crawl model leads to the worst accuracy (and second worst macro-averaged F_1) on NomBank in comparison to the other models in Table 6.9a. In the fine-tuned mode, however, all the embedding models achieve relatively similar accuracy scores with the largest difference being less than one percentage point (i.e. the Gigaword model vs. the Common Crawl model). About 57% of the errors in the static mode and 60% of the errors in the fine-tuned mode are the same across all the models.

For PCEDT, in the static mode, the classification accuracy seems to improve when using an embedding model trained on significantly larger corpora (i.e. the Wikipedia model vs. Common Crawl). This observation, however, is not applicable in the fine-tuned mode, where the best accuracies are in fact achieved with the embedding models trained on smaller corpora (Wikipedia and Gigaword models). In terms of F_1 scores, the model that leads to the best accuracy in the fine-tuned mode (Gigaword) leads to the worst macro-averaged F_1 score. Overall, the Gigaword model achieves lower per-relation F_1 scores in comparison to the Wikipedia model except on `RSTR` and `ACT` (the former being the most frequent relation in PCEDT). For completeness, we inspect if the models overall make different types of errors by intersecting their misclassification errors. We find that about 75% of the PCEDT errors by all the models are the same in both the static and fine-tuned modes.

As mentioned before, the Common Crawl model actually differs from other embedding models in more than just the size of training data, and therefore we exclude it from the final comparison to choose an embedding model for the three datasets. Among the three other embedding models, the Gigaword model leads to the worst accuracies and macro-averaged F_1 scores on NomBank and Tratz. The Wikipedia and Wiki+Giga models lead to comparable accuracies on NomBank and Tratz in the fine-tuned mode, with a slight preference for the former on NomBank and for the latter on Tratz. However, given that the difference between the accuracy of Wikipedia and Wiki+Giga models on NomBank is smaller than that on Tratz, we take the final decision based on the model's performance on the Tratz dataset. The Wiki+Giga model yields the highest accuracy on Tratz, and hence we will

use it in the remaining experiments in this chapter and the following one.[11]

Finally, even though we do not see a monotonic relationship between the size of the training corpora and performance in our experiments, these results alone are not enough to conclude that no such relation exists. The corpus size is not the only factor at play here; the training text domain or genre vary across the corpora we used to train the embedding models.

Which Embedding to Use? Based on the experiments we presented so far, we can conclude that different embedding models—i.e. word embedding properties—are preferable to the three compound datasets. In other words, there is no cross-cutting model that fits all scenarios, which is consistent with what Schnabel et al. (2015) report: "different tasks favor different embeddings". There are, however, a few observations that generally hold across the different configurations and datasets. For example, we can safely assume that increasing the embedding dimensionality to 600 or 1000 dimensions become less important when the embedding models are fine-tuned as part of the neural classifier. Furthermore, even though it might seem practically appealing to use lemma-based embeddings for some datasets (especially PCEDT), our manual error analysis reveals that lemmatization will inescapably lead to loss of potentially important features. Overall, it is empirically clear that fine-tuning the embedding models leads to better classification accuracy and macro-averaged F_1 score. In addition, using word embeddings trained on substantially larger copora does not necessarily lead to better performance in terms of accuracy and macro-averaged F_1 scores. To ensure that the results of the following experiments are somewhat comparable—feasible indeed—we pick one embedding model as we investigate the impact of different neural architectures and hyperparameters on compound interpretation. For the rest of this chapter, unless stated otherwise, we will use the 300-dimensional embedding model trained using GloVe on non-lemmatized text from Wikipedia and Gigaword (Wiki+Giga).

[11] We relied on NomBank and Tratz to make the final decision because the results of experiments so far have been more consistent on those two datasets than PCEDT. However, if we are to consider the accuracy and macro-averaged F_1 scores on PCEDT, then the Wikipedia model would be the best one for the majority of datasets (i.e. for NomBank and PCEDT).

6.4 Model Architecture

In this section we move to experimenting with the architecture of the neural classification model. The architectures we experiment with are motivated by the problem at hand, that is, noun–noun compound interpretation. More specifically, we introduce head- and modifier-specific layers (constituent-specific layers) to the classification model; for example, instead of having just one embedding layer for both the head and modifier nouns we experiment with two separate embedding layers. The rationale behind using modifier- and head-specific layers is based on the assumption that the head and modifier nouns can play different roles in defining the semantic relation of noun–noun compounds. Kim and Baldwin (2005) find the head noun to be "a more reliable predictor" of compound interpretation than the modifier noun. However, they also report that the best overall accuracy is achieved when both nouns contribute equally to their interpretation algorithm. The neural models we introduce in this section somehow capture the same idea of Kim and Baldwin (2005), but instead of manually setting the weights of the head and modifier nouns' contribution as in Kim and Baldwin (2005), we rely on the neural model to learn different weights for the head and modifier nouns.

Based on the architecture of the classification model we have been using so far, we identify three alternative architectures which include constituent-specific layers, namely:

1. Embedding-specific (E_S): the model has two embedding layers, one for the head noun and one for the modifier noun. This, in practice, means that we end up with two sets of word embedding vectors, where the embedding vector of the same word could be different depending on whether it occurs in the head or modifier position (as a result of fine-tuning the word embedding models during training).

2. Hidden-specific (H_S): the model has two hidden layers but only one embedding layer. The output of the hidden layers are concatenated in a so-called merge layer and passed to the output layer. The architecture of this model is shown in Figure 6.2.

3. Embedding- and hidden-specific (EH_S): the model has two embedding layers and two hidden dense layers. In other words, the representations and weights of the head and modifier nouns are separated throughout

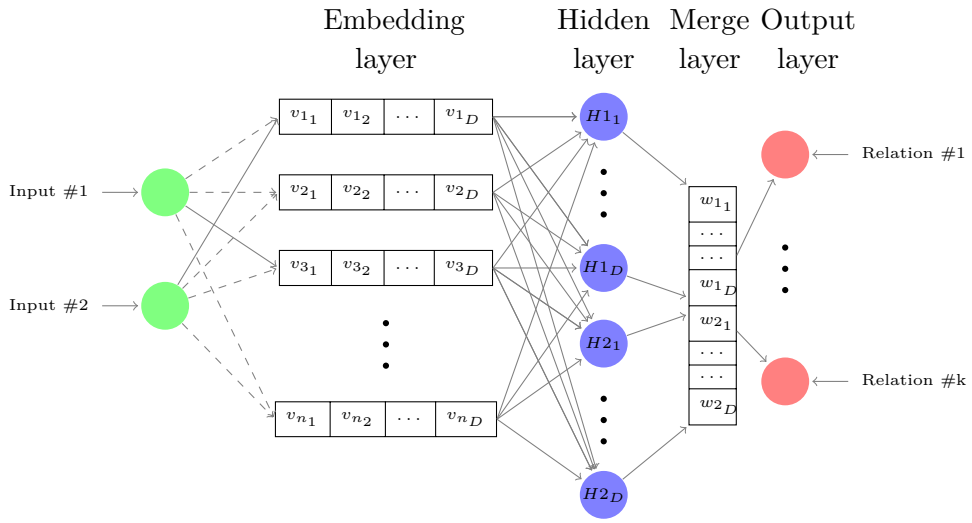


Figure 6.2: Architecture of the neural classifier with constituent-specific hidden layers (i.e. two hidden layers). The solid lines in the figure indicate how the information flows throughout the network for a given compound, i.e. vector embedding look-up, then feeding the selected vectors into the dense layer. Note that the merge layer is fully connected with the output layer but the figure is simplified.

the model except on the output layer where the output of the hidden layers are concatenated using a merge layer (like in the H_S model).

We use the exact same data splits and hyperparameters described in the previous section to train and evaluate the three constituent-specific models. Table 6.10a shows the accuracy scores of the three models on the development splits of the Tratz, NomBank and PCEDT datasets. Likewise, Table 6.10b shows the macro-averaged F_1 scores across the three models and datasets.

As evident from Table 6.10a, the three constituent-specific models do not improve over the accuracy of the model with no constituent-specific layers (referred to as $None_S$ in Table 6.10a) for the Tratz dataset. However, the macro-averaged F_1 scores in Table 6.10b draw the opposite picture where all the constituent-specific models outperform the model with no constituent-specific layers, albeit with a small margin. This is mainly due to the $None_S$ model failing to predict one specific relation (`PARTIAL_ATTRIBUTE_TRANSFER`) which the three constituent-specific models predict. Not predicting this relation at all does not have a remarkable influence on the accuracy score because it is one of the least frequent relations in the Tratz dataset; in fact,

Model	Tratz	NomBank	PCEDT
E_S	76.46	77.93	59.02
H_S	76.09	77.28	**59.46**
EH_S	75.99	77.28	58.37
$None_S$	**77.45**	**78.04**	58.80

(a) Accuracy on the constituent-specific models on the development set.

Model	Tratz	NomBank	PCEDT
E_S	**71.63**	50.13	**37.86**
H_S	70.51	56.38	36.95
EH_S	70.08	58.09	36.61
$None_S$	69.89	**58.92**	37.10

(b) Macro average F_1 of the constituent-specific models on the development set.

Table 6.10: Accuracy (a) and macro-averaged F_1 (b) of the constituent-specific models on the development set. Embedding: Wiki+Giga. Full form. Adam. Batch: 5. Fine-tuned. The last line is the model with no constituent-specific layer.

it constitutes only 0.3% of the dataset (Tratz, 2011). The so-called Personal relations in Tratz (`PERSONAL_NAME` and `PERSONAL_TITLE`) also benefit from the constituent-specific setups, sometimes leading to 100% F_1 score. However, these two relations also fall down the list in terms of frequency, each making up only 0.5% of the dataset. From these results, we can conclude that if accuracy is the most important metric in learning the Tratz dataset, then no gains can be achieved by using constituent-specific layers. However, a midway compromise between accuracy and macro-averaged F_1 would favor the constituent-specific model on the embedding layer E_S.

On NomBank, we do not see any improvement in accuracy using the constituent-specific models. The macro-averaged F_1 scores in Table 6.10b, however, point to stark variation between the models that are otherwise invisible if we only consider the accuracy scores. For example, the $None_S$ model outperforms the embedding-specific model (E_S) by a fraction of 0.11 points in terms of accuracy, but the difference goes up to 8.79 points in terms of macro-averaged F_1 score. This remarkable contrast between accuracy and F_1 can be blamed on the E_S model performance on two specific relations:

ARGM-MNR and ARGM-LOC. The E_S model does not predict the former at all (while all other models do) and achieves a very low F_1 score on the latter (compared to the other models). The model's F_1 score is also lower on other relations, but it achieves a comparable F_1 score to the $None_S$ model on the most frequent relation, which is why these differences are not clearly reflected in terms of accuracy.

We observe small improvements in accuracy on PCEDT (less than 1%) using embedding- and hidden-specific layers (i.e. E_S and H_S). The macro-averaged F_1 scores, though not fully in line with the accuracy scores, do not show high divergence from the accuracy results (unlike the results for NomBank). We remain unsure, nonetheless, about the final conclusion regarding PCEDT, and NomBank to some extent. Hence in order to judge which model is in fact better than the rest we take a closer look at their actual predictions.

For these two datasets, we compare the misclassification errors made by the different models to determine if there is any change in the error patterns when the head and modifier nouns are (partly) isolated. We find that 89% of the PCEDT classification errors made by all models (including the model with no constituent-specific layers) are the same, and 60% of the errors on NomBank are the same (also including all model architectures). This indicates a potential change in the type of errors made by the different models on NomBank (much less so on PCEDT); i.e. some architectures might be better (or worse) at predicting certain relations than others.

Upon closer inspection of the output of the different models, we observe a small change in the error patterns when predicting the NomBank locative and temporal relations (ARGM-LOC and ARGM-TMP, respectively). All the constituent-specific models make fewer classification errors on the ARGM-TMP relation, but only the EH_S model makes fewer errors on ARGM-LOC. However, with the exception of these two relations, the $None_S$ model (with no constituent-specific layers) still performs best at predicting all other relations. Most notably, the model without any constituent-specific layers outperforms all the constituent-specific models in predicting the 'core' arguments in NomBank (ARG0, ARG1, ARG2 and ARG3). These observations indicate that introducing constituent-specific layers can help when there are prototypical heads or modifiers that occur with a given relation, which is obviously the case for ARGM-TMP, but not for the aforementioned core arguments.

On PCEDT, the prediction of some functors such as the patient functor

PAT also improves when we introduce constituent-specific layers. However, we do not see a consistent behavior across the different constituent-specific models.

Finally, the observation that *some* relations can benefit from introducing separate layers for the head and modifier nouns is somewhat in line with what Kim and Baldwin (2005) report (albeit in a pre-neural classification approach); that is, the impact of the head and modifier noun varies depending on the semantic relation. Our experimental results and error analysis indicate that the relations that might benefit most from constituent-specific layers are the ones that are likely to have 'prototypical' modifiers or heads, such as the temporal relations. Such relations, however, are among the least frequent ones in the three datasets we are studying. Thus, in the grand scheme of things, introducing constituent-specific layers to the classification model offers little to no improvement in *accuracy*.

6.5 Model Hyperparameters

In all the experiments presented in this chapter so far, we relied on the findings reported by Dima and Hinrichs (2015) to set the hyperparameters of our models. Even though our focus has been to isolate the effect of different aspects of our model (such as the input representation § 6.3 and the model architecture § 6.4), we believe it is equally important to gauge whether the choice of hyperparameters by Dima and Hinrichs (2015) also leads to (near-)optimal results on our datasets.

Hyperparameter optimization can be a major challenge in neural networks as the process can easily become intractable with too many hyperparameters and settings to experiment with. More importantly, it is computationally expensive to perform an exhaustive grid search on all possible combinations of hyperparameter values, because of the exponential growth of the combined search space.[12] In this section, we approach the question of hyperparameter optimization using two strategies. However, before we introduce these strategies, it is important to highlight that our primary aim here is to determine the plausibility of the combination of hyperparameters and settings we use in terms of the model's classification accuracy. We acknowledge that

[12]If we have m hyperparameters each taking n values, then the grid search space (i.e. number of combinations) grows as $O(n^m)$ (Goodfellow et al., 2016, p. 429).

additional gains maybe could be achieved through even more comprehensive hyperparameter optimization experiments than what we present in this section, but this would of course require more computational time and analysis. We therefore deliberately choose to focus more on the methodological and representation questions related to the problem at hand, i.e. noun–noun compound interpretation, at the expense of more hyperparameter optimization experiments.

In the following sections, we start by exploring the sensitivity of the model's performance with respect to a selection of hyperparameters in isolation (§ 6.5.1), in a manner somewhat similar to how Zhang and Wallace (2017) study the sensitivity of convolutional neural networks (CNNs) for sentence classification. The sensitivity analysis allows us to gauge the impact of individual hyperparameters on our particular task, but it also makes strong assumptions about hyperparameter independence. Therefore, we also conduct a series of hyperparameter optimization experiments on the three compound datasets (§ 6.5.2), where the correspondence or dependence relation among the hyperparameters is taken into account. We use random search for hyperparameter optimization, which was shown to be more efficient than grid search (with exponential growth) and manual search (Bergstra & Bengio, 2012).

6.5.1 Sensitivity Analysis

Zhang and Wallace (2017) present a sensitivity analysis with respect to the parameterization of a CNN model on nine datasets for sentence classification. The goal of their study is to identify the sensible value ranges to explore as well as the important settings or hyperparameters in contrast to those that have little or no effect on the model's performance independent of dataset. In their experimental setup, Zhang and Wallace (2017) study several hyperparameters and settings, such as dropout and word embedding models, in isolation; that is, they change one hyperparameter at a time. We follow a similar approach in this section, by staging our hyperparameter search through several consecutive experiments. While we also aim to determine the sensitivity of our classification model to a selection of hyperparameters, our experiments are limited to three hyperparameters only: optimization

function, batch size and dropout.[13] Since we continue to fix the random seed, we run the experiments in this section one time only, unlike Zhang and Wallace (2017) who run each of their experiments ten times and report the mean accuracy in addition to ranges of maximum and minimum scores. In order to isolate the effect of these hyperparameters as much as possible, we do not change the model architecture throughout the experiments. Finally, we do not detail the results of the individual experiments in tables, but instead we include plots of the results in Appendix C.

Optimization Function: The first hyperparameter we vary is the optimization function. We experiment with almost all the available optimizers in *Keras* using their default learning rates, namely: AdaMax, RMSprop, Adam, Adadelta, AdaGrad and SGD. In Figures C.1, C.2 and C.3, we plot the validation accuracy and loss when training our neural classification models on PCEDT, NomBank and Tratz, respectively, using the aforementioned optimization functions. Note that these plots correspond to training for 15 epochs without updating the embedding layer (i.e. training in the static mode)—this is mainly due to the computational cost of running all the experiments in the fine-tuned mode.

Overall, the validation accuracy and loss when training on the Tratz dataset look more smooth than on NomBank and PCEDT; training on the latter two datasets seems to converge much earlier than on Tratz, and hence the validation accuracy starts to fluctuate after several epochs. Further, none of the optimizers lead to distinctly better validation accuracy on any of the datasets (in comparison to the other optimizers), but their overall behavior differs—expectedly so. For example, the traditional Stochastic Gradient Descent (SGD) takes much longer to converge compared to the other optimization functions which have some sort of adaptive learning rate. RMSprop, in all three datasets, might be overfitting the training data without harming the validation accuracy after five epochs (because we see a significant peak in the validation loss while the validation accuracy remains more or less the same).

As mentioned before, all the optimization functions arrive at more or less the same validation accuracy except SGD and AdaGrad. The latter accumulates the square of all gradients (i.e. always positive-valued numbers),

[13] Our word embedding experiments, in § 6.3, are also similar in spirit to the work by Zhang and Wallace (2017).

which means that it monotonically shrinks the learning rate throughout training, and therefore the learning slows down early on (Zeiler, 2012). This property of SGD and AdaGrad is at odds with our experimental design decision to (pre-)set the same number of epochs for all optimizers. We compensate for this unfair setup in § 6.5.2 where we set the number of epochs to a higher value and use an early stopping criterion. For the sake of this section, however, the main goal is not to determine the optimizer that leads to the best validation accuracy, but rather examine the overall behavior of the classifier when different optimizers are used and look out for potential anomalies. Finally, we believe there are more interesting observations to be made about the different optimization functions, but this falls outside the scope of our project as we aim to invest more in finding out which features, model architectures and learning strategies improve the classification accuracy, in contrast to extensive experimentation with and analysis of the model's hyperparameters.

Batch Size: The second hyperparameter we experiment with is the batch size. Thus far, we have been using a mini-batch size of 5 in our of all experiments, which is partly motivated by the work of Dima and Hinrichs (2015). Here, we experiment with nine different values for the batch size: 1, 5, 10, 20, 50, 100, 150, 200 and 250. We train and plot the validation accuracy and loss of the model on the three datasets using the AdaGrad optimization function over 15 epochs. We do not update the weights of the embedding model in these experiments for the same reason as before (i.e. computational cost). The plots are shown in Figures C.4, C.5 and C.6 in Appendix C.

From these plots, we see that using larger batch size means the model needs more epochs to converge on the three datasets. Relatively small mini-batches (such as 5 or 10) converge in the early epochs on the three datasets. In terms of validation accuracy, the best accuracies are achieved with smaller batches. On the one hand, these are partly a by-product of setting the number of epochs to 15 for all batch sizes, which might be 'unfair' towards the larger batches which need more epochs to converge. On the other hand, it has been shown that large batches can indeed degrade the model's ability to generalize (Keskar et al., 2017).

Even though these experiments are not enough to provide evidence of whether large batches degrade our model's performance given a larger number of epochs, we use the overall observation—that smaller mini-batches are

likely to yield better results—to guide our random search experiments in the following section § 6.5.2.

Dropout: We now turn to experimenting with adding a dropout layer to our neural classification model. We add the dropout layer after the hidden layer which sets a fraction of the hidden units to zero; this fraction is specified by a parameter called the dropout rate. We experiment with four values for the dropout rate: 0, 0.25, 0.50 and 0.75. A zero-valued dropout rate is equivalent to no dropout. We plot the validation accuracy for the neural model on the three datasets using the different dropout rates in Figures C.7, C.8 and C.9.

Overall, we find that adding a dropout layer might be of potential help to improve the generalization of the model. However, similar to what Zhang and Wallace (2017) find, the 'ideal' dropout rate seems to be dependent on the dataset; for example, the NomBank model performs better with a low dropout rate (0.25), whereas the PCEDT model achieves the best validation accuracy using the highest dropout rate (0.75), and on the Tratz dataset a dropout rate of 0.5 leads to the best accuracy. Even though we do not rule out the benefit of adding a dropout layer, we do not use it in our overall experimentation scheme because it requires a dataset-dependent dropout rate, while we aim to have a unified setup for the three datasets. Therefore, none of the experiments in the following chapter use a dropout layer, unless otherwise stated.

6.5.2 Random Search

The experiments introduced in the previous section obviously make strong assumptions about the independence of the different hyperparameters. Such assumptions do not hold in practice, but they are acceptable in the context of exploring the model's sensitivity to specific hyperparameters. To optimize the model's hyperparameters, however, we need to experiment with different combinations of these hyperparameters. Therefore, in this section we present the results of the random search experiments we conduct to identify the best combination of hyperparameters for the three datasets.[14] To keep the problem tractable, we restrict our experiments to seven hyperparameters and settings:

[14]We use a Python library called *Talos* to perform the hyperparameter optimization random search experiments `https://github.com/autonomio/talos`

Hyperparameter/Settings	Values
Hidden layer size	4, 8, 16, 32, 64, 128, 256, 300
Hidden layer activation	Sigmoid, Hard Sigmoid, ReLU, SELU, ELU
Optimizer	Adam, Adamax, Adadelta, Adagrad, SGD, Nadam
Batch size	2, 4, 5, 8, 16, 32
Epochs	20, 40, 50
Dropout rate	0, 0.2, 0.4, 0.6
Trainable embeddings	True, False

Table 6.11: Hyperparameter values and settings for the random search experiments. Note that we also use an early stopping criterion in all the experiments regardless of the number of epochs.

(1) optimization function, (2) activation function, (3) dropout rate, (4) batch size, (5) number of epochs, (6) size of the dense layer and (7) whether or not the word embedding layer is trainable. The full set of hyperparameter values and settings are listed in Table 6.11. We choose the values of some of these seven hyperparameters and settings based on what we observed in the previous section; for example, we limit the mini-batch size to a maximum of 32 (cf. discussion in § 6.5.1). However, we also choose some 'extreme' values to measure the effect on specific settings; for example, we expect the size of the hidden layer to play a major role in the overall performance, so we select values ranging from just four units to 300. Overall, we end up with a total of $34,560$ hyperparameter combinations for each of the three datasets, which would translate to $103,680$ experiments if we were to perform a pure grid search for the three datasets. This is obviously prohibitively expensive in terms of computational cost, so we downsample the grid size by randomly selecting 0.5% of the hyperparameter combinations using a quantum random number generator.[15]

Before we delve into the results of the different hyperparameter combinations on the three datasets, we observe two general patterns that are dataset-independent (in the context of our three datasets). First, fine-tuning the word embeddings helps improve the model's performance across the three datasets, but more importantly, across the different values of hyperparameters. This pattern can be easily seen in Figure 6.3, where the experiments in which

[15]The Australian National University Quantum Random Number Server `https://qrng.anu.edu.au`

the word embedding models are fine-tuned achieve higher validation accuracy than the experiments with static embeddings. Second, using an extremely small number of units in the hidden layer (such as four or eight) degrades the performance of the model substantially, regardless of whether or not the embedding layer is fine-tuned. In Appendix C, we include the figures corresponding to model performance with different levels of granularity of the hidden layer size. Note, however, that we cannot read too much into these plots because there are many other variants across all the experiments; in other words, the plots presented in Figures C.10, C.11 and C.12 oversimplify the dependency relations to other hyperparameters by just indicating the hidden layer size and 'trainability' of the embedding layer.

Our random search experiments reveal that the top performing hyperparameter combinations on the three datasets differ among themselves and from the configuration we used in the previous sections. One way forward would be to simply select the set of hyperparameter values and settings that lead to the best validation accuracy on the three datasets *separately*. While this approach has its merits, we choose to continue to limit the variation in hyperparameters across the three datasets for reasons that will become clear in the following chapter, but for now it suffices to say that our experiments focus more on the benefits of different learning architectures and features. It therefore makes sense for us to try and limit the variation in other variables (i.e. hyperparameters) as much as possible. Nonetheless, it is still important to verify that the combination of hyperparameters and settings we used so far lead to a reasonably high performance across the three datasets; we will refer to the configuration of hyperparameters and settings used so far (cf. Table 6.3) as the "unified hyperparameter configuration" or, for simplicity, the "unified configuration".

Given the list of hyperparameter combinations explored in our random search experiments, we rank it based on the model's validation accuracy and check where would the unified configuration rank in this list. We do this comparison for the three datasets. We find that the unified configuration would be the seventh best performing system on Tratz, eleventh best on NomBank and twenty-second best on PCEDT. Half of the hyperparameter combinations that lead to better validation accuracy on Tratz use the same optimizer as the one we used in our experiments (i.e. Adam), but they differ in terms of the dropout rate. Similarly, 30% of the hyperparameter combinations that lead to better validation accuracy on NomBank also use

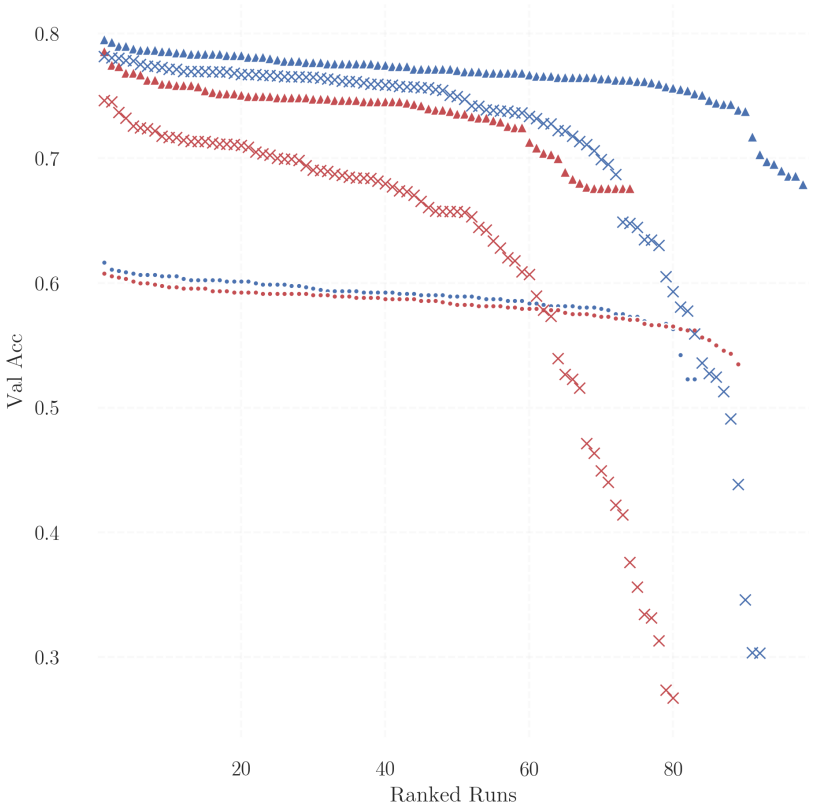


Figure 6.3: Validation accuracy on Tratz (▲), NomBank (×) and PCEDT (●) over 172 hyperparameter random search experiments for each dataset. The figure shows the accuracy per experiment per dataset ranked by decreasing accuracy (on the x-axis). The red markers correspond to the static embedding setting and the blue ones to the fine-tuned embedding settings. Since we draw these two settings separately, the x-axis does not span to 172 points and stops at 98, which is the largest number of experiments using one of the two settings (the fine-tuned setting on Tratz).

Adam as optimizer with variations in the dropout rate and batch size. On PCEDT, 20% of the hyperparameter combinations outperforming ours use Adam with some variations in other hyperparameters as well. Overall, we find the results on Tratz and NomBank reassuring, especially that we have already observed that dropout can help (cf. § 6.5.1). The results on PCEDT are arguably less assuring, but as will be shown soon (in § 6.4) PCEDT in fact exhibits the highest level of sensitivity to random initialization which adds yet another variable to the results of the random search experiments on PCEDT.

Final Words on Hyperparameter Optimization: Even though we remain assured that our choice of hyperparameters yields robust and high-performing results, we do observe that some hyperparameter combinations may lead to better performance than what has been reported earlier in this chapter. We will, however, continue to use the same unified hyperparameter configuration we used so far to allow comparability across the different experiments in the following chapter. We also acknowledge that there are other hyperparameters that could have been explored, such as the learning rate, in addition to further analysis of the interplay between the different hyperparameters.

6.6 WordNet Features

Our experimental results in § 6.3 and § 6.4 as well as the previous section indicate that improving the classification accuracy potentially requires adding more input 'features' to the neural classification model, rather than just more intensive fine-tuning of the neural network's hyperparameters. In this section, we briefly present our experiments with new features based on information from classical lexical resources such as WordNet (Miller, 1995).

Tratz and Hovy (2010) use several lexical resources to encode features about noun–noun compounds in their MaxEnt model, which still achieves state-of-the-art cross-validation accuracy on their dataset. One of these lexical resources is WordNet. Inspired by their choice of features, we run a few pilot experiments using similar WordNet features as additional input to our neural classifier, aiming to provide a tentative indication of the utility of this type of information in our neural classification architecture.

There are many different types of features one can extract from WordNet, but the most straightforward choice is to use the set of all synonyms in the so-called *synsets* (synonym sets) of a given compound's constituent. However, since the *synsets* in WordNet sometimes include synonyms of different word classes (e.g. verbs, adjectives and adverbs), we distinguish between three design decisions with respect to the part-of-speech (PoS) tags of the words in the *synset* we use:

1. N: Nouns only
2. N+V: Nouns and verbs (as Tratz and Hovy (2010) did)
3. E: Everything (i.e. all the synonyms regardless of their PoS tag)

For each constituent (noun) in the constituent dictionary, we extract its synonym set from WordNet following the three PoS options enumerated above. Once the synonyms are extracted, the next step is to decide how to represent them; the most straightforward representation in our setup is to use the average vector of the embedding vectors of all words in the synset. This average vector (the average *synset* vector, henceforth) can then be used as input to the neural network, and here—again—we also distinguish between at least two approaches: the 'concatenated' approach and the 'averaged' approach. In the 'concatenated' approach, we concatenate the compound's constituent embedding vector and the average *synset* vector (leading to a 600-dimensional input vector per constituent, if the embedding model is 300-dimensional). In the 'averaged' approach, we take the average of the constituent's embedding vector and the average *synset* vector (leading to a 300-dimensional input vector per constituent, if the embedding model is 300-dimensional). Finally, if a constituent does not have a *synset* in WordNet or if some of the words in the constituent's *synset* do not have embedding vectors (i.e. are out-of-vocabulary), we then use the vector of the unknown word.

Table 6.12a shows the accuracy results of training and evaluating the six WordNet models using three PoS tag sets for the words allowed in the synsets and two ways to input the averaged vector of the synonyms to the neural classifier (i.e. averaged and concatenated). Table 6.12b shows the macro-averaged F_1 scores for the same setups. Overall, we see that using WordNet-based features does not lead to substantial improvements over the

Synset	Concatenated			Averaged		
	Tratz	NomBank	PCEDT	Tratz	NomBank	PCEDT
N	77.29	78.48	59.46	**77.55**	**78.80**	58.59
N+V	76.82	78.70	**59.78**	77.29	78.48	58.80
E	76.98	**78.80**	59.67	77.45	78.69	**58.91**
No-WordNet	**77.45**	78.04	58.80	77.45	78.04	58.80

(a) Accuracy

Synset	Concatenated			Averaged		
	Tratz	NomBank	PCEDT	Tratz	NomBank	PCEDT
N	**70.46**	49.92	38.19	**70.61**	50.85	36.63
N+V	70.23	50.27	**39.00**	69.76	50.61	35.34
E	70.64	50.84	38.91	70.02	50.80	35.13
No-WordNet	69.89	**58.92**	37.10	69.89	**58.92**	**37.10**

(b) Macro-average F_1 score

Table 6.12: Accuracy (a) and macro-averaged F_1 score (b) of the WordNet models on the development sets of the three datasets. N: Nouns. N+V: Nouns and verbs. E: All synonyms regardless of their PoS tag.

model that does not use any WordNet features (referred to as 'No-WordNet' in the last row in Tables 6.12a and 6.12b). More specifically, the model's accuracy on the Tratz dataset becomes slightly worse when using the concatenated representation of the synonyms. Further, the Tratz accuracy remains more or less the same using the averaged representation of the synonyms. The macro-averaged F_1 scores remain more or less the same also. The accuracy on NomBank improves by 0.76% points when using the full synset (regardless of the PoS tag) in the concatenated setup, and when using nouns only in the averaged setup. The macro-averaged F_1 results are, however, dramatically worse in all the models that use WordNet features in comparison to the model that does not use any WordNet features. For PCEDT, the concatenated setup seems to help overall, with the greatest improvement in accuracy being about 1% when using the N+V synset. This improvement is also observed on the macro-averaged F_1 score. In contrast, the averaged setup either does not help or slightly improves (or worsens) the accuracy depending on the PoS tags of the synonyms considered from the synset.

Since the improvements in accuracy across the three datasets are rather small, we semi-manually analyze the errors of the WordNet models. Whether in PCEDT or NomBank, we find that 94% of the errors are the same regardless of the PoS of the synset words or the choice of feature representation (i.e. concatenated or averaged). In Tratz, 86.70% of the errors are the same across all the PoS tags of the synset and the two feature representation setups.

Comparing the misclassification errors of the models that use WordNet features and the model that does not use any WordNet features, we find that about 92.60% of the misclassification errors are the same in PCEDT; therefore, it is clear that the neural classification model makes highly similar predictions with and without WordNet features. In NomBank, the picture is slightly different as the percentage of common misclassification errors goes down to 72% (from 94%) when we include the model that does not use WordNet features.

To understand the difference between the models that use WordNet features, on the one hand, and the one that does not, on the other hand, we compare the misclassification errors made by these models on NomBank. More concretely, we inspect the cases where any of the WordNet models predicted the correct relation while the No-WordNet model failed to do so (i.e. we compare the union of the correctly predicted compounds by the WordNet models to the error set of the No-WordNet model). Further, we also inspect

the cases where the No-WordNet model made the correct predictions and all of the WordNet models made classification mistakes (i.e. we compare the intersection of the misclassified compounds by the WordNet models to the predictions of the No-WordNet model). We find that about 63% of the correctly predicted compounds by the WordNet models, but misclassified by the No-WordNet model, are of type `ARG1`, which is the most frequent relation in NomBank. On the other hand, the correct predictions by the No-WordNet model only (i.e. the compounds that none of the WordNet models gets right) are more diverse in terms of their relations, including 34.78% of type `ARG2`, 19.56% `ARG1` 15.22% `ARG3` and 13.04% `ARG0`. Further, none of the WordNet models predict the relation `ARGM-MNR` which explains the relatively large deterioration in macro-averaged F_1 score when WordNet features are introduced.

Based on these insights, we believe that the WordNet features we encode do not help improve the model's ability to learn the different relations in the three datasets considered. That said, we cannot conclude here that WordNet features—in general—would not be helpful, as we realize that our representation of WordNet features using word embedding vectors is potentially suboptimal. In essence, word embedding vectors implicitly encode semantic similarity information, and therefore adding more word embedding vectors to the network as representation of WordNet features may be redundant. A better approach to exploit the information encoded in lexical resources like WordNet might be to use them directly as input to the neural classifier instead of using their word embedding representation. The utility of WordNet features in neural architectures, therefore, remains in part an open question for future research.

6.7 Training Data

Throughout this chapter, we experimented with different approaches to improve the performance of our neural classification model; from varying the input representation models (word embeddings) to introducing new architectures and additional features. While some of these experiments revealed interesting findings, the overall performance of the model did not exhibit any 'breakthrough' improvements. This prompts us to ask whether adding more training data would be of help. Therefore, in this section, we experiment with the effect of incrementally increasing the size of the training data for

the three compound datasets—that is, we investigate the learning curves of the neural classification model on the three datasets.

We start by training the neural classification model with only 20% of the available training data in each dataset, and then add one tenth of the training data and repeat until we reach 100% of the training data. We evaluate the models on the full development split at each point. Furthermore, since those subsets of training data are drawn randomly from the full training split, we repeat these experiments ten times for each subset to make sure that the effect we see is not an artifact of the selected examples. Note, that the 'randomness' here only affects the selection of training examples. We still use the same random seed to initialize the weights of the neural network.

Figures 6.4, 6.5 and 6.6 show the learning curves on the Tratz, NomBank and PCEDT datasets, respectively. The curves are drawn using a second degree polynomial regression. The regression confidence intervals are shown around the regression line. The far ends of the vertical bars in the three figures represent the minimum and maximum accuracy of the ten runs at each point and the solid circles represent the mean accuracy of the ten runs. Given that the neural classifier arrives at varied accuracy ranges depending on the dataset, we draw the learning curves separately. Further, these accuracy ranges can be both wide (in the case of the Tratz dataset) and narrow (for PCEDT), and hence to avoid the visual amplification of one in comparison to the other, we use the same relative range size on the y axis for all the figures.

From the figures, we can immediately see that more training data can actually help improve the accuracy of the neural classifier on the three datasets. However, this observation is most obvious on the Tratz dataset, and to a slightly lesser degree on NomBank but it is perhaps least obvious on PCEDT. It is important to highlight here, though, that the comparison across the three datasets is not completely fair, because 20% of the training data in the Tratz dataset, in terms of the absolute number of compound instances, amounts to a little more than 40% of the training data in NomBank and PCEDT. We see a relatively large deviation in accuracy across the ten runs for each of the three datasets on the 20% point; this effect is not surprising, since the random subset of 20% sampled for each of the ten runs is more likely to have varying degrees of representativeness of the full dataset. However, what is surprising is to see the large deviation in the model's accuracy on PCEDT when we use 100% of the training data (cf. Figure 6.6). In this particular scenario it seems that the order of the training data has a large impact on the performance

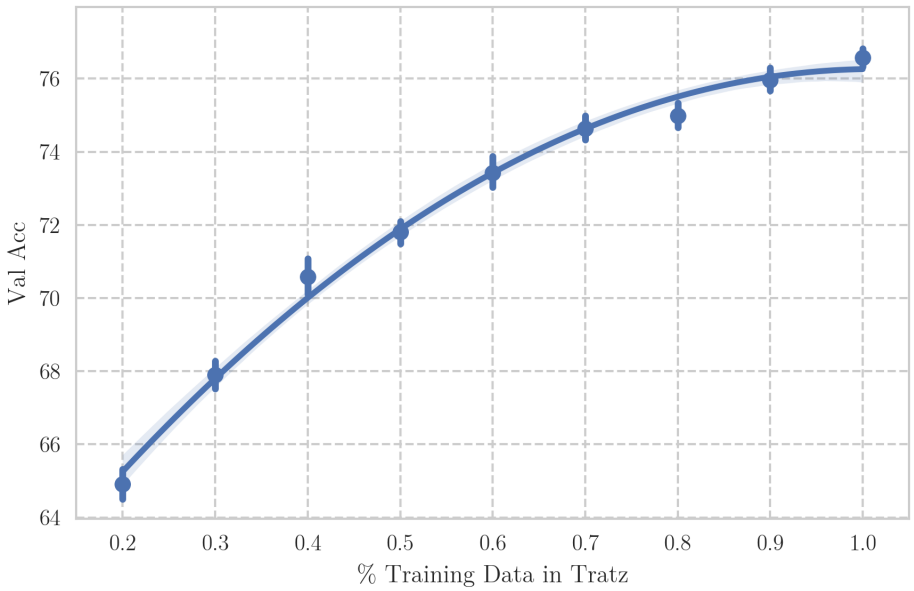


Figure 6.4: Learning curve on the Tratz dataset. The vertical bars show the minimum and maximum accuracy of the ten runs at each point and the solid circles represent the mean accuracy of the ten runs. The regression confidence interval are shown around the regression line.

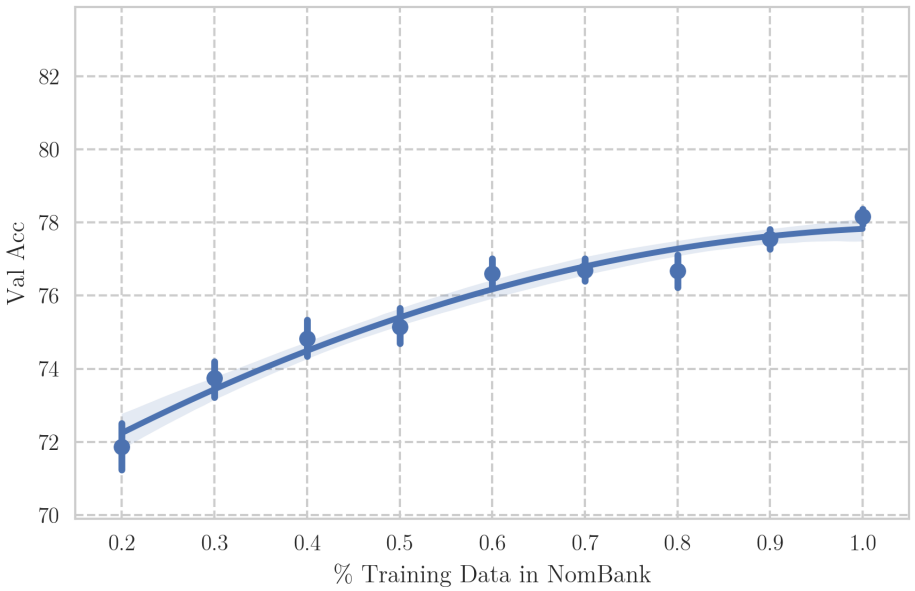


Figure 6.5: Learning curve on the NomBank dataset. The vertical bars show the minimum and maximum accuracy of the ten runs at each point and the solid circles represent the mean accuracy of the ten runs. The regression confidence interval are shown around the regression line.

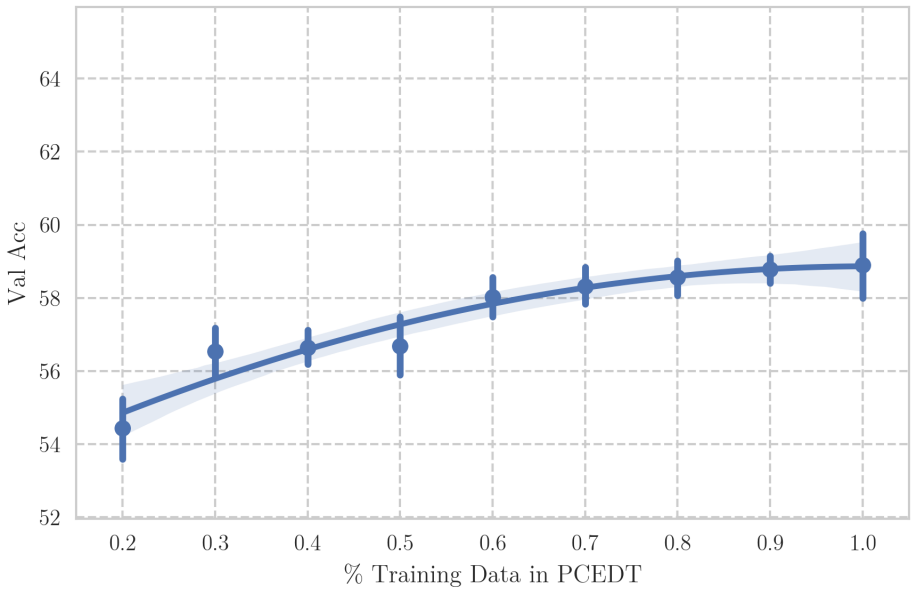


Figure 6.6: Learning curve on the PCEDT dataset. The vertical bars show the minimum and maximum accuracy of the ten runs at each point and the solid circles represent the mean accuracy of the ten runs. The regression confidence interval are shown around the regression line.

	Tratz	NomBank	PCEDT
Min	76.51	77.17	55.98
Max	77.60	78.91	60.87
Mean	76.95	77.99	59.06
STD	0.30	0.43	1.10

Table 6.13: Minimum, maximum, mean and standard deviation (STD) of accuracy over 20 runs with 20 different random seeds for initialization and shuffling.

of the model, because in each of the ten runs that use the full training split the only difference is the order of training data—as mentioned before, the random seeds were fixed when initializing the weights of the neural network across the ten runs, but not upon selecting a random sample of the dataset.

To confirm that the effect we see is indeed a result of the order in which the training instances were presented to the classifier, we run two additional series of experiments. In the first one, we train the classifier using the exact same hyperparameters and input representation but with 20 different random seeds on all of our datasets. These random seeds are used throughout the training process, which means that the order of the training instances will be different from one random seed to another (because they are shuffled prior to each training epoch). In the second series of experiments, we use the same 20 random seeds but only to initialize the weights of the neural network and after that the random seed is fixed across all the experiments; that is, the training instances are shuffled in the same way for the 20 runs. Tables 6.13 and 6.14 summarize the maximum and minimum accuracies observed for each dataset as well as the mean and standard deviation of the 20 runs per dataset.

Contrasting the results in Tables 6.13 and 6.14, it becomes more likely that the order of training examples influences the final performance of the model on PCEDT. This is much less of a problem for both NomBank and Tratz, where the two series of experiments lead to similar values of mean and standard deviation. Arguably, observing variance in the model's performance due to different random seeds is not surprising given the nondeterministic nature of neural networks (Reimers & Gurevych, 2017; Crane, 2018). However, it is the contrast between the PCEDT results, on the one hand, and the Tratz and NomBank results, on the other hand, that is of interest here.

	Tratz	NomBank	PCEDT
Min	76.61	77.93	58.04
Max	77.24	79.13	59.35
Mean	76.86	78.48	58.54
STD	0.16	0.31	0.29

Table 6.14: Minimum, maximum, mean and standard deviation (STD) of accuracy over 20 runs with 20 different random seeds to initialize the weights in the neural network, but the same random seed for shuffling. The training instances are also presented in the same order.

The two primary takeaway results from the tables above are: First, the PCEDT results vary depending on the order of training examples during training. Therefore, we will report the (averaged) results of multiple runs towards the end of Chapter 7, following the same experimental setup introduced in this section. The inconsistent empirical results on PCEDT nonetheless support our previously stated suspicion regarding the annotations in the English part of PCEDT. Second, in all cases, we see some variation between the minimum and maximum accuracy achieved as a result of random initialization. Hence, in the following chapter, we experiment with learning strategies (namely transfer learning) which could provide a more principled approach to initialization yielding more stable, and potentially better, performance. Seeing that adding more training data can be of help, we also experiment with multi-task learning which also provides us with a way to exploit the dual annotation in our training data for NomBank and PCEDT.

6.8 Conclusion

In this chapter, we covered a rather wide and comprehensive series of questions related to neural classification models for compound interpretation in light of three compound datasets, viz. Tratz, NomBank and PCEDT. In the following, we try to summarize some of the important findings in our efforts to replicate past work § 6.1.1 and the effect of the following 'factors' on the classification model: (1) word embeddings § 6.3, (2) model architecture § 6.4, (3) hyperparameters § 6.5, (4) WordNet features § 6.6 and (5) training data § 6.7 and random initialization.

Replicating Dima and Hinrichs (2015): In § 6.1, we reviewed and replicated the first study on neural compound interpretation by Dima and Hinrichs (2015) as a starting point for the experiments that followed in the chapter. Dima and Hinrichs (2015) used a feedforward neural network with word embeddings to learn the semantic relations in the Tratz dataset. The results of our replication experiment were on par with what was reported by Dima and Hinrichs (2015) using the GloVe embedding model, but we achieved remarkably better results using the word2vec model. Since we are mainly interested in a successful replication of Dima and Hinrichs's (2015) work—which we have achieved—we did not pursue a detailed explanation of why our model arrives at better results when using the same word2vec model. We did highlight, however, that our replica implementation is in fact a near-replica, because we used a different optimization function and different library for neural networks.

Effect of Word Embeddings: In § 6.3, we investigated the effect of systematically varied word embeddings (used as input representation) on the final performance of our neural classifiers for the Tratz, NomBank and PCEDT datasets. More specifically, we focused on three aspect of word embeddings: (1) text pre-processing, (2) embedding dimensionality and (3) embedding training data. We also evaluated the effect of embedding fine-tuning along with the second and third aforementioned aspects. The overarching conclusion of our embedding experiments is, simply put, there is no one-size-fits-all word embedding model; that is, the absolute best embedding model (in terms of classification accuracy and macro-averaged F_1 score), to a large degree, depends on the dataset itself. For example, we found that lemmatization improves the NomBank classifier accuracy while it harms the Tratz classifier accuracy (and it almost has no effect on PCEDT). That said, we observed a few patterns that, by and large, hold across all three datasets. First, our experiments confirmed that fine-tuning word embeddings leads to better accuracy and macro-averaged F_1 scores. Second, we showed that the embedding dimensionality and size of training data play less of an important role when the embedding models are fine-tuned. That is not to say that the dimensionality of word embeddings is not important, but rather the impact of increasing the number of dimensions becomes less pronounced with fine-tuning. Third, our experimental results revealed that there are limits on how far higher dimensionality is better (especially in the fine-tuned mode), which somewhat

modulates the conclusion by Dima and Hinrichs (2015, p. 177) who write that "it was always the highest dimensional embedding that gave the best results".

Neural Architectures: In § 6.4, we experimented with alternative neural architectures which account for the assumption that the modifier and head nouns can contribute differently to determining the compound's relation. More concretely, we defined three architectures with constituent-specific layers in which the head and modifier nouns have separate embedding layers (E_S), hidden layers (H_S) or both (EH_S). Our experimental results showed that the benefit of the constituent-specific architecture depends on the evaluation measure we consider. We did not see any improvement in accuracy when using the three new architectures on NomBank and Tratz, and a small improvement on PCEDT. However, we did observe larger variation in the macro-averaged F_1 scores even though the accuracy scores were comparable. Upon further inspection of the results, we found that the relations that have 'prototypical' heads or modifiers, especially in NomBank, tend to benefit more from the constituent-specific classifiers. These relations, however, are relatively infrequent and therefore such an improvement was not reflected in terms of accuracy.

Model Hyperparameters: In § 6.5, we studied the impact of different hyperparameters on our classification models following two strategies: (1) sensitivity analysis in § 6.5.1 and (2) random search in § 6.5.2. In the sensitivity analysis study, we evaluated the impact of a selection of hyperparameters, in isolation, on the model's performance. Such an experimental setup, of course, makes strong assumption on the independence of different hyperparameters, but we compensated for this assumption in the random search experiments which consider seven hyperparameters and settings at the same time. In either case, however, our hyperparameter search experiments were geared towards confirming the plausibility of our hyperparameter choices rather than finding the configuration that leads to the absolute best results. In addition to confirming the validity of our hyperparameter choices, the results of the hyperparameter optimization experiments revealed a few patterns that apply to the three compound datasets. First, fine-tuning word embeddings helps improve the classification accuracy regardless of the values of other hyperparameters; this pattern was clear across 172 random search experiments per dataset. Second, extremely small hidden layers (e.g. four or eight units)

substantially degrade the classification accuracy of the models for all datasets, which is unsurprising given the reduced representational capability of the model. Third, adding a dropout layer might help improve the performance but the ideal dropout rate is dataset-dependent. Lastly, our unified hyperparameter configuration (cf. Table 6.3) ranks comparatively high among the 172 hyperparameter combinations explored by the random search study, especially on the Tratz and NomBank datasets. Our decision to continue to use the unified hyperparameter configuration is mainly to facilitate straightforward comparison with the experiments in Chapter 7.

WordNet Features: In § 6.6, we experimented with adding WordNet-based features to our models, which were inspired by the features used by Tratz and Hovy (2010). Overall, we did not see any remarkable improvement in the model's performance, even though we experimented with several types of features (extracted from the constituents' synsets in WordNet) and different representations of such features. However, we also explained that our representation of the WordNet features (using word embedding vectors) might have been suboptimal, and therefore further experimentation is required to determine the utility of WordNet-based features with neural models.

Training Data: In § 6.7, we studied the effect of incrementally increasing the size of the training data on the performance of the neural classifier for each of the three datasets. Specifically, we trained classification models on increasing amounts of data starting with only 20% of the training examples and then added one more tenth of the training data per step. To make sure that the results were not an artifact of how the subsets of the training data were sampled, we ran each experiment ten times and finally drew the learning curves based on the mean accuracy results. From the experimental results, it was clear that with more training data the model can achieve higher accuracy scores, especially for the Tratz and NomBank datasets. However, our experiments also revealed that the models can be sensitive to the order of training data (primarily on PCEDT), which promoted us to experiment with the impact of random initialization and ordering of training examples. In a series of experiments that control for the effect of random initialization and the shuffling of training examples, we found that the model's performance on PCEDT is susceptible to the order of examples presented during training. This proved to be less of an issue for NomBank and Tratz.

Chapter 7

Transfer and Multi-Task Learning for Compound Interpretation

In this chapter, we empirically evaluate the utility of two machine learning strategies, transfer and multi-task learning, that exploit the relationship between multiple, parallel annotations in our compound datasets. Through a comprehensive series of experiments, we investigate whether transfer learning via parameter initialization and multi-task learning via parameter sharing can help improve the performance of our neural classification model. Furthermore, we demonstrate how the dual annotation with relations over the same set of compounds in our dataset can be exploited to improve the overall accuracy of a neural classifier on the less frequent, but more difficult, relations in PCEDT and NomBank. We start with a brief introduction and motivation in § 7.1. We then, in § 7.2, move to define some basic terminology as well as the learning strategies used in this chapter and review a selection of related studies (§ 7.2.2). In § 7.3, we introduce how transfer and multi-task learning can be applied to compound interpretation and in § 7.4 we outline our experimental setup and models. In § 7.5, we detail the results of our experiments and present an in-depth error analysis on the development and test splits. In § 7.6, we gauge the ability of our neural models to generalize over unseen examples and the influence of lexical memorization on their performance. Lastly, in § 7.7, we investigate the benefits of transfer and multi-task learning to 'stabilize' the model's performance with respect to random initialization and order of training examples—a recurring issue we observed in the previous chapter.

7.1 Introduction and Motivation

Thus far we have only considered single-task learning (STL) as a learning strategy in our experiments—that is, we experimented with separate neural classifiers to learn the interpretation of noun–noun compounds in the three datasets. In contrast, we dedicate this chapter to comprehensive experimentation on transfer learning (TL) and multi-task learning (MTL), two learning strategies that can take advantage of the relationship between the NomBank and PCEDT compound datasets. This chapter, furthermore, partly aims to contribute to the more general goal of studying the use of TL and MTL for NLP problems. The work presented in this chapter is hence motivated by several observations rooted in properties of two of the compound datasets we use (viz. PCEDT and NomBank), the very nature of the neural classification models introduced in the preceding chapter and—of course—findings by previous research on related problems.

First, in Chapter 6 we found that the (random) initialization of the neural classifier's weights as well as the order of training examples non-trivially affect the final performance of the classifier in terms of accuracy. Furthermore, we also observe that adding more training data is likely to improve the performance of the neural classifier (cf. § 6.7). As will be explained later in this chapter, transfer learning and multi-task learning can potentially offer solutions for the two problems stated above in terms of parameter (or weight) initialization and additional training examples.

Second, over the past few years, the interest in using transfer learning and multi-task learning in NLP has surged, all the while showing 'mixed' results depending on, inter alia, the main and auxiliary tasks, model architecture and datasets (Collobert & Weston, 2008; Mou et al., 2016; Søgaard & Goldberg, 2016; Martínez Alonso & Plank, 2017; Bingel & Søgaard, 2017). These 'mixed' results coupled with the fact that neither TL nor MTL has been applied to compound interpretation before further motivate our extensive empirical study on the use of TL and MTL for compound interpretation, not only to supplement existing research on the utility of TL and MTL for semantic NLP tasks in general, but also to determine their benefits for compound interpretation in particular. It has been observed, however, that TL and MTL often work best when the main and auxiliary tasks are somehow related (Mou et al., 2016), which is the case for our noun–noun compound datasets (cf. § 7.3).

Third, one of the primary motivations for using multi-task learning is to improve generalization by "leveraging the domain-specific information contained in the training signals of *related* tasks" (Caruana, 1997, p. 41). In this chapter, we investigate whether or not TL and MTL can be used as a kind of *regularizer* to learn to predict the infrequent relations in a highly skewed distribution of relations in the noun–noun compound datasets of NomBank and PCEDT.[1]

7.2 Terminology and Definitions

In this section, we introduce the basic idea behind transfer and multi-task learning in general without distinguishing between the two learning strategies; though, as will be shown later, we do in fact distinguish between them, but until we do that we will refer to both learning strategies as multi-task learning (or MTL). In addition, we define some of the terminology that we perceive as generally vague, underspecified concepts in the context of multi-task learning.

Multi-task learning is perhaps best understood in contrast to the more common approach of single-task learning. To a certain degree, all the experiments we introduced in Chapter 6 fall under the definition of single-task learning, in the sense that we train and optimize three separate classification models to learn the semantic relations in the three compound datasets.[2] At no point in Chapter 6 did we exploit the fact that at least two of our compound datasets are closely related (viz. PCEDT and NomBank); rather we treated them as completely separate tasks even though we know that, in practice, this is not the case. Multi-task learning—as a machine learning strategy—allows us to take advantage of the relationship between our tasks (or datasets) by learning one model for the two tasks, among other approaches.

Multi-task learning makes use of multiple related tasks to either (1) improve the generalization of one specific task only or (2) learn multiple tasks in one model at the same time. The rationale behind multi-task learning can be summarized in that task-specific information and representations available for a given a model can be also beneficial for other related tasks. More precisely, in his seminal work, Caruana (1997, p. 41) writes:

[1] Parts of this chapter are based on the work we published in Fares et al. (2018)

[2] In fact, using pre-trained word embedding models already introduces some aspect of transfer learning to our experiments in Chapter 6, but we will come back to this point later.

> "MTL improves generalization by leveraging the domain-specific information contained in the training signals of *related* tasks. It does this by training tasks in parallel while using a shared representation. In effect, the training signals for the extra tasks serve as an inductive bias."

At first glance, the excerpt from Caruana (1997) leaves little room for interpretation, but in the grand scheme of things we find that some terms require further specification. One of these terms is *generalization* which has been described as a "suitcase word" by Lipton and Steinhardt (2018). In their article, Lipton and Steinhardt (2018) try to identify some of the problematic aspects in machine learning scholarship, which include the overuse of certain terms; they describe such terms as "suitcase words"—a concept they borrow from Minsky (2007). Therefore, in the interest of full transparency, in the following we explain some of the keywords in the quote by Caruana (1997) above as we transition towards a more detailed definition of multi-task learning in our experiments.

Generalization: Given the context of multi-task learning, generalization can be understood in at least two ways. First, the more 'traditional' definition, which is the (expected) difference between the model's performance on the training split and the development or test splits. This difference in performance is often computed between the training and test splits and referred to as generalization error (Hastie et al., 2009, p. 220). Second, it is possible to interpret generalization as the model's ability to generalize over a new domain or type of input (Williams, 2013); for example, the ability of part-of-speech taggers trained on edited newspaper text to perform equally well on social media text. We will use generalization in both senses throughout this chapter; in § 7.6, for example, we quantify the model's ability to generalize over compounds with unseen constituents. In addition, we will also consider a slightly variant perspective on generalization in § 7.5.2, where we compare the difference between the model's performance (in terms of accuracy) on the development set and the test set.

Inductive bias: This concept is somewhat related to generalization. Training an ML model is essentially looking for a hypothesis that characterizes the training examples and allows the model to 'generalize' over unseen examples (be it from a different domain or not). Inductive bias can be then understood as bias to make the ML model favor one hypothesis over others

(Caruana, 1993). In the context of multi-task learning, the training signals from other tasks become an additional source of inductive bias. The final set of weights for a given neural model are somehow the result of finding a local (or sometimes global) minimum. However, if there are several local minima (which is often the case), then the decision of which local minimum to 'choose' can be influenced by inductive bias. In multi-task learning, the idea is to influence the choice of the local minimum or hypothesis based on signals (i.e. inductive bias) from other related tasks. This hypothesis is hence assumed to lead to favorable generalization abilities because it combines information from multiple 'related' tasks.

Relatedness: The earliest work on multi-task learning by Caruana (1997) does not offer a clear, formal definition of when two tasks are considered *related*. On the one hand, he states that if the predictions on two tasks are functions of the same input, then these two tasks are related. On the other hand, he also states that injecting noise into one task can also help the model generalize (by way of regularization). Of course, such noise is by no means to be considered 'related' to the main task even though it might help improve generalization. This may be the reason why Caruana (1997, p. 71) writes "[w]e may never have a theory of relatedness that allows us to reliably predict which tasks will help or hurt each other when used for inductive transfer." Perhaps for the same reason, Goldberg (2017, p. 244) also claims that choosing related tasks for MTL is "more of an art than science". In the following we present a more recent formal definition of relatedness, based on which we show why two of our compound datasets (viz. NomBank and PCEDT) can be considered related.

Formal definition: We follow the same notation used by Pan and Yang (2010) to define how two tasks can be related as well as explain multi-task learning in more formal terms.

Assume we have a classification task $\mathcal{T}$, which can be defined in terms of all training pairs (X, Y) and a probability distribution $P(X)$, where:

$$X = x_1, \ldots, x_N \in \mathcal{X}$$
$$Y = y_1, \ldots, y_N \in \mathcal{Y}$$

$\mathcal{X}$ is the input feature space, $\mathcal{Y}$ is the set of all labels and N is the size of the

training data. Further, the classification task domain $\mathcal{D}$ is defined by the pair $\{\mathcal{X}, P(X)\}$. The goal of a machine learning algorithm is to learn a function $f(X)$—from the training pairs (X, Y)—to predict Y based on the features of the input examples X.

Now assuming that we have two ML tasks, $\mathcal{T}_a$ and $\mathcal{T}_b$, we would train two separate models (i.e. learn two separate functions f_a and f_b) to predict Y_a and Y_b in a single-task learning setup. In MTL, however, we can either learn one function for the two tasks or still learn two separate functions while relying on information from the two tasks. As mentioned earlier, if $\mathcal{T}_a$ and $\mathcal{T}_b$ are related somehow, either explicitly or implicitly, MTL can improve the generalization of either task or both (Caruana, 1997; Pan & Yang, 2010; Mou et al., 2016). Two tasks are considered related when their domains ($\mathcal{D}_a$ and $\mathcal{D}_b$) are similar but their label sets are different $\mathcal{Y}_a \neq \mathcal{Y}_b$, which is similar to the scenario Caruana (1997) describes as two tasks being functions of the same input. Another case of two tasks being related is when their domains are different but their label sets are identical, i.e. $\mathcal{Y}_a = \mathcal{Y}_b$ (Pan & Yang, 2010).[3]

Given the former definition of 'relatedness', noun–noun compound interpretation in the context of the NomBank and PCEDT datasets is a well-suited candidate for TL and MTL, because the training examples in the two datasets are identical, but the label sets are different, i.e.:

$$X_{PCEDT} = X_{NomBank}$$
$$\mathcal{Y}_{PCEDT} \neq \mathcal{Y}_{NomBank}$$

Thus, we can reasonably assume that it is potentially beneficial to use MTL in the context of our noun–noun compound datasets.

7.2.1 TL vs. MTL

In 1995, researchers gathered in a post-conference workshop under the annual Conference on Neural Information Processing Systems to discuss "knowledge consolidation and transfer in inductive systems".[4] The overarching aim was to develop "methods that capitalize on previously acquired domain knowledge".

[3]When the label sets are identical, multi-task learning (in the general sense) practically becomes a technique for domain adaptation.

[4]The workshop web page: `http://plato.acadiau.ca/courses/comp/dsilver/NIPS95_LTL/transfer.workshop.1995.html`. Accessed: 11 December 2018.

These methods came under different names: multi-task learning, lifelong learning, knowledge consolidation, inter alia. Even though it has been well over twenty years since that workshop took place, the names of such methods are still at times conflated and used somewhat interchangeably in the literature; for example, what we call transfer and multi-task learning in this chapter are both referred to as transfer learning by Mou et al. (2016). Therefore, for the sake of clarity, in this section we aim to define and make explicit the difference between what we refer to as *transfer learning* (TL) versus *multi-task learning* (MTL). In doing so, we do not seek to argue that such a distinction between TL and MTL should be made at all times, rather merely make explicit the meaning of two concepts central to this chapter.

We define transfer learning as using the parameters (i.e. weights in neural networks) of one model trained on task $\mathcal{T}_a$ to initialize another model for task $\mathcal{T}_b$. Mou et al. (2016) refer to this method as "parameter initialization".[5] That is, the weights of a model that have been trained on a given task can be used to initialize another model for another related task. Hence, in this sense, TL as a learning strategy entails some sort of knowledge 'transfer' across tasks but in a sequential manner. We interpret multi-task learning as training (parts of) the same model (e.g. neural network) to learn tasks $\mathcal{T}_a$ and $\mathcal{T}_b$ at the same time, i.e. learning one set of parameters for both tasks. This approach to multi-task learning is also known as hard parameter sharing. Hence, not only is there simultaneous learning in MTL (in contrast to 'sequential' transfer), but also parts of the model itself are shared among the tasks. It is important to highlight here that the ultimate aim of using MTL need not be training one model for many tasks. In fact, Caruana (1997, p. 68) clearly states that the main benefit of MTL is not to "reduce the number of models that must be learned", but rather to exploit the training information contained in other tasks to learn a specific one. Consequently, we often speak of 'auxiliary' and 'main' tasks in MTL, where the auxiliary tasks are the tasks used to introduce inductive bias so that the model is able to learn better generalizations over the main task. Likewise, we also speak of 'source' and 'target' tasks in transfer learning; the source task being the task whose model's weights are used to initialize a model for the target task.

In the following section, we review some of the previous work on TL and

[5] In fact, using *pre-trained* word embeddings as input representation (like we did in Chapter 6) is in a sense a form of *unsupervised* transfer learning, but in this work we focus on transfer methods based on supervised learning.

MTL for NLP, not only to situate our work within the realm of existing studies but also to help paint a clearer picture of what TL and MTL mean.

7.2.2 Related Work

To the best of our knowledge, transfer and multi-task learning have never been applied to noun–noun compound interpretation before. Therefore, in this section, we review several recent studies on TL and MTL for other NLP tasks that are somehow related to our experimental setup or present comprehensive experiments on the use of TL and MTL for a variety of NLP tasks, including named entity recognition and semantic labeling (Martínez Alonso & Plank, 2017), sentence-level sentiment classification (Mou et al., 2016), super-tagging and chunking (Bingel & Søgaard, 2017) and semantic dependency parsing (Peng et al., 2017).

Mou et al. (2016) study the 'transferability' of knowledge across different NLP tasks using six datasets for sentence classification (sentiment and question type classification) and sentence pair classification (sentence relation classification such as entailment and contradiction). They organize their experiments in two series, each comprising three datasets corresponding to whether these datasets deal with sentence classification or sentence pair classification. The largest datasets in these two series of experiments serve as the 'source' task for the other two target tasks. Furthermore, they train two neural networks for these experiments; a long short term memory (LSTM) recurrent neural network (RNN) for sentence classification and a convolutional neural network (CNN) for sentence pair classification. Mou et al. (2016) define two approaches for what they call transfer learning: parameter initialization (using the parameters of a model trained on the source task to initialize a model for the target task) and multi-task learning (training a model on the source and target tasks simultaneously). As explained in § 7.2.1, we refer to the first approach as transfer learning and the second one as multi-task learning, whereas Mou et al. (2016) refer to both approaches as transfer learning, i.e. Mou et al. (2016) consider MTL a type of TL. One of the interesting findings by Mou et al. (2016) is that knowledge 'transferability' largely depends on the semantic relatedness or similarity between the source and target tasks. Mou et al. (2016) seem to define semantic similarity based on the similarity between the classification labels in the source and target tasks; for example, they consider two sentence classification datasets, namely the IMDB dataset

and the Movie Review Data (Pang et al., 2002), semantically similar because they both annotate sentiment polarity (positive vs. negative). Given their conclusion, our NomBank and PCEDT datasets potentially stand to benefit from TL and MTL, because they can be considered semantically similar in terms of their labels (in addition to annotating the exact same set of noun–noun compounds). Lastly, Mou et al. (2016) report that the performance of parameter initialization is generally comparable to that of multi-task learning, and that combining the two approaches does not lead to further gains in the tasks they study.

Martínez Alonso and Plank (2017) try to establish when MTL works for semantic sequence classification in terms of task-dependent characteristics. They experiment with five semantic tasks (such as named entity recognition) as their main tasks and four morphosyntactic and frequency-based tasks (such as PoS tagging) as their auxiliary tasks. The neural model underlying all of their experiments is a bidirectional LSTM neural network. Their experimental design leads to a total of $1,440$ models where not only the combination of auxiliary and main tasks differs, but also the placement of the output layer of the auxiliary task. Overall, Martínez Alonso and Plank (2017) report that only one of the five main tasks sees a significant improvement using MTL in comparison to the STL baseline. Note that Martínez Alonso and Plank (2017) do not use pre-trained word embeddings as input representation in their experiments. The main conclusion Martínez Alonso and Plank (2017) draw from their experiments relates the auxiliary task label distribution to the potential benefits of MTL. More concretely, their MTL architecture could benefit most from "compact and more uniform label distributions" of the auxiliary tasks. Given the architecture of their model (i.e. having three hidden layers), Martínez Alonso and Plank (2017) experiment with placing the output layer of some auxiliary tasks, such as PoS tagging, at inner layers. They find that the choice of where to place the output layer does not lead to any systematic variation in the final results.

Bingel and Søgaard (2017) conduct a comprehensive set of experiments in an effort to determine what relations among the tasks can guarantee benefits from MTL. Their study is focused on sequence labeling using a bidirectional LSTM, in which they compare the performance of 90 MTL models (i.e. 90 combinations of ten NLP tasks) to the corresponding STL models. Of these 90 combinations, 40 see improvement using MTL over the STL models. They then analyze these results by training a logistic regression model to predict the

benefits of MTL based on features inherent to the tasks themselves as well as features extracted from the learning curves of the STL models. This method of analysis allows them to find which features of the logistic regression model were most predictive, and since these features are basically characteristics of the tasks themselves they are able to determine what contributes to making MTL beneficial. Based on the above, they find that the strongest predictor of MTL benefits is the learning patterns of the main and auxiliary tasks in terms of the shape of their learning curves; if the main task is likely to plateau rather early in training (i.e. get stuck in a local minimum), then using a non-plateauing auxiliary task would be of help. Another strong predictor is the out-of-vocabulary rate for the main task, which is to be expected since the embedding weights are shared in the MTL models across the tasks, and hence more embedding weights are updated during training. Upon private correspondence with the authors, they explained that their definition of out-of-vocabulary words actually refers to words that are only observed in the test set of the target task (and not in the training set).[6] If some of these unseen words in the main task occur in the training set of the auxiliary task they will be updated during training, and hence the potential benefit of MTL in this context. We find this to be an interesting observation and stipulate that it may perhaps provide a partial remedy to the undesired effect of fine-tuning a subset of the vectors in word embedding models while keeping the rest static (Astudillo et al., 2015).

Unlike the three studies we reviewed above, Peng et al. (2017) focus on one NLP problem only, viz. semantic dependency parsing. We include their work in our literature review because their hypothesis and motivation to use multi-task learning are similar to ours. They experiment with learning dependency parsing across three linguistic formalisms from the 2014 and 2015 SemEval shared tasks on broad-coverage semantic dependency parsing (Oepen et al., 2014, 2015). They hypothesize that the "overlap among the theories and their corresponding representations can be exploited using multitask learning" (Peng et al., 2017, p. 2037). We assume the same hypothesis for noun–noun compound interpretation across parallel annotations in § 7.3. In brief, Peng et al. (2017) train a bidirectional LSTM composed with a multi-layer perceptron (MLP) as their single-task learning model (which also serves as their baseline).

[6]Out-of-vocabulary words or rate sometimes refers to the words that are not represented in the word embedding model itself (i.e. words that do not have an embedding vector), but that is not what Bingel and Søgaard (2017) refer to in their work.

They then extend the STL model to an MTL model via parameter sharing (in the bidirectional LSTM) as well as a single MTL model that uses joint inference across the three formalisms. One of the important findings Peng et al. (2017) report is that structural similarity between the formalisms affects the utility of multi-task learning in their experiments. More specifically, they find that two of the formalisms are more similar to each other than the third one, and this pair of formalisms is the one that sees the highest improvement when MTL is applied.

Finally, even though the aforementioned studies experiment with different NLP tasks and assume slightly different definitions of TL and MTL, we find an overarching conclusion across all of them: The potential benefits of transfer and multi-task learning largely depend on the properties of the main and auxiliary tasks as well as the datasets at hand. To summarize, Mou et al. (2016) emphasize the importance of semantic similarity between the source and target tasks, whereas Martínez Alonso and Plank (2017) report that the skewedness of the data distribution in auxiliary tasks plays an important role in determining whether or not MTL helps. Bingel and Søgaard (2017) find the learning pattern of the auxiliary and main tasks to be a strong predictor of the benefit of MTL, where "target tasks that quickly plateau" benefit most from "non-plateauing auxiliary tasks". Peng et al. (2017) observe "structural similarity" between the main and auxiliary tasks as an important factor.

7.3 TL & MTL for Compound Interpretation

Taken together, the conclusions of the previous studies reviewed in § 7.2.2 indicate that transfer and multi-task learning might be of help in our tasks of noun–noun compound interpretation. The definite answer, however, has to be pursued empirically. That said, boosting the model's performance is but one of several potential benefits of transfer and multi-task learning. In § 6.7 we observed that random initialization of the model's parameters directly affects the overall performance of the model as well as its stability. In addition, we also observe that adding more training data can help boost the model's performance, though the order of the training examples seems to have an effect as well (especially on PCEDT). In this section we explain how transfer and multi-task learning can potentially offer a remedy for these two issues in the context of our task and datasets—though the ultimate answer remains to be seen empirically (in § 7.7). We also explain our hypothesis of

why the NomBank and PCEDT datasets are especially suited for a transfer and multi-task learning setup.

In transfer learning, as defined in § 7.2.1, we use the weights of a model trained on some task $\mathcal{T}_a$ to initialize a model for another task $\mathcal{T}_b$. Seen as an initialization strategy, transfer learning—in theory at least—offers a more informed, and potentially stable, alternative to random initialization. Transfer learning can be considered an 'informed' initialization strategy in our case, because our source and target tasks are closely related (cf. § 7.2 and the following paragraph). While multi-task learning in our particular setup does not add more training data in terms of the numbers of examples, it does add another source of information to the model, by requiring it to learn the semantic interpretation of the same noun–noun compounds in two distinct annotation frameworks (NomBank and PCEDT).

The last point leads to the question of how 'related' are these two datasets. In § 7.2, we explained the 'relatedness' between NomBank and PCEDT in terms of their identical inputs X, i.e. the fact that the actual compound examples are the same across the two datasets but their labels are different. Here we argue that even their labels, $\mathcal{Y}_{NomBank}$ and $\mathcal{Y}_{PCEDT}$, are also related. Abstractly, many relations in PCEDT and NomBank describe similar semantic concepts, since they annotate the semantics of the same text. We detailed the correspondence between the annotations in the two datasets in § 4.5.2; for example, we reported that the temporal relation in NomBank (`ARGM-TMP`) and its counterpart in PCEDT (`TWHEN`) exhibit a relatively consistent behavior across frameworks as they annotate many of the same compounds. However, as also explained in § 4.5.2, some abstractly similar relations do not align well in practice; for example, the functor `AIM` in PCEDT and the modifier `ARGM-PNC` in NomBank express a similar semantic concept (*Purpose* according to the annotation guidelines of the two datasets), but the overlap between the sets of compounds they annotate in practice is rather small. Nonetheless, it is still plausible to hypothesize that the (partial) similarities between the two annotation frameworks and label sets can be exploited in the form of transfer and multi-task learning. As such, the NomBank and PCEDT datasets maximally enable TL and MTL perspectives, as they offer dual annotation with relations over the same underlying set of compounds. Furthermore, our TL and MTL experiments can be perceived as a method to conduct further empirical, contrastive analysis and comparison between the two annotation frameworks themselves (NomBank and PCEDT).

When it comes to the Tratz dataset, we expect TL and MTL to have less of an effect because (1) the labels of Tratz are tailored to noun–noun compound interpretation, unlike NomBank and PCEDT that express the semantics of compounds in a broader linguistic context and (2) the Tratz dataset annotates a different set of compounds, which means there is little lexical overlap between the Tratz dataset on one hand and PCEDT and NomBank on the other hand. Therefore, we limit our experiments to NomBank and PCEDT in this chapter.

7.4 Experimental Setup

In this section, we present the neural classification models used in our TL and MTL experiments. To isolate the effect of TL and MTL, we use the single-task learning model introduced in Chapter 6 as our baseline, and then we extend it to apply TL and MTL.

7.4.1 Single-Task Learning Model

Before we embark on describing how we implement TL and MTL, we briefly review the single-task learning (STL) model introduced in Chapter 6. The baseline STL model is a feed-forward neural network consisting of: (1) an input layer, (2) an embedding layer, (3) a hidden layer and (4) an output layer. The input layer is simply two integers specifying the indices of a compound's constituents in the embedding layer where the word embedding vectors are stored; the selected word embedding vectors are then fed to a fully connected hidden layer whose size is the same as the number of dimensions of the word embedding vectors. Finally, a Softmax function is applied on the output layer and the most likely relation is selected.

Throughout this chapter, the compound's constituents are represented using a 300-dimensional GloVe word embedding model trained on an English Wikipedia dump and Gigaword Fifth Edition (which is the same embedding model we settled for in § 6.3). When looking up a word in the embedding model, if it is not found we check if the word is uppercased and look up the same word in lowercase. If a word is hyphenated and is not found in the embedding vocabulary, we split it on the hyphen and average the vectors of its parts (if they exist in the vocabulary). If after these steps the word is still not found, we use a designated vector for unknown words.

The model's hyperparameters and settings remain the same as in the 'unified hyperparameter configuration', as outlined in Table 6.3, but we repeat them here for simplicity. The weights of the embedding layer (i.e. the word embeddings) are updated during training in all the experiments. The optimization function we use in all the models is Adaptive Moment Estimation, known as *Adam*, with the default learning rate ($\eta = 0.001$). The loss function is negative-log likelihood (aka categorical cross-entropy). We use a *Sigmoid* activation function on the hidden layer units and *Softmax* on the output layer. All the models are trained using mini-batches of size five. The number of epochs is set to 50, but we also use an early stopping criterion based on the model's accuracy on the development set (i.e. training is interrupted if the development accuracy does not improve over five consecutive epochs). Like the experiments in Chapter 6, we implement all the models in *Keras* with *TensforFlow* as backend.

Finally, in line with many previous studies (Collobert & Weston, 2008; Martínez Alonso & Plank, 2017; Bingel & Søgaard, 2017), we use the same hyperparameters for STL, TL and MTL to isolate the effect of TL and MTL. Needless to say, our TL and MTL might benefit from hyperparameter optimization but this would make direct comparison to our STL models less straightforward.

7.4.2 Transfer Learning Models

We define our transfer learning setup in a way similar to the work by Mou et al. (2016), in that we experiment with parameter initialization on all the layers of the neural model, except the output layer because it is task- or dataset-specific (Mou et al., 2016).

More specifically, transfer learning in our experiments amounts to training an STL model on PCEDT relations, for example, and then using (some of) its weights to initialize another model for NomBank relations. Given the architecture of the neural classifier described in the previous section, we identify the following three ways to implement TL:

1. TL_{E}: Transfer of the embedding layer weights.

2. TL_{H}: Transfer of the hidden layer weights.

3. TL_{EH}: Transfer of both the embedding and hidden layer weights.

In addition to the those three TL configurations, we distinguish between transfer learning from PCEDT to NomBank and vice versa; that is, either task can be used as source task or target task. Hence, we either start by training on the NomBank dataset and use the weights of the corresponding transfer layer to initialize the PCEDT model or the other way around. This leads to a total of six TL setups or models. Lastly, we do not freeze the transfered weights in any of the setups, since TL is partly perceived as a parameter initialization strategy. We acknowledge, however, that it could be well worth experimenting with freezing some of the TL models' layers or gradually unfreezing some of them.

7.4.3 Multi-Task Learning Models

In our MTL experiments, we train one model simultaneously to learn both PCEDT and NomBank relations, and therefore all the MTL models have two loss functions and two output layers. The two loss functions contribute equally (i.e. have the same weight) to computing the overall loss during training. We extend the STL model introduced in Chapter 6 to enable (hard) parameter sharing. Hence, we implement two MTL setups to learn NomBank and PCEDT:

1. Shared embeddings (MTL_{E}): a model with shared embedding layer but two task-specific hidden layers.

2. Fully shared (MTL_{EH}): apart from the output layer, the model does not have task-specific layers, i.e. both the embedding and hidden layers are shared. Figure 7.1 shows the architecture of the model.[7]

Like in transfer learning, we also distinguish between the auxiliary and main tasks in MTL, but the distinction is based on which development accuracy (NomBank or PCEDT) is monitored by the early stopping criterion. Hence, we end up with a total of four MTL configurations, based on which dataset (or task) is the main task and the architecture of the MTL model (i.e. MTL_{E} and MTL_{EH}).

[7] We refer to this model as MTL_{F} in Fares et al. (2018). The notation was changed here for clarity.

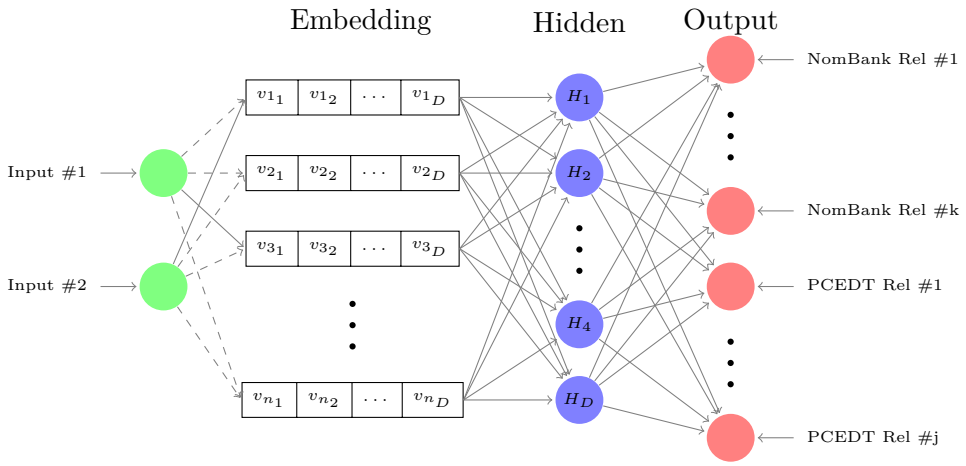


Figure 7.1: Architecture of the MTL_{EH} model

7.5 Experimental Results

In this section, we present and analyze the results of our experiments on TL and MTL. To allow comparison with the results presented in Chapter 6, we use the exact same data splits (i.e. training and development sets) to train and evaluate the models. We start by reporting results on the development split (§ 7.5.1), and then move to in-depth analysis of the STL, TL and MTL models' performance on the test split (§ 7.5.2).

Before we present the results, however, we recall the distribution of the most frequent relations in NomBank and PCEDT across the three data splits in Figures 7.3a and 7.2b. As we have repeatedly stated, and as the figures indicate, the imbalanced distribution of relations in NomBank and PCEDT renders accuracy *alone* a less informative evaluation measure to identify the best performing model in our experiments. Therefore, it is important to report and analyze the (macro-averaged) F_1 scores of the NomBank and PCEDT relations across all the STL, TL and MTL models. Furthermore, these figures demonstrate the difficulty of the problem at hand; for example, almost 71% of the relations in the NomBank training split are `ARG1` (proto-typical patient), and 52% of the PCEDT relations are of type `RSTR` (underspecified adnominal modifier). Such highly skewed distributions of the relations make learning some of the other relations more difficult, if not impossible in some cases. In fact, of the 11 NomBank relations observed in the development split, three relations are never predicted by any of the STL, TL and MTL models, and

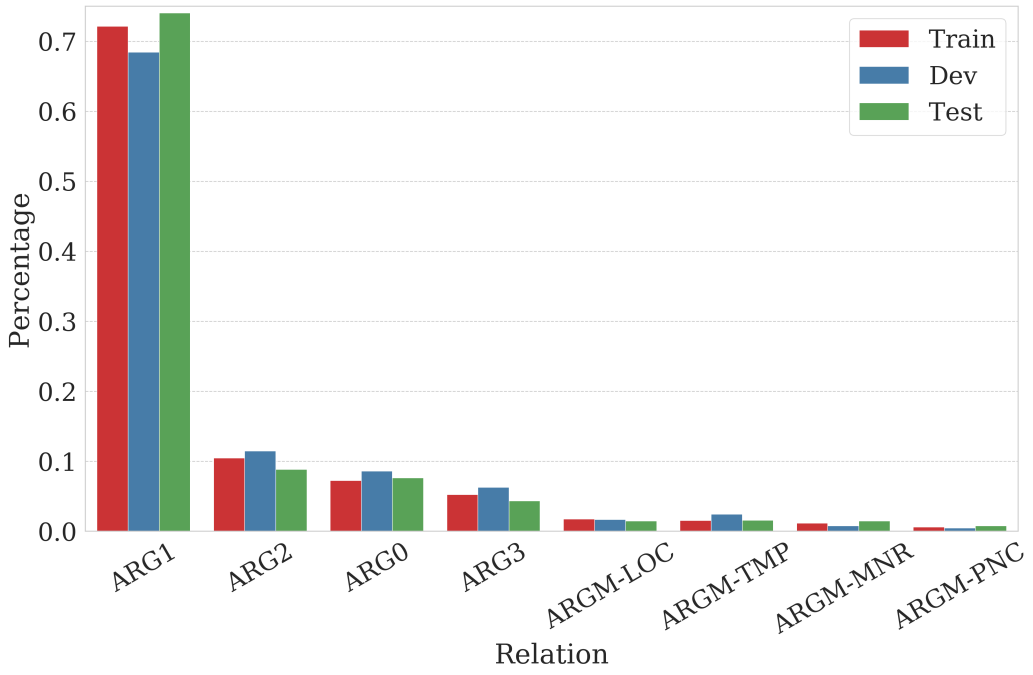
of the 20 PCEDT relations observed in the development split only six are predicted. That said, the non-predicted relations are extremely infrequent in the training set (e.g. 23 PCEDT functors each occurs less than 20 times in the training set), and it is therefore questionable if an ML approach will be able to learn them under any circumstances. The macro-averaged F_1 scores in this section are hence computed over the subset of 'learnable' relations in NomBank and PCEDT (cf. § 4.5.4 for the definition of the subset of 'learnable' relations in both datasets).

7.5.1 Evaluation on the Development Split

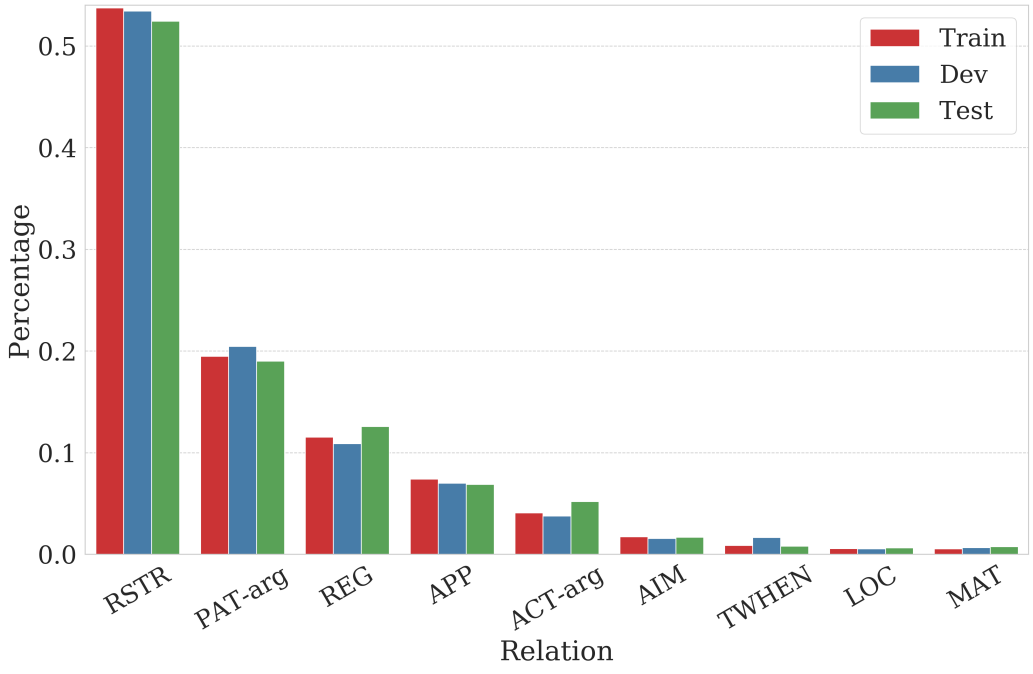
Tables 7.1a and 7.1b present the accuracy and macro-averaged F_1 scores of the different TL and MTL models on the development split in NomBank and PCEDT.[8] The top row in both tables shows the results of the STL model, which also serve as our baseline in the TL and MTL experiments. The NomBank column refers to the results when said dataset is used as a target task (in TL) or main task (in MTL), and PCEDT is used as a source or auxiliary task. For example, the second cell in the NomBank column refers to the model (TL_E) whose embedding layer is initialized using the embedding weights from the STL model trained on PCEDT. The same applies inversely to the PCEDT column.

Looking at the accuracy results first, we see that the benefits of TL and MTL vary depending on the dataset. For example, on PCEDT (i.e. predicting PCEDT functors as the target or main task), all the TL models improve the accuracy over the STL model, with the largest—though still modest—improvement achieved using the TL_{EH} model (1.09 absolute points). On NomBank, however, the best improvement we observe is 0.44, using the TL_{EH} model also. It is also important to highlight here that in terms of error reduction (i.e. $\frac{\Delta Error}{Error_{\text{STL}}} \times 100$), the 0.44 improvement in NomBank accuracy amounts to 2% error reduction, whereas the 1.09 increase on PCEDT corresponds to 2.64% error reduction. Both of the MTL models worsen the

[8]Note that the numbers reported in this chapter marginally differ from what we report in Fares et al. (2018). While the differences observed do not change the overall conclusion, it is important to highlight that those experiments were conducted on different computational infrastructures using different versions of Tensorflow. For example, the models in Fares et al. (2018) were trained using Tensorflow version 1.8.0, whereas the models in this chapter were trained using version 1.12.0. Crane (2018) identifies changing the framework version as one of the factors that can potentially cause "irreproducibility of results".



(a) NomBank



(b) PCEDT

Figure 7.2: Distribution of NomBank and PCEDT relations in the training, development and test sets.

Model	NomBank	PCEDT
STL	78.04	58.80
TL_E	78.37	59.57
TL_H	78.15	59.24
TL_{EH}	**78.48**	**59.89**
MTL_E	77.93	59.78
MTL_{EH}	76.74	58.80

(a) Accuracy

Model	NomBank	PCEDT
STL	58.92	37.10
TL_E	**59.36**	44.24
TL_H	59.08	42.74
TL_{EH}	59.12	**44.87**
MTL_E	59.09	43.55
MTL_{EH}	48.01	37.37

(b) Macro-average F_1 score

Table 7.1: Accuracy (left) and macro-averaged F_1 score (right) of the STL, TL and MTL models on the development splits of NomBank and PCEDT. TL_E: Transfer of the embedding weights. TL_H: Transfer of the hidden layer weights. TL_{EH}: Transfer of both the embedding and hidden layer weights. MTL_E: The embedding layer weights are shared. MTL_{EH}: Both the embedding and hidden layers weights are shared.

accuracy on NomBank, most notably in the case of the fully shared model MTL_{EH} (1.3 absolute points decrease in accuracy). The MTL accuracy results on PCEDT are better than on NomBank in the sense that MTL_E improves over the STL accuracy and MTL_{EH} leads to the same accuracy as the STL model.

Switching focus from accuracy to macro-averaged F_1 scores, Table 7.1b shows that all the TL models and the MTL_E one remarkably improve the PCEDT F_1. The results on NomBank are somehow in line with the accuracy results, as we either see modest improvement, or in the case of MTL_{EH} the macro-averaged F_1 drops (the NomBank accuracy also drops using the same model, but the drop in F_1 is much more dramatic). To further understand these results, we detail the per-relation F_1 scores on NomBank and PCEDT in Tables 7.2a and 7.2b, respectively. With the exception of the `ARGM-TMP` and `ARGM-LOC` relations, none of the NomBank relations improve by more than one point in F_1 score, and the F_1 scores of `ARG3` and `ARGM-MNR` remarkably decrease across all models. The MTL_{EH} model in particular leads to degraded performance on all the NomBank relations, except on `ARGM-TMP`. Moreover, the model no longer predicts the relation `ARGM-MNR`, as it has an F_1 score of 0 (the F_1 score in Table 7.2a decreases by 22.22 points), which largely explains the substantial drop in macro-averaged F_1 in Table 7.1b.

The PCEDT per-relation F_1 scores, in Table 7.2b, show a different pattern.

	A0	A1	A2	A3	LOC	MNR	TMP
Count	78	622	104	57	15	7	22
STL	58.90	87.46	50.00	67.27	47.62	22.22	78.95
TL_E	−2.49	+0.49	+0.27	−7.86	+10.71	−4.04	+6.05
TL_H	−2.32	+0.34	−0.53	−9.27	+6.93	−5.55	+11.53
TL_{EH}	+0.45	+0.47	+0.26	−8.68	+6.93	−6.84	+8.85
MTL_E	−3.06	−0.02	+0.54	−8.45	+8.38	−2.22	+6.05
MTL_{EH}	−7.92	−0.99	−10.84	−8.09	−35.12	−22.22	+8.85

(a) Per-relation F_1 score on the NomBank development split

	ACT	TWHEN	APP	PAT	REG	RSTR	AIM
Count	34	15	63	184	98	481	14
STL	48.39	43.48	31.91	46.10	19.05	70.76	0.0
TL_E	+4.07	+38.87	−3.02	+1.21	+8.92	−0.03	0.0
TL_H	−2.24	+38.87	−6.91	+3.56	+6.12	+0.07	0.0
TL_{EH}	+3.22	+38.87	−1.80	+1.96	+11.82	+0.30	0.0
MTL_E	+0.79	+38.87	−2.70	+0.83	+6.85	+0.50	0.0
MTL_{EH}	−0.77	+6.52	−7.73	−1.74	+5.19	+0.45	0.0

(b) Per-relation F_1 score on the PCEDT development split

Table 7.2: Per-relation F_1 score on the development split of NomBank and PCEDT. The numbers indicate the increase or decrease in F_1 score in comparison to the STL model.

From the table we see that the improvements in macro-averaged F_1 score on PCEDT (in Table 7.1b) are due to better performance on some of the less frequent relations. All the models that lead to increased F_1 score on PCEDT (i.e. all the TL models and MTL_E), achieve the same score as the STL model on the most common relation RSTR—and sometimes even better, e.g. TL_{EH}—while boosting the F_1 score on relations like REG, PAT and TWHEN. However, all the TL and MTL models lead to varyingly worse F_1 scores on APP. Perhaps more importantly, none of the STL, TL and MTL models predict the relation AIM; we will return to analyzing why this is the case in § 7.5.3.

In Chapters 4, we showed that there is likely some annotation inconsistency in PCEDT. In addition, in § 6.7, we observed that random initialization and the order of training examples have an impact on the PCEDT results.

Therefore, we need to verify if the improvements we see in the TL and MTL experiments actually hold when we compare them to the results of the experiments in § 6.7. Since `ACT`, `REG` and `TWHEN` are the three PCEDT relations that benefit most from TL and MTL, we compute their average F_1 scores in the 20 experiments with random initialization and order of the training example (which is the setup that led to highest standard deviation in accuracy in § 6.7). The average F_1 scores over 20 runs are as follows: 45.02% for `ACT`, 23.86% for `REG` and 61.32% `TWHEN`. While the average F_1 of the last two relations are higher than their scores in the STL model (cf. Table 7.2b), the improvement by the TL and MTL models still leads to higher F_1 scores on all three relations (compared to the averaged scores from the 20 random experiments).

Based on the results presented in this section, we can—for now—conclude that transfer and multi-task learning (with the exception of the MTL_{EH} model) tend to help more for PCEDT than NomBank. In other words, in absolute numbers, using NomBank as an auxiliary (or source) task for PCEDT is more effective than the other way around. However, one has to keep in mind that the performance levels on NomBank are higher than on PCEDT, which means that an improvement of one point on both datasets, for example, corresponds to different values of error reduction. In the following section, we evaluate our STL, TL and MTL models on the test splits of NomBank and PCEDT to, among other reasons, help us further investigate, and potentially explain, the patterns observed in this section.

7.5.2 Evaluation on the Test Split

We have thus far only evaluated our models on the development split of our compound datasets. As we now reach the final set of experiments in this thesis, we can safely evaluate our STL, TL and MTL models on the test split of NomBank and PCEDT. Overall, we believe that conducting evaluation and result analysis on the test split is more rigorous than on the development set for several reasons. First and foremost, our test split is almost twice the size of the development one (920 examples in development vs. 1,759 in test). Second, all of our neural models use an early stopping criterion that monitors the model's performance on the development split; and while using such a stopping criterion can help prevent overfitting on the training data, in the end we still choose a model that achieves the best accuracy on the development

Model	NomBank		PCEDT	
	Dev	Test	Dev	Test
STL	78.04	76.75	58.80	56.05
TL_E	78.37	78.05	59.57	**57.36**
TL_H	78.15	78.00	59.24	56.56
TL_{EH}	**78.48**	77.94	**59.89**	56.68
MTL_E	77.93	78.45	59.78	56.90
MTL_{EH}	76.74	**78.51**	58.80	56.00

(a) Accuracy

Model	NomBank		PCEDT	
	Dev	Test	Dev	Test
STL	58.92	52.54	37.10	34.42
TL_E	**59.36**	52.83	44.24	**41.42**
TL_H	59.08	53.03	42.74	39.90
TL_{EH}	59.12	53.11	**44.87**	40.44
MTL_E	59.09	**53.21**	43.55	40.47
MTL_{EH}	48.01	42.07	37.37	34.89

(b) Macro-averaged F_1

Table 7.3: Accuracy and macro-averaged F_1 score of the STL, TL and MTL models on the development and test splits of NomBank and PCEDT.

split. In addition, since our development split is relatively small in size, there is no guarantee that it is in fact representative of the problem. The last two points might cast some doubt on the benefits of using an early stopping criterion based on the development split (Prechelt, 2012). In other words, it is not unlikely that our stopping criterion favors a model that performs well on the development data but that is not necessarily the best model overall. For all the reasons above, in this section and the following ones we use the test split of NomBank and PCEDT to evaluate our models and systematically analyze their performance based on insights from the dataset as well as the classification errors of the models. Note that all the models remain the same as in the previous section, i.e. they are still only trained on the training split.

The accuracy and macro-averaged F_1 scores of the STL, TL and MTL models on the test split are shown in Tables 7.3a and 7.3b. We repeat the results on the development split to make it easier to compare the performance

of the models on the two data splits. There are several observations one can draw from these tables. First, the accuracy and macro-averaged F_1 scores of the *STL models* drop when the models are evaluated on the test split, whether on NomBank or PCEDT, in comparison to their accuracy and F_1 on the development split. The same observation holds for the development-versus-test accuracy and F_1 for all the TL and MTL models on PCEDT. On NomBank, however, most TL and MTL models achieve more or less the same accuracy on development and test, and some are even better on test (e.g. MTL_E and MTL_{EH}), but the F_1 scores drop on test (in comparison to their scores on development). The overall drop in accuracy and F_1 moving from development to test can be interpreted as an indicator of 'overfitting' on the development set. Second, all the TL models outperform the STL model on the test split of NomBank, even though transfer learning does not remarkably improve accuracy over STL on the development split of the same dataset (cf. § 7.5.1). Furthermore, while the MTL models, especially MTL_{EH}, have a negative effect on accuracy on the development split of NomBank, they still lead to the same improvement as the TL models on the test split. Third, transfer learning improves the test accuracy on PCEDT by about 1.31 absolute points (in comparison to STL), which is more or less the same effect observed on the development split. However, the TL model that leads to this improvement on test (TL_E) is different from the one (TL_{EH}) that leads to a 1.09 improvement on the development split. The fully shared multi-task learning model (MTL_{EH}) yields slightly worse PCEDT accuracy on test than the STL model; which is also somewhat in line with what we observe on the development split, where the MTL_{EH} model does not lead to any improvement in accuracy.

All in all, the PCEDT results on test are more or less similar to the results we see on development. Both the TL and MTL models improve accuracy on the test split of NomBank by at least 1.19 absolute points. That said, the NomBank results on development versus test are somehow inconsistent; the model that achieves the best accuracy on development (TL_{EH}) records the worst accuracy on test among the TL and MTL models, and the model that achieves the best accuracy on test has the worst accuracy on development (MTL_{EH}) among all other models. To understand this effect, we—once again—turn to study the macro-averaged F_1 scores.[9]

[9]Note that the macro-averaged F_1 scores in Table 7.3b differ from what we report in Table 8 in Fares et al. (2018) because in the latter we computed macro-averaged on a

From Table 7.3b, we see that the best model on NomBank test in terms of accuracy, MTL_{EH}, is in fact dramatically worse than all other models w.r.t. its macro-averaged F_1 score. The rest of the TL and MTL models improve over the F_1 score of the STL model, though these improvements are relatively small, with the largest being 0.67 by the MTL_{E} model (which also achieves the best accuracy if we exclude MTL_{EH}). The macro-averaged F_1 results on PCEDT correspond to the accuracy results, where the model that achieves the best accuracy on test also achieves the best F_1 score (TL_{E}). Furthermore, the relative ranking of the models based on their accuracy and F_1 scores is highly similar, with the exception of the MTL_{EH} model which achieves slightly higher F_1 than the STL model.

Based on the F_1 scores in Table 7.3b, it becomes quite clear that TL and MTL on the embedding layer yield remarkable improvements on PCEDT, with about 7 absolute points increase in F_1 in contrast to 0.67 in the best case on NomBank. From an error reduction perspective, the improvements on the PCEDT and NomBank macro-averaged F_1 scores correspond to 10.67% and 1.41%, respectively.

Lastly, even though we see remarkable improvements in macro-averaged F_1 score on PCEDT, the improvements in accuracy remain relatively small. In addition, the accuracy results on the development and test splits of NomBank are somewhat contradictory (in terms of which model is the best and which one is the worst across the development and test splits). In the following section, therefore, we take a closer look at the per-relation F_1 scores to further our understanding of how the two evaluation metrics interact, and ultimately when, and how, TL and MTL help.

Per-Relation F_1 Scores

Tables 7.4a and 7.4b show the per-relation F_1 scores on the test split of NomBank and PCEDT, respectively. Like the previous section, we only include the results for the subset of relations deemed 'learnable' based on their frequency in the training split (cf. § 4.5.4), which is the same subset of relations used to compute the macro-averaged F_1 score.[10]

different subset of PCEDT relations that excludes the relation `AIM` (which is not predicted by any of the models). The overall ranking of the models remains the same nonetheless.

[10]Five of the NomBank relations observed in the test split are never predicted by any of the STL, TL and MTL models, and of the 26 PCEDT relations observed in the test split only six are predicted (one of the presumably 'learnable' relations in PCEDT is never

	A0	A1	A2	A3	LOC	MNR	TMP
Count	132	1282	153	75	25	25	27
STL	49.65	87.41	45.65	60.40	28.57	29.41	66.67
TL_E	+5.37	+0.57	−4.04	−0.26	−0.66	+3.92	−2.84
TL_H	+5.16	+0.52	−2.74	−0.40	−3.57	+5.88	−1.36
TL_{EH}	+3.97	+0.54	−2.95	+0.71	+0.70	+3.92	−1.45
MTL_E	+4.42	+0.93	−2.79	+1.57	+1.43	−0.84	0.0
MTL_{EH}	+3.44	+1.00	−7.51	+2.29	−28.57	−29.41	−14.5

(a) Per-relation F_1 score on the NomBank test split

	ACT	TWHEN	APP	PAT	REG	RSTR	AIM
Count	89	14	118	326	216	900	29
STL	43.90	42.11	22.78	42.83	20.51	68.81	0.0
TL_E	+5.47	+28.86	+4.89	−1.31	+10.26	+0.80	0.0
TL_H	+10.09	+19.96	+2.22	+0.26	+5.58	+0.24	0.0
TL_{EH}	+5.93	+22.41	+5.79	+0.08	+8.15	+0.21	0.0
MTL_E	+10.19	+24.56	+1.27	−0.72	+6.61	+0.41	0.0
MTL_{EH}	+3.90	0.0	+2.86	−2.19	−1.29	+0.02	0.0

(b) Per-relation F_1 score on the PCEDT test split

Table 7.4: Per-relation F_1 score of the STL, TL and MTL models on the test split of NomBank and PCEDT. The numbers indicate the increase or decrease in F_1 score in comparison to the baseline model (STL).

We observe several interesting patterns in Tables 7.4a and 7.4b. First, based on our analysis of the MTL_{EH} results on the development split as well as its accuracy and macro-averaged F_1 score on test, we know by now that despite its relatively high accuracy the model does not perform well on the less frequent relations. We notice the same pattern on the test split; MTL_{EH} leads to substantially degraded F_1 scores on four NomBank relations, including the locative modifier `ARGM-LOC` and manner modifier `ARGM-MNR` (shortened to `LOC` and `MNR` in Table 7.4a) which the model is no longer able to predict. The reason that the MTL_{EH} model achieves the highest accuracy on the NomBank test split (cf. Table 7.3a) is partly because it correctly predicts more compounds of type `ARG1`, which is the most frequent relation in NomBank—in fact, it achieves the highest improvement on `ARG1` (and `ARG3`) F_1 score in comparison to all other TL and MTL models. The same model, MTL_{EH}, also has the worst F_1 score, compared to all other models, for two PCEDT relations, namely `REG` (which expresses a circumstance) and `PAT` (Patient).

Second, with the exception of the MTL_{EH} model, all the TL and MTL models consistently improve the F_1 score of all the PCEDT relations except `PAT` (and `AIM` which is not predicted by any of the models). Most notably, the F_1 scores of the relations `TWHEN`, `REG` and `ACT` see a remarkable boost, compared to other PCEDT relations, when the embedding layer's weights are shared (MTL_{E}) or transfered (TL_{E} and TL_{EH}). The improvement on `TWHEN` and `ACT` can be partly explained by looking at the correspondence matrices between NomBank arguments and PCEDT functors. In Tables 7.5b and 7.5a, we present a compact version of the correspondence matrices (introduced in § 4.5.2); the tables show how the PCEDT functors map to NomBank arguments in the training split (Table 7.5a) and the other way around (Table 7.5b). From Table 7.5a, we see that 80% of the compounds annotated as `TWHEN` in PCEDT were annotated as `ARGM-TMP` in NomBank. Inspecting the predictions of the STL and TL_{E} models, we find that all the `TWHEN` examples that the latter model predicts correctly, but the STL one does not, are indeed annotated as `ARGM-TMP` in NomBank; for example the compound *future power* is annotated as `ARGM-TMP` and `TWHEN` in NomBank and PCEDT, respectively, and the STL model misclassifies it as `RSTR` whereas the TL_{E} model predicts the correct relation. In addition, 47% of the `ACT`

predicted, viz. `AIM`).

relations map to ARG0 (proto-agent) in NomBank; even though this mapping is not as clear as one might have expected, it is still relatively high if we consider how other PCEDT relations map to ARG0. The correspondence matrices demonstrate how the assumed theoretical similarities between the NomBank and PCEDT relations do not always hold as clearly, e.g. considering the mapping from the proto-typical agent and patient relations in NomBank, ARG0 and ARG1, to ACT and PAT in PCEDT in Table 7.5b. Nonetheless, even such 'imperfect' correspondence seems to provide a 'training signal' that helps the TL and MTL models learn relations such as ACT.

Since the TL_E model outperforms the STL model in predicting REG by ten absolute points, we inspect all the REG compounds that are correctly classified by the TL_E model but are misclassified by the STL model. We find that the latter always misclassifies them as RSTR which indicates that transfer learning from NomBank helps the TL_E model recover from the STL's over-generalization in RSTR prediction (the most frequent relation in PCEDT).

The two NomBank relations that receive the highest boost in F_1 scores (about five absolute points) are ARG0 and ARGM-MNR, but the improvement in the latter relation, actually, corresponds to only one more compound which might well be predicted correctly by chance. To analyze the improvement in ARG0, we look again at the correspondence matrix between PCEDT and NomBank in Tables 7.5a and 7.5b. The PCEDT functor ACT maps to ARG0 47% of the time, and 26% the other way around, which can partially explain the relative boost observed in ARG0.

Overall, the (macro-averaged) F_1 scores indicate that TL and MTL from NomBank to PCEDT are more helpful than from PCEDT to NomBank. The correspondence matrices in Tables 7.5a and 7.5b can offer an insight on why this might be the case. In the first rows of the aforementioned tables, we see that five PCEDT relations (including the four most frequent ones) map to ARG1 in NomBank in more than 60% of the time for each relation. This means that the weights learned to predict PCEDT relations potentially offer little or no inductive bias for NomBank relations. Whereas if we consider the mapping from NomBank to PCEDT, we see that even though many NomBank arguments map to RSTR in PCEDT, the percentages are comparatively lower, and hence the mapping is more 'diverse' (i.e. discriminative) which seems to help the TL and MTL models learn the less frequent PCEDT relations.

	RSTR	PAT	REG	APP	ACT	AIM	TWHEN
A1	0.70	0.90	0.78	0.62	0.47	0.65	0.10
A2	0.11	0.05	0.10	0.21	0.03	0.12	0.03
A0	0.06	0.01	0.04	0.13	0.47	0.07	-
A3	0.06	0.02	0.06	0.02	0.01	0.06	-
LOC	0.02	0.01	0.00	0.01	0.01	0.01	0.02
TMP	0.01	-	0.00	0.00	-	-	0.80
MNR	0.02	0.00	0.00	-	0.01	-	-
Count	3617	1312	777	499	273	116	59

(a) Correspondence matrix between PCEDT functors and NomBank arguments and modifiers

	A1	A2	A0	A3	LOC	TMP	MNR
RSTR	0.51	0.54	0.47	0.63	0.66	0.36	0.78
PAT	0.24	0.09	0.03	0.08	0.07	-	0.05
REG	0.12	0.11	0.07	0.13	0.02	0.01	0.01
APP	0.06	0.14	0.13	0.03	0.05	0.01	-
ACT	0.03	0.01	0.26	0.01	0.03	-	0.03
AIM	0.02	0.02	0.02	0.02	0.01	-	-
TWHEN	0.00	0.00	-	-	0.01	0.46	-
Count	4932	715	495	358	119	103	79

(b) Correspondence matrix between NomBank arguments and modifiers and PCEDT functors

Table 7.5: Correspondence matrix between (a) PCEDT and NomBank and (b) NomBank and PCEDT. Slots with '-' mean zero, 0.00 is a very small number but not zero.

7.5.3 What Happened to AIM?

As mentioned in the previous sections, none of the STL, TL and MTL models predicts the functor `AIM` in the development and test splits of PCEDT, even though it is more frequent than `TWHEN` in the training split (cf. Figure 7.2b). Upon inspecting the classification errors of all the models on the test split, we find that `AIM` is almost always misclassified as `RSTR` by all the models.

Furthermore, through analyzing the training data, we discover that the relations `AIM` and `RSTR` have the highest lexical overlap in the training set among all other pairs of relations in PCEDT: 78.35% of the modifier nouns (or left constituents) and 73.26% of the head nouns (or right constituents) of the compounds annotated as `AIM` occur in other compounds annotated as `RSTR`. This at least in part explains why none of the models manage to learn the relation `AIM` but raises a question about the models' ability to learn relational representations; we further pursue this question in the following section.

7.6 Generalization on Unseen Compounds

We now turn to analyze the models' ability to generalize over compounds unseen in the training split. Recent work by Dima (2016) and Shwartz and Waterson (2018) suggests that the gains achieved in compound interpretation using word embeddings and somewhat similar neural classification models are in fact a by-product of a phenomenon called *lexical memorization*. Levy, Remus, et al. (2015) define lexical memorization as the phenomenon in which the classifier learns that a specific word in a specific slot is a strong indicator of the label. In other words, the classification models may merely learn that a specific set of nouns is a strong indicator of a specific relation in NomBank or PCEDT. Therefore, in order to gauge the role of lexical memorization in our models also, we quantify the number of unseen compounds that the STL, TL and MTL models predict correctly.

The term 'generalization' in this section refers to the model's ability to correctly predict the relations of nominal compounds whose constituents are partly or completely unseen in the training data (i.e. the second definition of generalization in § 7.2 above). We distinguish between 'partly' and 'completely' unseen compounds as follows. A compound is considered 'partly' unseen if one of its constituents (right or left) is not seen in the training data at all. A

Model	NomBank			PCEDT		
	L	R	L&R	L	R	L&R
Count	351	286	72	351	286	72
STL	27.92	39.51	50.00	45.01	47.55	41.67
TL_E	25.93	36.71	48.61	**43.87**	47.55	41.67
TL_H	26.21	38.11	50.00	46.15	49.30	47.22
TL_{EH}	26.50	38.81	52.78	45.87	47.55	43.06
MTL_E	24.50	**33.22**	**38.89**	44.44	**47.20**	43.06
MTL_{EH}	**22.79**	34.27	40.28	44.16	47.90	**38.89**

Table 7.6: Generalization errors on the subset of unseen compounds from the test set. L: Left constituent is unseen; R: Right constituent is unseen; L&R: Left and right constituents are unseen.

completely unseen compound is one whose left *and* right constituent are not seen in the training data (i.e. completely unseen compounds are the subset of compounds in the test split that have zero lexical overlap with the training split).[11] Overall, almost 20% of the compounds in the test split have an unseen left constituent, about 16% of the compounds have an unseen right constituent and 4% are completely unseen. In Table 7.6, we compare the performance of the different models on these three groups in terms of the proportion of compounds a model *misclassifies* in each group (i.e. error rate).

From Table 7.6, we see that TL and MTL reduce the NomBank generalization error in all cases, except TL_H and TL_{EH} on completely unseen compounds; the latter (TL_{EH}) leads to higher generalization error. The MTL models lead to the biggest error reduction across the three types of unseen compounds; MTL_E leads to about six points error reduction on compounds with unseen right constituents and eleven points on completely unseen ones, and MTL_{EH} reduces the error on unseen left constituent by five points. Note,

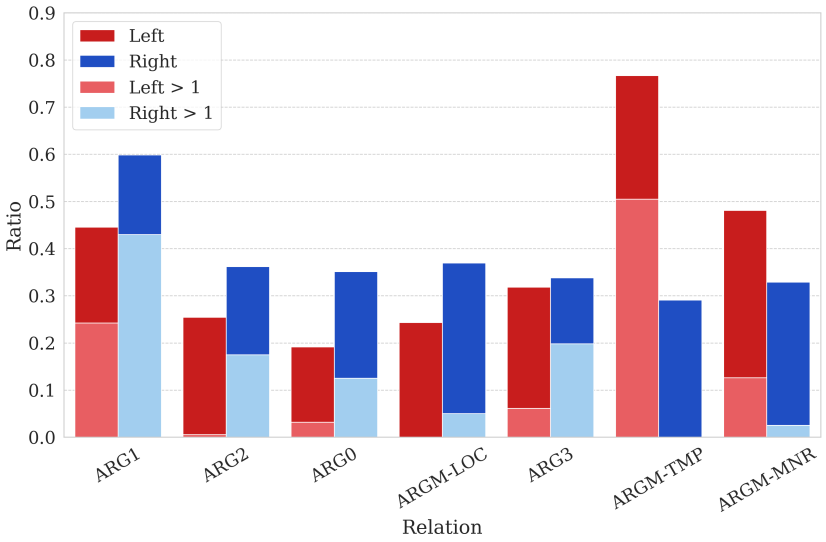
[11] Of course, all the compounds in the development and test splits are 'unseen' as a whole, given that our compound datasets are type-based (i.e. there are no duplicates). However, the criterion for defining unseen compounds here is based on the compound's constituents. It is, therefore, possible to have compounds in the test splits whose constituents occur in the training split, but with different compounds. For example, if the compounds *account manager* and *credit account* occur in the training split, none of them can occur in test. However, a compound like *credit manager* can still occur in test. We do not consider this compound 'unseen' (neither partly nor completely) because both of its constituents occur in training, albeit as part of different compounds.

however, that these results have to be read together with the *Count* row in Table 7.6 to get a complete picture. For instance, an eleven-point decrease in error on completely unseen compounds amounts to eight compounds. In PCEDT, the largest error reduction on unseen left constituents is 1.14 points which amounts to four compounds, 0.35 (just one compound) on unseen right constituents and 2.7 (or two compounds) on completely unseen compounds.

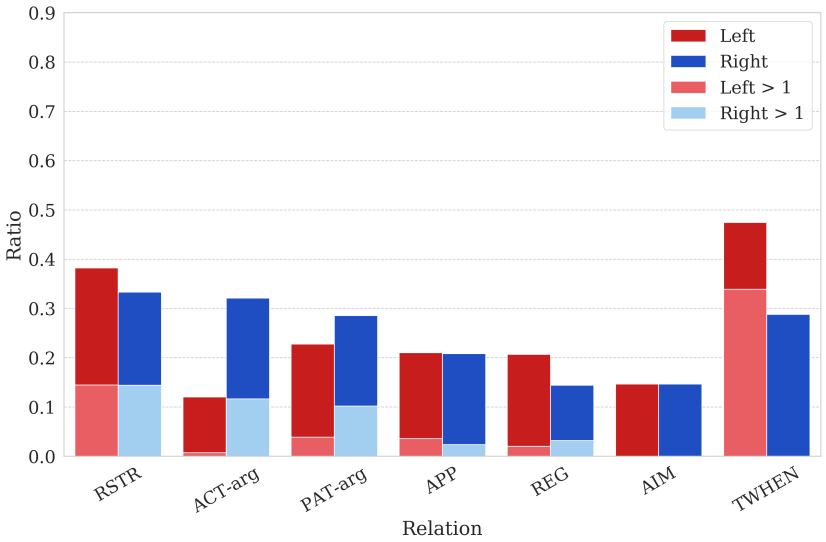
Since we see larger reductions in the generalization error in NomBank, we manually inspect the compounds that led to these reductions; i.e. we inspect the distribution of relations in the set of the correctly predicted unseen compounds. The MTL_E model reduces the generalization error on completely unseen compounds by a total of eight compounds compared to the STL model, but seven of these compounds are annotated with `ARG1`, which is not unexpected given that `ARG1` is the most frequent relation in NomBank.[12] When it comes to the unseen right constituents, the 24 compounds that MTL_E improves on consist of 18 `ARG1` compounds, five `ARG0` compounds and one `ARG2` compound. We see a similar pattern upon inspecting the gains of the TL_E model; where most of the improvement arises from predicting more `ARG1` and `ARG0` relations correctly. The majority of the partly or completely unseen compounds that were misclassified by all models are *not* of type `ARG1` in NomBank or `RSTR` in PCEDT. This observation in and of itself is in line with the overall pattern we have seen so far, i.e. less frequent relations are, obviously, more difficult to learn. What remains to be determined, nonetheless, is the role lexical memorization plays in learning the interpretation of noun–noun compounds.

To gauge the effect of lexical memorization, we study the ratio of relation-specific constituents in NomBank and PCEDT, plotted in Figure 7.3. We define relation-specific constituents as left or right constituents that only occur in compounds of the same relation in the training split, and their ratio is simply their proportion in the overall set of left or right constituents per relation. Some constituents, however, occur only once in the training split which obviously makes them relation-specific, but that can also make their

[12]Note that these eight compounds do not fully represent the difference in predictions between the MTL_E model and the STL model, because the MTL_E predictions introduce a new error on one compound of type `ARGM-PNC`, but compensate by predicting another one correctly. In other words, there is a total of nine differences between the two models on completely unseen compounds. However, since we are only discussing the set of unseen compounds the TL and MTL models predict correctly, in contrast to the STL model, we will leave out these details in order to keep the discussion tractable.



(a) NomBank



(b) PCEDT

Figure 7.3: Ratio of relation-specific constituents in (a) NomBank and (b) PCEDT. The bars in light red and light blue correspond to the ratio of relation-specific constituents whose frequency is greater than one in the training split.

corresponding relations more difficult to learn; i.e., more lexical 'diversity' in a given relation means less 'prototypical' examples, which can make learning more challenging. Hence, in Figure 7.3 we plot the ratio of relation-specific constituents overall (regardless of their frequencies) as well as the ratio of relation-specific constituents that occur more than once.

Looking at Figure 7.3, we see that NomBank relations have higher ratios of relation-specific constituents in comparison to PCEDT, even when we only consider constituents that occur more than once. This arguably makes learning the former comparatively easier if the model is only to rely on lexical memorization. Furthermore, `ARGM-TMP` in NomBank and `TWHEN` in PCEDT stand out from other relations in Figure 7.3, which are also the two relations with the second highest F_1 scores in their respective dataset—except in STL on PCEDT (cf. Tables 7.4a and 7.4b). Lexical memorization is, therefore, the most likely explanation of such relatively high F_1 scores. We also observe some correlation between lower ratios of relation-specific constituents and relatively low F_1 scores, e.g. `APP` and `REG` in PCEDT. Interestingly, all the constituents specific to the PCEDT relation `AIM` occur only once in the training split (cf. Figure 7.3 where `AIM` does not have bars in light red or light blue), which adds to the reasons why this particular relation was not predicted by any of the STL, TL and MTL models (cf. § 7.5.3). Based on these observations, we cannot rule out that our models exhibit some degree of lexical memorization effects, even though manual result analysis has also revealed 'counter-examples' where the models generalize and make correct predictions where lexical memorization is impossible.

7.7 Performance Stability

In Chapter 6, we observed that the model's performance (in terms of accuracy) depends to some extent on the random initialization and the order in which the training examples are presented. In this section, we study whether or not the same effect holds in the TL and MTL realm. However, since we have six TL models and four MTL models, we choose only the two models that allow maximal parameter transfer and sharing (i.e. TL_{EH} and MTL_{EH}) to measure their performance stability with regard to random initialization (using different random seeds) and order of the training data. We follow the same experimental setup introduced in § 6.7, where we try to isolate the

	PCEDT
Min	58.15
Max	59.35
Mean	58.75
STD	0.35

Table 7.7: Minimum, maximum, mean and standard deviation (STD) of TL_{EH} accuracy over 20 runs with 20 different random seeds. The training examples are presented in the same order across the 20 runs.

effect of parameter random initialization and data shuffling, using 20 different random seeds (which are the exact same random seeds used in § 6.7).

In the first experiment, we try to quantify the effect of random initialization on the performance of the TL_{EH} model. We train 20 STL models on NomBank (using the same 20 random seeds from § 6.7) and we then use the weights of these models to initialize 20 TL_{EH} models for PCEDT. In other words, we have 20 pairs of source (NomBank) and target (PCEDT) models and each pair is initialized using one of the 20 random seeds. Note that the random seeds are only used to initialize the model weights, but the training examples are presented in the same order across all the experiments. Table 7.7 presents the maximum, minimum and average accuracy as well as the standard deviation of the 20 runs on the development split of PCEDT. The results in the table do not support our intuition about the benefits of transfer learning as a more 'informed' parameter initialization strategy (cf. § 7.1). The standard deviation observed on the TL_{EH} models (0.35) is in fact a little higher than what we report in Table 6.14 (0.29). The mean accuracy of the TL_{EH} models is only slightly higher than the mean accuracy reported in Table 6.14, the former being 58.75 and the latter 58.54.

In the second experiment, we focus on the effect of the order of training examples on the performance of the MTL_{EH} model. We train 20 MTL_{EH} models where PCEDT is the main task, also using the same 20 random seeds as in the previous experiment (and § 6.7). Unlike the previous experiment, however, these random seeds are used throughout training including the shuffling of training examples. The results of these experiments on the development split of PCEDT are listed in Table 7.8. The standard deviation of the MTL_{EH} models' accuracy is a little lower than for the STL models

	PCEDT
Min	57.06
Max	61.30
Mean	59.26
STD	0.95

Table 7.8: Minimum, maximum, mean and standard deviation (STD) of MTL_{EH} accuracy over 20 runs with 20 different random seeds. The training examples are shuffled differently across the 20 runs.

(1.10, cf. Table 6.13) but it is still relatively high overall. The average accuracy of the MTL_{EH} models is also only marginally higher than for the STL models in the previous chapter (59.26 vs. 59.06).

Given the results in Tables 7.7 and 7.8, we can conclude that TL and MTL do not necessarily help improve the model's stability in terms of its accuracy on the development split. While this conclusion contradicts our intuition about the benefits of transfer learning as a parameter initialization strategy, it is also likely that our hyperparameter choices partly led to those results. In § 7.4.1, we explained that using the exact same hyperparameters for the STL model, on the one hand, and the TL and MTL models, on the other hand, would allow isolating the effects of TL and MTL as much as possible. However, there is a flip side of this decision; for example, by using the same optimization function (viz. Adam) without lowering the learning rate we risk overwriting the 'knowledge' (i.e. weights) learned in the source task. We, therefore, believe that further experimentation with different hyperparameters specific to TL and MTL can still potentially yield different results in terms of the model's performance stability.

7.8 Conclusion

We presented in this chapter a relatively comprehensive series of experiments on two learning strategies, transfer and multi-task learning, that currently receive a lot of attention in NLP and other fields. Despite this attention, there remains considerable uncertainty about which task properties and experimental settings actually make such learning methods effective. In addition to our immediate goal of learning compound interpretation in the

context of NomBank and PCEDT, our experiments shed light on the utility of TL and MTL perspectives on the semantic interpretation of noun–noun compounds, which has not been investigated before in this realm.

In § 7.2, we gave a brief introduction to transfer and multi-task learning, established some terminology and reviewed a selection of relevant NLP studies. Then, in § 7.3, we explained our hypothesis for why TL and MTL could be beneficial in the context of our NomBank and PCEDT datasets. We introduced our experimental setup in § 7.4, which consists of six TL models and four MTL models. In § 7.5, we reported the results of a series of minimally contrasting experiments using the TL and MTL models. The accuracy and macro-averaged F_1 scores we reported explore an elaborate space of distinct experimental configurations, and therefore we only present a high-level summary in the following.

Through our experiments and in-depth analysis of results and prediction errors on both the development and test splits, we demonstrated the ability of both TL and MTL to mitigate the challenges of class imbalance and substantially improve prediction of low-frequency relations, when using (the harder) PCEDT as the target (in TL) or main (in MTL) task in particular. More specifically, we showed that transfer of representations or sharing across our tasks (or datasets) is most effective at the embedding layers, i.e. the model-internal representation of the two compound constituents involved. In multi-task learning, full sharing of the model architecture across tasks (i.e. the MTL_{EH} model) dramatically worsens the model's ability to generalize on the less frequent relations in both NomBank and PCEDT. Moving from evaluation on the development split to the held-out test data in § 7.5.2, we noticed a drop in the accuracy of the STL model (i.e. the model from Chapter 6) on both datasets, but TL and MTL helped bring up the test accuracy, especially on NomBank. One recurring pattern throughout our experiments was that the accuracy and macro-averaged F_1 scores tend to sometimes paint different pictures. For example, we observed that the model (MTL_{EH}) that achieved the best test accuracy on NomBank is also the model with the worst macro-averaged F_1 score on the same split. This is not an exceptional observation per se—given the skewed distribution of relations in NomBank and PCEDT—but it serves as a reminder of the importance of methodological decisions when reporting empirical results on datasets like ours.

In § 7.6, we showed that our TL and in particular MTL models made quantitatively and qualitatively better predictions, especially so on the 'hard-

est' inputs involving at least one constituent not seen in the training data. However, indicators of remaining 'lexical memorization' effects arise from our error analysis of unseen compounds as well as the ratio of relation-specific constituents in Figure 7.3.

Finally, given the results of § 6.7, we sought to determine whether or not transfer and multi-task learning can help improve the 'stability' of our models on the PCEDT dataset. However, contrary to our preliminary intuition about TL as an initialization strategy, the performance stability experiments—in § 7.7—unveiled that the final performance of our TL (and MTL) models remains somewhat dependent on the order of the training data (i.e. data shuffling). That said, this conclusion can also be a by-product of the hyperparameters and settings in our experiments, which were deliberately *not* fine-tuned to allow direct comparison with the STL experiments.

Overall, the experiments in this chapter demonstrated how our NomBank and PCEDT datasets present an interesting opportunity for innovative neural approaches to compound interpretation, as they relate this sub-problem to broad-coverage semantic role labeling or semantic dependency parsing in PCEDT and NomBank. In the following, and last, chapter we will discuss, inter alia, potential avenues for future work and improvement on our TL and MTL setup.

Chapter 8

Summary and Concluding Remarks

In this thesis, we set out to unravel some of the essential questions surrounding noun–noun compound analysis in NLP, while weaving the problem into a more holistic perspective on meaning representation. We argued for a new approach to compound interpretation, created a new dataset that enables such an approach and conducted a thorough empirical study using this dataset.

Our approach to compound analysis took a 'reverse' view on the problem; that is, instead of starting with a compound-centric perspective, where the datasets and taxonomies are carefully crafted for the sake of compound interpretation only, we sought to determine how noun–noun compounds are analyzed when they are part of broader meaning representation frameworks. Though this approach might be considered unorthodox in contrast to previous studies, it is indeed the disagreement among past studies (both in NLP and theoretical linguistics) that partly motivated our perspective. In our theoretical and literature reviews, in Chapters 2 and 3, we explained how past studies presented starkly different views on the semantic relations of noun–noun compounds—at times, ranging from an inventory of only nine relations to 'infinity' (cf. § 2.1.2). We therefore took a somewhat pragmatic decision to delegate the definition of the compound relations to well-established resources like NomBank and PCEDT, which also grant straightforward integration of compound analysis with other NLP tasks.

In the following, we summarize some of the main results of our work and provide an outlook for future research.

Compound Identification and Dataset Creation

Aiming for a high-quality dataset of compounds, in Chapter 4, we revisited the task of compound identification as the first step in compound analysis. In § 4.3.3, we proposed an identification method that relies on syntactic structure to identify valid compounds and addressed some of the issues of PoS-based identification methods that were used in past studies. By exploiting the syntactic trees in the PTB, we were able to exclude invalid 'compounds' whose constituents are dominated by different parent nodes—which is a typical source of errors in the PoS-based method. Quantifying the accuracy of compound identification methods proved challenging due to the lack of gold-standard evaluation data. We therefore opted for manual inspection of a subset of the compounds identified by the PoS- and syntax-based methods (cf. § 4.3.4). We found that our proposed method successfully excludes the false positive examples identified by the PoS-based method (at the very least on the subset of compounds we inspected). We also observed, however, that nominal coordinate structures in principle would call for linguistic representations that make explicit the distinction between joint and distributive coordination structures.

In Chapter 4, we introduced a relatively large, multi-layered dataset of noun–noun compounds derived from: (1) the WSJ corpus in the Penn Treebank, (2) the annotation of noun phrases in the PTB by Vadas and Curran (2007), (3) DeepBank, (4) the English part of the PCEDT and (5) NomBank. Through this dataset, we showed that even though these five resources were not created to annotate compounds exclusively, they can still be exploited to analyze noun–noun compounds—and through doing so, to reflect on the resources themselves. For example, in § 4.4.1, we evaluated the cross-framework annotation agreement and found that PCEDT, DeepBank and Vadas and Curran (2007) agree on the bracketing of almost three quarters of the multi-word compounds we identified in the WSJ corpus of the PTB. To date, our dataset is the second largest available in terms of the number of binary compound types (viz. 10,596). However, what makes our dataset unique is indeed its triple syntactic and dual semantic annotations of compounds.

Each compound in our dataset is annotated with two relations based on the argument structure of nominal predicates in NomBank and the tectogrammatical layer in PCEDT. The highly imbalanced distribution of relations in our dataset might be perceived as a disadvantage, and it is indeed legitimate

to ask if such a distribution is a by-product of the underlying resources not being exclusively concerned with compounds. The flip side of this same question, however, is whether the compound-specific datasets are constructed in such a way that guarantees an artificially balanced representation of the compound relations, i.e. to what degree their distributions reflect those in running text. Even though we did not pursue an answer for this question, we did embrace our dataset as-is and addressed the problem of skewed label distributions by using appropriate evaluation measures (i.e. macro-averaged F_1) and machine learning strategies like transfer and multi-task learning (in Chapter 7).

Unlike the bracketing annotations, the semantic relations are not directly comparable across NomBank and PCEDT, and therefore we analyzed them based on our postulation of which relations are semantically comparable. Even though we observed partial agreement across the two frameworks (in the relations we believed to express the same concepts), it was not obvious whether and to what degree 'complete' correspondence could be established between the NomBank and PCEDT relations. We further investigated the utility of such cross-framework agreement in our empirical study in Chapter 7. This contrastive analysis gave rise to the question of token vs. type semantic interpretation of compounds. Considering the token-based version of our dataset (in which a compound type can occur more than once), we found that 13% of the compound types in PCEDT are annotated with more than one relation. We observed the same pattern on 1.3% of the compound types in NomBank. The variation in PCEDT relations (per compound type) seemed to be, at least in large parts, a result of annotation inconsistency, rather than a deliberate decision to distinguish between the meaning of the compound types across their different occurrences. In NomBank, however, we found a few examples that would call for a token-based perspective, but there were also obvious examples of annotation errors.

Word Embeddings

Word embeddings have become one of the major building blocks in NLP, and our work is no exception in this regard. We used embeddings mainly as input representations for the neural models, but we also investigated their ability to recover relational similarity as defined in the compound datasets. To this end, in Chapter 5, we cast compound interpretation as an analogy

problem and adapted two vector arithmetic methods (that are often used for word analogy) to predict the compound relations in the Tratz, NomBank and PCEDT datasets. We also experimented with another method that relies solely on attributional similarity (i.e. similarity between the compound constituents). Our experiments showed varyingly limited ability of word embeddings to capture the semantic relations defined in the three datasets. On the one hand, we found that we can achieve an accuracy well above the majority class baseline on the Tratz dataset by just applying simple vector arithmetic operations. The results on PCEDT, on the other hand, were the opposite; i.e. the majority class baseline led to better results than all other similarity-based methods. In addition, we found that attributional similarity leads to higher accuracy on the Tratz and NomBank datasets, which suggests that the methods used for analogy-like tasks are not equally applicable to our compound interpretation task.

In Chapter 6, using the NomBank, PCEDT and Tratz datasets and eight systematically varied GloVe models, we investigated the utility of word embeddings as input representations to neural classifiers. More specifically, we studied the effect of three embedding properties on the classifier's performance, viz. text pre-processing, embedding dimensionality and size of training data for the embedding. We also quantified the impact of fine-tuning word embeddings as a part of the neural architecture. Among other things, we were able to determine that fine-tuning word embeddings does improve the classification accuracy on all three datasets. At the same time, we found that fine-tuning downplays the impact of the size of the embedding training data and the embedding dimensionality on the classifier's accuracy and macro-averaged F_1 scores. That said, there are still gains in increasing the embedding dimensionality, but these become less pronounced when the embeddings are fine-tuned; e.g. without fine-tuning the 300-dimensional model outperforms the 50-dimensional one by about 12 points in accuracy on the Tratz dataset, but the difference goes down to 2.7 points when the models are fine-tuned (cf. Table 6.6a). Even though the observations above hold for all three datasets, it was clear that no single word embedding model leads to the best performance across the three datasets. For example, the lemma-based embedding model achieved higher accuracy on NomBank than the non-lemmatized model, but we observed the inverse effect on the Tratz dataset.

Neural Architectures and Hyperparameters

To quantify the contribution of the head and modifier nouns to predicting the compound relation, in § 6.4, we defined three neural architectures with constituent-specific layers. These architectures isolated the head and modifier constituents at the embedding layer, hidden layer and both of them. We observed somewhat incompatible patterns between the accuracy and macro-averaged F_1 scores; e.g. on the Tratz dataset, all the constituent-specific models achieved lower accuracy than the model with no constituent-specific layers, but the macro-averaged F_1 scores were the other way around (i.e. all constituent-specific models achieved higher F_1 scores). We saw a highly similar pattern on NomBank also. The accuracy and macro-averaged F_1 results on PCEDT were almost consistent. By inspecting the actual predictions of the models, we found that the impact of the constituent-specific models varies depending on the semantic relations. The overarching prediction patterns, however, indicate that the constituent-specific models tend to improve the performance on relations with prototypical heads or modifiers such as the temporal relation in NomBank.

A significant part of training neural networks often goes to the task of hyperparameter optimization. In § 6.5, we approached this task in two ways. First, we conducted a so-called sensitivity analysis in which we only varied one hyperparameter at a time (much like our embedding experiments). Second, knowing that these hyperparameters are in no way independent, we complemented the sensitivity analysis with random search experiments for hyperparameter optimization. In either case, our focus was more on trying to uncover some patterns and, of course, validate our choice of hyperparameters and less on finding the absolute best set of hyperparameters (in terms of classification accuracy). The sensitivity analysis study partly confirmed what we already knew, but it also informed our hyperparameter values in the random search experiments. For example, it was unsurprising to see that optimizers like SGD take longer to converge, but we also observed that the optimization function RMSprop seemed to be harming the generalization of the model without affecting the accuracy (i.e. the validation loss was increasing with no noticeable impact on accuracy). Further, we determined that adding a dropout layer is likely to help the classifier's performance on all three datasets, but the ideal dropout rate was dataset-specific. In the random search setup, we experimented with 172 hyperparameter combinations per

dataset. We found that even though our original choice of hyperparameters ranked high in comparison to all other configurations (especially on the Tratz and NomBank datasets), even higher accuracy scores can be still achieved on all three datasets with different hyperparameters than the ones we used. However, these hyperparameters were dataset-specific, but for our purpose, it was more important to maintain a 'unified' dataset-independent hyperparameter configuration to isolate the effect of transfer and multi-task learning in Chapter 7, among other reasons.

In an effort to tease apart the impact of 'randomness' on our models, in § 6.7, we conducted two sets of experiments (each consisting of 20 experiments per dataset) focused on random initialization and shuffling (i.e. order of the training data) on all three datasets. Given the nondeterministic nature of training neural networks, it is expected to see varyingly different results with different random seeds and shuffling. However, our experiments revealed that such variance is likely to be higher on PCEDT than on NomBank and Tratz; especially so in the case of random shuffling (cf. Tables 6.13 and 6.14).

Transfer and Multi-Task Learning

In Chapter 7, we presented a comprehensive series of experiments on transfer (TL) and multi-task learning (MTL) in the context of our NomBank and PCEDT datasets. We defined three transfer learning models and two multi-task learning models. In either case, the two datasets were used as auxiliary and main (or source and target) tasks, leading to a total of ten experimental setups. These experiments were evaluated in contrast to the so-called single-task learning approach we followed throughout Chapter 6. In addition to accuracy and macro-averaged F_1, we also analyzed the per-relation F_1 scores on both datasets. Overall, we found that transfer and multi-task learning are most effective on the embedding layer, i.e. when the embedding weights are either transferred or shared. Furthermore, it was quite clear that full sharing of the architecture between the two tasks (in multi-task learning) worsens the performance on both datasets. Perhaps most importantly, the TL and MTL models helped improve the F_1 score on some of the least frequent relations in NomBank and PCEDT, which are—of course—the hardest relations to learn. This result is particularly important given the skewed distribution of relations in PCEDT and NomBank. On the development split, using NomBank as an auxiliary (or source) task to PCEDT was often more effective than the

other way around. However, on the test split both NomBank and PCEDT proved helpful to each other. For example, hard sharing of the embedding layer weights led to 7.31% error reduction on the test split of NomBank.

Throughout our error analysis it transpired that the neural models might be relying on lexical memorization to predict the relations in NomBank and PCEDT. Therefore, in § 7.6, we evaluated the performance of our STL, TL and MTL models on partly and fully unseen compounds. Quantitatively, the TL and MTL models decreased the generalization error on the subset of unseen compounds in both datasets. However, effects of lexical memorization were obvious when we considered the ratio of relation-specific constituents vis-à-vis the relations' F_1 scores. Hence, we could not confidently rule out at least some degrees of lexical memorization in our models.

Finally, we investigated the utility of transfer learning as an initialization strategy to—potentially—reduce the variance in the model's performance (which was observed in § 6.7). The experiments in this regard were focused on the PCEDT dataset only. However, contrary to our intuition, we observed little to no effect on the stability of the model when it was initialized with weights from another model trained on NomBank. It is important to highlight here that our experimental setup may have led to overwriting the transferred weights, i.e. the initialized parameters could have been overwritten quickly because we did not dynamically adapt the optimizer's learning rate. Therefore, this question requires further investigation, which takes us to the last section of this thesis.

Future Work and Outlook

In the following, we sketch out some of the potential avenues for future research; we start with where we left off in the previous section and work our way backward.

On the transfer and multi-task learning front, we foresee numerous directions for future research. For instance, one can incorporate other NLP tasks defined over the NomBank and PCEDT frameworks as auxiliary (or source) tasks to learn the noun–noun compound relations in our dataset. Such tasks include semantic role labeling of nominal predicates in NomBank annotations as well as verbal predicates in PropBank. Furthermore, we believe that there is room for further improvement and experimentation in terms of how the TL and MTL models are defined and how the learning strategies are implemented.

For example, we believe it is worth experimenting with other transfer learning strategies where the transferred weights are kept static (i.e. frozen), updated using a less aggressive optimizer (i.e. with lower learning rate) or unfrozen gradually (i.e. after a certain number of epochs for each layer). In addition, in all the MTL experiments presented in this work, the loss of the main and auxiliary tasks contributed equally to the overall loss during training. We believe it is also worthwhile experimenting with dynamic weighting of the loss contribution of the main and auxiliary tasks, following the work by Lauscher et al. (2018).

In terms of the underlying resources, our noun–noun compound dataset inherently lends itself to further improvements and extension. As highlighted in § 4.6, there are other parallel annotations that we have yet to tap into; for example, the VerbNet thematic roles in NomBank. Even though the VerbNet roles do not cover the full NomBank, we believe adding them to our dataset would be beneficial for more contrastive analysis of the underlying resources, and perhaps for further experimentation in the transfer and multi-task learning realm. The potential issue of annotation inconsistency in PCEDT can be exploited to reduce the frequency of the underspecified relation `RSTR`. For example, for the compound types that are annotated with both `RSTR` and non-`RSTR` functors, one can replace all `RSTR`s with the most frequent non-`RSTR` functor.

Even without any improvement, our dataset still has potential for new research avenues. The triple annotation of the internal structure of multi-word compounds in our dataset allows straightforward experimentation with the task of compound bracketing. More importantly, the multi-layered annotation in the dataset also opens up for experimentation with compound bracketing and interpretation in a joint learning or multi-task learning setups. Even though Ó Séaghdha and Copestake (2013) showed that context features are of little help to compound interpretation on their own dataset, we believe it is worthwhile revisiting this conclusion on other datasets—and our dataset provides just the information needed for such experiments.

Lastly, with our new approach to compound interpretation, we would like to believe that we have sowed the seeds for integrating the analysis of noun–noun compounds in well-established whole-sentence meaning representation frameworks. Therefore, an obvious way forward would be to include more and more such meaning representations in compound analysis.

References

Astudillo, R., Amir, S., Ling, W., Silva, M., & Trancoso, I. (2015). Learning Word Representations from Scarce and Noisy Data with Embedding Subspaces. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (p. 1074 – 1084). Association for Computational Linguistics.

Baldwin, T., & Tanaka, T. (2004). Translation by Machine of Complex Nominals. Getting it right. In *Proceedings of the Second ACL Workshop on Multiword Expressions: Integrating Processing* (p. 24 – 31). Association for Computational Linguistics.

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., ... Schneider, N. (2014). *Abstract Meaning Representation (AMR) 1.1 Specification.* Retrieved from `http://www.isi.edu/~ulf/amr/help/amr-guidelines.pdf` (Version of February 11, 2014)

Barker, K., & Szpakowicz, S. (1998). Semi-Automatic Recognition of Noun Modifier Relationships. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Meeting of the Association for Computational Linguistics* (p. 96 – 102). Association for Computational Linguistics.

Baroni, M., Dinu, G., & Kruszewski, G. (2014). Don't Count, Predict! A Systematic Comparison of Context-Counting vs. Context-Predicting Semantic Vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (p. 238 – 247). Association for Computational Linguistics.

Bauer, L. (2011). Typology of Compounds. In R. Lieber & P. Štekauer (Eds.), *The Oxford Handbook of Compounding.* Oxford University Press.

Bergsma, S., Pitler, E., & Lin, D. (2010). Creating Robust Supervised Classifiers via Web-Scale N-Gram Data. In *Proceedings of the 48th Annual*

Meeting of the Association for Computational Linguistics (p. 865 – 874). Association for Computational Linguistics.

Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, *13*(Feb), 281 – 305.

Bingel, J., & Søgaard, A. (2017). Identifying Beneficial Task Relations for Multi-Task Learning in Deep Neural Networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)* (p. 164 – 169). Association for Computational Linguistics.

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. Beijing: O'Reilly.

Bloomfield, L. (1984). *Language*. University of Chicago Press.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, *5*, 135 – 146.

Bonial, C., Bonn, J., Conger, K., Hwang, J. D., & Palmer, M. (2014). PropBank: Semantics of New Predicate Types. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)* (p. 3013 – 3019). European Language Resources Association (ELRA).

Bos, J., Basile, V., Evang, K., Venhuizen, N. J., & Bjerva, J. (2017). The Groningen Meaning Bank. In N. Ide & J. Pustejovsky (Eds.), *Handbook of Linguistic Annotation* (p. 463 – 496). Dordrecht: Springer Netherlands.

Bos, J., & Nissim, M. (2015). Uncovering Noun-Noun Compound Relations by Gamification. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NoDaLiDa 2015)* (p. 251 – 255). Linköping University Electronic Press, Sweden.

Brants, T., & Franz, A. (2006). *Web 1T 5-gram Corpus Version 1.1*. Linguistic Data Consortium.

Burnard, L. (2000). *Reference Guide for the British National Corpus Version 1.0*. Oxford University Computing Services Oxford.

Butnariu, C., Kim, S. N., Nakov, P., Ó Séaghdha, D., Szpakowicz, S., & Veale, T. (2010). SemEval-2010 Task 9: The Interpretation of Noun Compounds Using Paraphrasing Verbs and Prepositions. In *Proceedings of the 5th International Workshop on Semantic Evaluation* (p. 39 – 44).

Association for Computational Linguistics.

Caruana, R. (1993). Multitask Learning: A Knowledge-Based Source of Inductive Bias. In *Proceedings of the Tenth International Conference on Machine Learning* (p. 41 – 48). San Francisco (CA): Morgan Kaufmann.

Caruana, R. (1997). Multitask Learning. *Machine Learning*, *28*(1), 41 – 75.

Cinková, S., Hajič, J., Mikulová, M., Mladová, L., Nedolužko, A., Pajas, P., ... Žabokrtský, Z. (2006). *Annotation of English on the Tectogrammatical Level: Reference Book* (Tech. Rep.). Prague: Charles University. Retrieved from `http://ufal.mff.cuni.cz/pcedt2.0/publications/TR_En.pdf` (version 1.0.1)

Cinková, S., Toman, J., Hajič, J., Čermáková, K., Klimeš, V., Mladová, L., ... Žabokrtskỳ, Z. (2009). Tectogrammatical Annotation of the Wall Street Journal. *The Prague Bulletin of Mathematical Linguistics*, *92*, 85 – 104.

Clark, S. (2015). Vector Space Models of Lexical Meaning. In *The Handbook of Contemporary Semantic Theory* (p. 493 – 522). John Wiley & Sons, Ltd.

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, *20*(1), 37 – 46.

Collobert, R., Kavukcuoglu, K., & Farabet, C. (2011). Torch7: A Matlab-like Environment for Machine Learning. In *BigLearn, NeurIPS Workshop*.

Collobert, R., & Weston, J. (2008). A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning, ICML* (p. 160 – 167).

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural Language Processing (almost) from Scratch. *The Journal of Machine Learning Research*, *12*(2), 2493 – 2537.

Copestake, A., & Briscoe, E. (2005). Noun Compounds Revisited. In J. I. Tait (Ed.), *Charting a New Course: Natural Language Processing and Information Retrieval: Essays in Honour of Karen Spärck Jones* (p. 129 – 154). Dordrecht: Springer Netherlands.

Cordeiro, S., Ramisch, C., Idiart, M., & Villavicencio, A. (2016). Predicting the Compositionality of Nominal Compounds: Giving Word Embeddings a Hard Time. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (p. 1986 – 1997). Association for Computational Linguistics.

Crane, M. (2018). Questionable Answers in Question Answering Research: Reproducibility and Variability of Published Results. *Transactions of the Association for Computational Linguistics*, *6*, 241 – 252.

Davies, M. (2009). The 385+ Million Word Corpus of Contemporary American English (1990–2008+): Design, Architecture, and Linguistic Insights. *International Journal of Corpus Linguistics*, *14*(2), 159 – 190.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, *41*(6), 391 – 407.

Dima, C. (2016). On the Compositionality and Semantic Interpretation of English Noun Compounds. In *Proceedings of the 1st Workshop on Representation Learning for NLP* (p. 27 – 39). Association for Computational Linguistics.

Dima, C., & Hinrichs, E. (2015). Automatic Noun Compound Interpretation using Deep Neural Networks and Word Embeddings. In *Proceedings of the 11th International Conference on Computational Semantics* (p. 173 – 183). Association for Computational Linguistics.

Downing, P. (1977). On the Creation and Use of English Compound Nouns. *Language*, *53*(4), 810 – 842.

Fabb, N. (2017). Compounding. In *The Handbook of Morphology* (p. 66 – 83). John Wiley & Sons, Ltd.

Farahmand, M., Smith, A., & Nivre, J. (2015). A Multiword Expression Data Set: Annotating Non-Compositionality and Conventionalization for English Noun Compounds. In *Proceedings of the 11th Workshop on Multiword Expressions* (p. 29 – 33). Association for Computational Linguistics.

Fares, M. (2016). A Dataset for Joint Noun-Noun Compound Bracketing and Interpretation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics – Student Research Workshop* (p. 72 – 79). Association for Computational Linguistics.

Fares, M., Kutuzov, A., Oepen, S., & Velldal, E. (2017). Word Vectors, Reuse, and Replicability: Towards a Community Repository of Large-Text Resources. In *Proceedings of the 21st Nordic Conference on Computational Linguistics* (p. 271 – 276). Association for Computational Linguistics.

Fares, M., Oepen, S., & Velldal, E. (2015). Identifying Compounds: On The Role of Syntax. In *Proceedings of the 14th International Workshop on*

Treebanks and Linguistic Theories (p. 273 – 283).

Fares, M., Oepen, S., & Velldal, E. (2018). Transfer and Multi-Task Learning for Noun–Noun Compound Interpretation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (p. 1488 – 1498). Association for Computational Linguistics.

Faruqui, M., Tsvetkov, Y., Rastogi, P., & Dyer, C. (2016). Problems With Evaluation of Word Embeddings Using Word Similarity Tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP* (p. 30 – 35). Association for Computational Linguistics.

Finin, T. W. (1980). The Semantic Interpretation of Nominal Compounds. In *Proceedings of the First Annual National Conference on Artificial Intelligence*. AAAI Press.

Firth, J. R. (1957). A Synopsis of Linguistic Theory 1930-55. In *Studies in Linguistic Analysis* (Vol. 1952-59, p. 1 – 32). Oxford: The Philological Society. (Reprinted in: Palmer, F. R. (ed.) (1968). Selected Papers of J. R. Firth 1952-59, pages 168-205. Longmans, London.)

Flickinger, D. (2000). On Building a More Efficient Grammar by Exploiting Types. *Natural Language Engineering*, *6 (1)*, 15 – 28.

Flickinger, D., Zhang, Y., & Kordoni, V. (2012). DeepBank. A Dynamically Annotated Treebank of the Wall Street Journal. In *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories* (p. 85 – 96). Edições Colibri.

Fort, K., Adda, G., & Cohen, K. B. (2011). Last Words: Amazon Mechanical Turk: Gold Mine or Coal Mine? *Computational Linguistics*, *37*(2).

Francis, W. N., & Kucera, H. (1979). *Brown Corpus Manual* (Tech. Rep.). Department of Linguistics, Brown University.

Girju, R. (2009). The Syntax and Semantics of Prepositions in the Task of Automatic Interpretation of Nominal Phrases and Compounds: A Cross-Linguistic Study. *Computational Linguistics*, *35*(2).

Girju, R., Badulescu, A., & Moldovan, D. (2003). Learning Semantic Constraints for the Automatic Discovery of Part-Whole Relations. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (p. 80 – 87). Association for Computational Linguistics.

Girju, R., Moldovan, D., Tatu, M., & Antohe, D. (2005). On the Semantics of Noun Compounds. *Computer Speech & Language*, *19*(4), 479 – 496.

Girju, R., Nakov, P., Nastase, V., Szpakowicz, S., Turney, P., & Yuret, D.

(2007). SemEval-2007 Task 04: Classification of Semantic Relations between Nominals. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)* (p. 13 – 18). Association for Computational Linguistics.

Goldberg, Y. (2017). *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Graff, D., Kong, J., Chen, K., & Maeda, K. (2005). *English Gigaword Second Edition*. Linguistic Data Consortium.

Grimshaw, J. (1990). *Argument Structure*. Cambridge, MA: MIT Press.

Hajič, J., Hajičová, E., Panevová, J., Sgall, P., Bojar, O., Cinková, S., ... Žabokrtský, Z. (2012). Announcing Prague Czech-English Dependency Treebank 2.0. In *Proceedings of the 8th International Conference on Language Resources and Evaluation* (p. 3153 – 3160). European Language Resources Association (ELRA).

Harris, Z. S. (1954). Distributional Structure. *Word*, *10*(2-3), 146 – 162.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference and Prediction* (2nd ed.). Springer.

Hendrickx, I., Kim, S. N., Kozareva, Z., Nakov, P., Ó Séaghdha, D., Padó, S., ... Szpakowicz, S. (2010). SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation* (p. 33 – 38). Association for Computational Linguistics.

Hendrickx, I., Kozareva, Z., Nakov, P., Ó Séaghdha, D., Szpakowicz, S., & Veale, T. (2013). SemEval-2013 Task 4: Free Paraphrases of Noun Compounds. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)* (p. 138 – 143). Association for Computational Linguistics.

Hermann, K. M., Blunsom, P., & Pulman, S. (2012). An Unsupervised Ranking Model for Noun-Noun Compositionality. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)* (p. 132 – 141). Association for Computational Linguistics.

Hill, F., Reichart, R., & Korhonen, A. (2015). SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. *Computational Linguistics*, *41*(4), 665 – 695.

Iordachioaia, G., van der Plas, L., & Jagfeld, G. (2016). The Grammar of English Deverbal Compounds and their Meaning. In *Proceedings of the Workshop on Grammar and Lexicon: Interactions and Interfaces (GramLex)* (p. 81 – 91).

Isabelle, P. (1984). Another Look at Nominal Compounds. In *Proceedings of the 10th International Conference on Computational Linguistics and 22nd Annual Meeting on Association for Computational Linguistics* (p. 509 – 516). Association for Computational Linguistics.

Jespersen, O. (1949). *A Modern English Grammar on Historical Principles, Part VI Morphology*. Enjnar Munksgaard.

Jurgens, D., Mohammad, S., Turney, P., & Holyoak, K. (2012). SemEval-2012 Task 2: Measuring Degrees of Relational Similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (SemEval 2012)* (p. 356 – 364). Association for Computational Linguistics.

Kay, P., & Zimmer, K. (1990). On the Semantics of Compounds and Genitives in English. In S. L. Tsohatzidis (Ed.), *Meanings and Prototypes: Studies in Linguistic Categorization* (p. 239 – 246). Routledge.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *Proceedings of the 5th International Conference on Learning Representations (ICLR).*

Kiela, D., & Clark, S. (2014). A Systematic Study of Semantic Vector Space Model Parameters. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)* (p. 21 – 30). Association for Computational Linguistics.

Kim, S. N., & Baldwin, T. (2005). Automatic Interpretation of Noun Compounds Using WordNet Similarity. In *Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP)* (p. 945 – 956). Springer Berlin Heidelberg.

Kim, S. N., & Baldwin, T. (2006). Interpreting Semantic Relations in Noun Compounds via Verb Semantics. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions* (p. 491 – 498). Association for Computational Linguistics.

Kim, S. N., & Baldwin, T. (2008). Standardised Evaluation of English Noun Compound Interpretation. In *Proceedings of the LREC Workshop: Towards a Shared Task for Multiword Expressions* (p. 39 – 42).

Kim, S. N., & Baldwin, T. (2013). A Lexical Semantic Approach to Interpreting and Bracketing English Noun Compounds. *Natural Language Engineering*, *19*(03), 385 – 407.

Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*.

Koehn, P. (2005). Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the 10th Machine Translation Summit* (p. 79 – 86).

Lapata, M. (2002). The Disambiguation of Nominalizations. *Computational Linguistics*, *28*(3), 357 – 388.

Lapata, M., & Keller, F. (2004). The Web as a Baseline: Evaluating the Performance of Unsupervised Web-based Models for a Range of NLP Tasks. In *Proceedings of the 2004 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (p. 121 – 128). Association for Computational Linguistics.

Lapata, M., & Lascarides, A. (2003). Detecting Novel Compounds. The Role of Distributional Evidence. In *Proceedings of the 10th Meeting of the European Chapter of the Association for Computational Linguistics* (p. 235 – 242).

Lauer, M. (1995). *Designing Statistical Language Learners: Experiments on Noun Compounds*. Doctoral dissertation, Macquarie University, Sydney, Australia.

Lauscher, A., Glavaš, G., Ponzetto, S. P., & Eckert, K. (2018). Investigating the Role of Argumentation in the Rhetorical Analysis of Scientific Publications with Neural Multi-Task Learning Models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (p. 3326 – 3338). Association for Computational Linguistics.

Lebret, R., & Collobert, R. (2014). Word Embeddings through Hellinger PCA. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics* (p. 482 – 490). Association for Computational Linguistics.

Levi, J. N. (1978). *The Syntax and Semantics of Complex Nominals*. Academic Press.

Levin, B. (1993). *English Verb Classes and Alternations A Preliminary Investigation*. Chicago and London: University of Chicago Press.

Levy, O., & Goldberg, Y. (2014). Linguistic Regularities in Sparse and Explicit Word Representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning* (p. 171 – 180). Association for Computational Linguistics.

Levy, O., Goldberg, Y., & Dagan, I. (2015). Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association of Computational Linguistics*, *3*, 211 – 225.

Levy, O., Remus, S., Biemann, C., & Dagan, I. (2015). Do Supervised Distributional Methods Really Learn Lexical Inference Relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (p. 970 – 976). Association for Computational Linguistics.

Li, C. N. (1972). *Semantics and the Structure of Compounds in Chinese*. Unpublished doctoral dissertation, University of California, Berkeley.

Lipton, Z. C., & Steinhardt, J. (2018). Troubling Trends in Machine Learning Scholarship. *arXiv preprint arXiv:1807.03341*.

Lund, K., & Burgess, C. (1996). Producing High-Dimensional Semantic Spaces from Lexical Co-occurrence. *Behavior Research Methods, Instruments, & Computers*, *28*(2), 203 – 208.

Lyu, C., & Titov, I. (2018). AMR Parsing as Graph Prediction with Latent Alignment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (p. 397 – 407). Association for Computational Linguistics.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., & McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (p. 55 – 60). Association for Computational Linguistics.

Marcus, M., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a Large Annotated Corpora of English. The Penn Treebank. *Computational Linguistics*, *19*, 313 – 330.

Martínez Alonso, H., & Plank, B. (2017). When Is Multitask Learning Effective? Semantic Sequence Prediction under Varying Data Conditions. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*

(p. 44 – 53). Association for Computational Linguistics.

McShane, M., Beale, S., & Babkin, P. (2014). Nominal Compound Interpretation by Intelligent Agents. *LiLT (Linguistic Issues in Language Technology)*, *10*.

Meyers, A. (2007). *Annotation Guidelines for NomBank – Noun Argument Structure for PropBank* (Vol. 44; Tech. Rep.). New York University.

Meyers, A., Reeves, R., Macleod, C., Szekely, R., Zielinska, V., Young, B., & Grishman, R. (2004). Annotating Noun Argument Structure for NomBank. In *Proceedings of the 4th International Conference on Language Resources and Evaluation* (p. 803 – 806). European Language Resources Association (ELRA).

Mikolov, T., Chen, K., Corrado, G. S., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. In *International Conference on Learning Representations Workshop*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (Volume 2)* (p. 3111 – 3119). Curran Associates Inc.

Mikolov, T., Yih, W., & Zweig, G. (2013). Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (p. 746 – 751). Association for Computational Linguistics.

Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM*, *38*(11), 39 – 41.

Minsky, M. (2007). *Emotion Machine: Commonsense Thinking, Artificial Intelligence, and the Future of the Human Mind*. Simon and Schuster.

Moldovan, D., Badulescu, A., Tatu, M., Antohe, D., & Girju, R. (2004). Models for the Semantic Classification of Noun Phrases. In *Proceedings of the HLT-NAACL Workshop on Computational Lexical Semantics* (p. 60 – 67). Association for Computational Linguistics.

Mou, L., Meng, Z., Yan, R., Li, G., Xu, Y., Zhang, L., & Jin, Z. (2016). How Transferable Are Neural Networks in NLP Applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (p. 479 – 489). Association for Computational Linguistics.

Nakov, P. (2007). *Using the Web as an Implicit Training Set: Application to*

Noun Compound Syntax and Semantics. Doctoral dissertation, EECS Department, University of California, Berkeley.

Nakov, P. (2013). On the Interpretation of Noun Compounds: Syntax, Semantics, and Entailment. *Natural Language Engineering*, *19*(3), 291 – 330.

Nakov, P., & Hearst, M. (2013). Semantic Interpretation of Noun Compounds Using Verbal and Other Paraphrases. *ACM Transactions on Speech and Language Processing*, *10*(3).

Nastase, V., Sayyad-Shirabad, J., Sokolova, M., & Szpakowicz, S. (2006). Learning Noun-modifier Semantic Relations with Corpus-based and WordNet-based Features. In *Proceedings of the 21st National Conference on Artificial Intelligence (Volume 1)* (p. 781 – 786). AAAI Press.

Nastase, V., & Szpakowicz, S. (2003). Exploring Noun-Modifier Semantic Relations. In *Proceedings of the Fifth International Workshop on Computational Semantics* (p. 285 – 301).

Oepen, S., Kuhlmann, M., Miyao, Y., Zeman, D., Cinková, S., Flickinger, D., ... Urešová, Z. (2015). SemEval 2015 Task 18. Broad-Coverage Semantic Dependency Parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation* (p. 915 – 926). Association for Computational Linguistics.

Oepen, S., Kuhlmann, M., Miyao, Y., Zeman, D., Flickinger, D., Hajič, J., ... Zhang, Y. (2014). SemEval 2014 Task 8. Broad-Coverage Semantic Dependency Parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation* (p. 63 – 72). Association for Computational Linguistics.

Ó Séaghdha, D. (2008). *Learning Compound Noun Semantics* (Tech. Rep. No. UCAM-CL-TR-735). Cambridge, UK: University of Cambridge, Computer Laboratory.

Ó Séaghdha, D., & Copestake, A. (2007). Co-occurrence Contexts for Noun Compound Interpretation. In *Proceedings of the Workshop on A Broader Perspective on Multiword Expressions* (p. 57 – 64). Association for Computational Linguistics.

Ó Séaghdha, D., & Copestake, A. (2008). Semantic Classification with Distributional Kernels. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)* (p. 649 – 656). COLING 2008 Organizing Committee.

Ó Séaghdha, D., & Copestake, A. (2009). Using Lexical and Relational Simi-

larity to Classify Semantic Relations. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics* (p. 621 – 629). Association for Computational Linguistics.

Ó Séaghdha, D., & Copestake, A. (2013). Interpreting Compound Nouns with Kernel Methods. *Journal of Natural Language Engineering*, *19*(3), 331 – 356.

Padó, S., & Lapata, M. (2003). Constructing Semantic Space Models from Parsed Corpora. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics* (p. 128 – 135).

Palmer, M., Gildea, D., & Kingsbury, P. (2005). The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, *31*(1), 71 – 106.

Pan, S. J., & Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345 – 1359.

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment Classification using Machine Learning Techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing* (p. 79 – 86). Association for Computational Linguistics.

Parker, R., Graff, D., Kong, J., Chen, K., & Maeda, K. (2011). *English Gigaword Fifth Edition LDC2011T07* (Tech. Rep.). Technical Report. Linguistic Data Consortium, Philadelphia.

Peng, H., Thomson, S., & Smith, N. A. (2017). Deep Multitask Learning for Semantic Dependency Parsing. In *Proceedings of the 55th Meeting of the Association for Computational Linguistics* (p. 2037 – 2048). Association for Computational Linguistics.

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (p. 1532 – 1543). Association for Computational Linguistics.

Pitler, E., Bergsma, S., Lin, D., & Church, K. (2010). Using Web-scale N-grams to Improve Base NP Parsing Performance. In *Proceedings of the 23rd International Conference on Computational Linguistics* (p. 886 – 894).

Pollard, C., & Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. Chicago, USA: The University of Chicago Press.

Ponkiya, G., Patel, K., Bhattacharyya, P., & Palshikar, G. (2018). Treat Us Like the Sequences We Are: Prepositional Paraphrasing of Noun

Compounds using LSTM. In *Proceedings of the 27th International Conference on Computational Linguistics* (p. 1827 – 1836). Association for Computational Linguistics.

Pourdamghani, N., Gao, Y., Hermjakob, U., & Knight, K. (2014). Aligning English Strings with Abstract Meaning Representation Graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (p. 425 – 429). Association for Computational Linguistics.

Prechelt, L. (2012). Early Stopping — But When? In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural Networks: Tricks of the Trade: Second Edition* (p. 53 – 67). Berlin, Heidelberg: Springer Berlin Heidelberg.

Pulman, S. (2013). Distributional Semantic Models. In C. Heunen, M. Sadrzadeh, & E. Grefenstette (Eds.), *Quantum Physics and Linguistics: A Compositional, Diagrammatic Discourse* (p. 333 – 358). Oxford University Press.

Reddy, S., McCarthy, D., & Manandhar, S. (2011). An Empirical Study on Compositionality in Compound Nouns. In *Proceedings of 5th International Joint Conference on Natural Language Processing* (p. 210 – 218). Asian Federation of Natural Language Processing.

Reimers, N., & Gurevych, I. (2017). eporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (p. 338 – 348). Association for Computational Linguistics.

Resnik, P. S. (1993). Selection and Information: A Class-Based Approach to Lexical Relationships. *IRCS Technical Reports Series*, 200.

Roeper, T., & Siegel, M. E. (1978). A Lexical Transformation for Verbal Compounds. *Linguistic Inquiry*, *9*, 199 – 260.

Rosario, B., & Hearst, M. (2001). Classifying the Semantic Relations in Noun Compounds via a Domain-Specific Lexical Hierarchy. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing* (p. 82 – 90).

Rose, T., Stevenson, M., & Whitehead, M. (2002). The Reuters Corpus Volume 1 - from Yesterday's News to Tomorrow's Language Resources. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation* (Vol. 2, p. 827 – 832).

Ryder, M. E. (1994). *Ordered Chaos: The Interpretation of English Noun-*

Noun Compounds (Vol. 123). University of California Press.

Sahlgren, M. (2008). The Distributional Hypothesis. *Italian Journal of Linguistics*, *20*(1), 33 – 54.

Santorini, B. (1990). Part-of-Speech Tagging Guidelines for the Penn Treebank Project (3rd Revision, 2nd printing). *Ms., Department of Linguistics, UPenn. Philadelphia, PA*.

Schnabel, T., Labutov, I., Mimno, D., & Joachims, T. (2015). Evaluation Methods for Unsupervised Word Embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (p. 298 – 307). Association for Computational Linguistics.

Schuler, K. K. (2005). *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. Doctoral dissertation, University of Pennsylvania, Philadelphia, PA.

Schulte im Walde, S., Müller, S., & Roller, S. (2013). Exploring Vector Space Models to Predict the Compositionality of German Noun-Noun Compounds. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity* (p. 255 – 265). Association for Computational Linguistics.

Semecký, J., & Cinková, S. (2006). Constructing an English Valency Lexicon. In *Proceedings of the Workshop on Frontiers in Linguistically Annotated Corpora 2006* (p. 94 – 97). Association for Computational Linguistics.

Sgall, P., Hajičová, E., & Panevová, J. (1986). *The Meaning of the Sentence and its Semantic and Pragmatic Aspects*. Dordrecht, The Netherlands: D. Reidel Publishing Company.

Shwartz, V., Goldberg, Y., & Dagan, I. (2016). Improving Hypernymy Detection with an Integrated Path-based and Distributional Method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (p. 2389 – 2398). Association for Computational Linguistics.

Shwartz, V., & Waterson, C. (2018). Olive Oil is Made *of* Olives, Baby Oil is Made *for* Babies: Interpreting Noun Compounds using Paraphrases in a Neural Model. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (p. 218 – 224). Association for Computational Linguistics.

Søgaard, A., & Goldberg, Y. (2016). Deep Multi-Task Learning with Low Level

Tasks Supervised at Lower Layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (p. 231 – 235). Association for Computational Linguistics.

Spärck Jones, K. (1983). *Compound Noun Interpretation Problems* (Tech. Rep.). University of Cambridge, Computer Laboratory.

Sproat, R. (1994). English Noun-Phrase Accent Prediction for Text-to-Speech. *Computer Speech & Language*, *8*(2), 79 – 94.

Su, S. (1969). *A Semantic Theory Based Upon Interactive Meaning* (Tech. Rep.). University of Wisconsin-Madison Department of Computer Sciences.

Tratz, S. (2011). *Semantically-Enriched Parsing for Natural Language Understanding*. Doctoral dissertation, University of Southern California.

Tratz, S., & Hovy, E. (2010). A Taxonomy, Dataset, and Classifier for Automatic Noun Compound Interpretation. In *Proceedings of the 48th Meeting of the Association for Computational Linguistics* (p. 678 – 687). Association for Computational Linguistics.

Turney, P. D. (2006). Similarity of Semantic Relations. *Computational Linguistics*, *32*(3), 379 – 416.

Turney, P. D., & Pantel, P. (2010). From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, *37*(1), 141 – 188.

Vadas, D., & Curran, J. (2007). Adding Noun Phrase Structure to the Penn Treebank. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics* (p. 240 – 247). Association for Computational Linguistics.

Vadas, D., & Curran, J. (2011). Parsing Noun Phrases in the Penn Treebank. *Computational Linguistics*, *37*(4), 753 – 809.

Vanderwende, L. (1994). Algorithm for Automatic Interpretation of Noun Sequences. In *Proceedings of the 15th Conference on Computational Linguistics (Volume 2)* (p. 782 – 788).

Warren, B. (1978). *Semantic Patterns of Noun-Noun Compounds* (Vol. Gothenburg Studies in English). Acta Universitatis Gothoburgensis.

Williams, J. (2013). Multi-Domain Learning and Generalization in Dialog State Tracking. In *Proceedings of the SIGDIAL 2013 Conference* (p. 433 – 441). Association for Computational Linguistics.

Wittgenstein, L. (1953). *Philosophical Investigations*. Oxford: Basil Blackwell.

Yazdani, M., Farahmand, M., & Henderson, J. (2015). Learning Semantic Composition to Detect Non-compositionality of Multiword Expressions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (p. 1733 – 1742). Association for Computational Linguistics.

Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. *CoRR*, *abs/1212.5701*. Retrieved from `http://arxiv.org/abs/1212.5701`

Zhang, Y., & Wallace, B. (2017). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (p. 253 – 263). Asian Federation of Natural Language Processing.

Appendix A

Noun Compound Relations

A.1 Nastase and Szpakowicz (2003)

The dataset of noun-modifier relations by Nastase and Szpakowicz (2003) allows adjectival modifiers not just nous, e.g. *periodic surveillance*. All the instances annotated as `object property` have an adjectival modifier. Nastase and Szpakowicz (2003) indicate the relation `manner` is represented in the dataset, but we could not find any instance of that relation. Therefore, we do not include this relation in Table A.1.

Relation	Count	Examples
CAUSALITY		
cause	17	cloud storm
effect	34	birth pain
purpose	31	bathing suit
detraction	4	sun block
TEMPORALITY		
frequency	16	periodic surveillance
time at	30	summer morning
time through	6	spring semester
SPATIAL		
direction	8	exit route
location	5	college town
location at	22	ocean side
location from	21	farm boy
PARTICIPANT		
agent	36	student protest
beneficiary	9	pet spray
instrument	35	needle work
object	33	horse doctor
object property	15	torn paper
part	9	party member
possessor	30	student loan
property	49	novelty item
product	16	light bulb
source	12	olive oil
stative	9	sleeping dog
whole	7	student committee
QUALITY		
container	3	story idea
content	15	photo album
equative	5	child actor
material	32	carbon deposit
measure	30	saturation point
topic	45	eviction notice
type	16	oak tree

Table A.1: Relations in Nastase and Szpakowicz (2003)

Appendix B

NomBank and PCEDT Relations

This appendix includes the definitions of the noun–noun compound relations in our dataset, introduced in Chapter 4.

B.1 NomBank Relations

1. `ARG0`: Typically expresses causer, agent or actor.
2. `ARG1`: Typically expresses patient or theme.
3. `ARG2`: Typically expresses recipient or beneficiary.
4. `ARG3` and `ARG4`: "While `ARG3` and `ARG4` are undoubtedly less regular than `ARG0`, `ARG1`, and `ARG2`, there are some detectable patterns. For example, if not already covered by the first three arguments, start-point/source tend to be `ARG3` and end-point/goal tend to be `ARG4`" (Meyers, 2007, p. 25).
5. `ARG5`, `ARG8` and `ARG9`: The interpretation of these arguments completely depends on the predicate. These arguments are extremely infrequent in our datasets; `ARG5` occurs three times and `ARG9` occurs four times.
6. `ARGM-LOC`: Locative modifiers that are not directional.[1]

[1]NomBank defines a directional modifier (`ARGM-DIR`) which expresses the start and end points of motion, giving or sending. This modifier is not attested in our dataset.

7. ARGM-MNR: Manner. Meyers (2007) distinguishes between several subtypes of manner adjuncts (such as instrumental, concomitant and measure), but these distinctions are only made for the annotators.

8. ARGM-TMP: Temporal modifiers which includes points in tie, time periods, durations, approximate times, frequencies and relative times (Meyers, 2007, p. 92).

9. ARGM-PNC: Purpose. Even though purpose is expressed as an adjunct in our dataset, Meyers (2007) states that some arguments can receive a purpose tag (-PNC) such as ARG2-PNC but such cases are not observed in our dataset.

10. ARGM-ADV: Adverbial modifier such as epistemic and attitudinal adverbs.

11. ARGM-EXT: Extent. Meyers (2007) explains that there is a subtle distinction between modifiers like ARGM-EXT, ARGM-MNR and ARGM-TMP because the latter two include notions of degree and frequency which could be viewed as extent also.

12. ARGM-CAU: Cause.

13. ARG1-H1, ARG3-H0, ARG1-H0, ARG0-H0: -H0, and -H1 are called hyphen tags and they specify which constituent of a hyphenated string is the argument. For example, ARG0-H0 specifies that the first constituent of the hyphenated string is the ARG0.

14. Support: Meyers (2007, p. 79) explains that some head nouns and prenominal modifiers can share their arguments or adjuncts, and in such cases the head noun is annotated with the relation Support.

B.2 PCEDT Functors

The following presents the definition of the PCEDT functors found in our dataset (in Chapter 4). Th definitions are adapted from the PCEDT documentation.

1. RSTR: Adnominal adjunct modifying its governing noun

2. PAT: Patient (argument)

3. APP: Adnominal adjunct expressing appurtenance
4. REG: Adjunct expressing a circumstance that the main predication takes into account
5. ACT: Actor (argument)
6. LOC: Locative adjunct, answering the question "where"
7. TWHEN: Temporal adjunct, answering the question "when"
8. AIM: Adjunct expressing purpose
9. ID: The nominative of identity and explicative genitive
10. MAT: Adnominal argument referring to the content of a container
11. NE: Part of a named entity
12. ORIG: Origo (argument)
13. MANN: Adjunct expressing the manner (of doing something)
14. MEANS: Adjunct expressing a means (of doing something)
15. EFF: Effect (argument)
16. AUTH: Adnominal adjunct referring to the author (of something)
17. BEN: Adjunct expressing that something is happening for the benefit (or disadvantage) of somebody or something
18. ADDR: Addressee (argument)
19. CAUS: Adjunct expressing the cause (of something)
20. THL: Temporal adjunct, answering the questions "how long" and "after how long"
21. CRIT: Adjunct expressing a criterion/measure/standard
22. DIR1: Directional adjunct, answering the question "where from"
23. DIR3: Directional adjunct, answering the question "where to"

24. TFRWH: Temporal adjunct, answering the question “from when”
25. EXT: Adjunct expressing extent
26. TFHL: Temporal adjunct, answering the question “for how long”
27. CPR: Adjunct expressing comparison
28. DIFF: Adjunct expressing a difference (between two entities, states etc.)
29. ACMP: Adjunct expressing accompaniment (in the broad sense of the word)
30. DIR2: Directional adjunct, answering the question “which way”
31. DPHR: The dependent part of an idiomatic expression
32. RESL: Adjunct expressing the result/effect of something
33. THO: Temporal adjunct, answering the questions “how often” and “how many times”
34. TPAR: Temporal adjunct, answering the questions “in parallel/simultaneously with what” and “during what time”

Appendix C

Model Hyperparameters

In this appendix we include the plots and figures related to the hyperparameter experiments presented in Chapter 6.

Figures C.1, C.2 and C.3 show the validation loss and accuracy of the STL model using several optimization functions on the PCEDT, NomBank and Tratz datasets. Figures C.4, C.5 and C.6 show the impact of the batch size on the validation loss and accuracy of the three datasets. Further, Figures C.7, C.8 and C.9 show the effect of different values of the dropout rate on PCEDT, NomBank and Tratz. Lastly, Figures C.10, C.11 and C.12 show the effect of the dense layer size on the validation accuracy in the hyperparameter random search experiments.

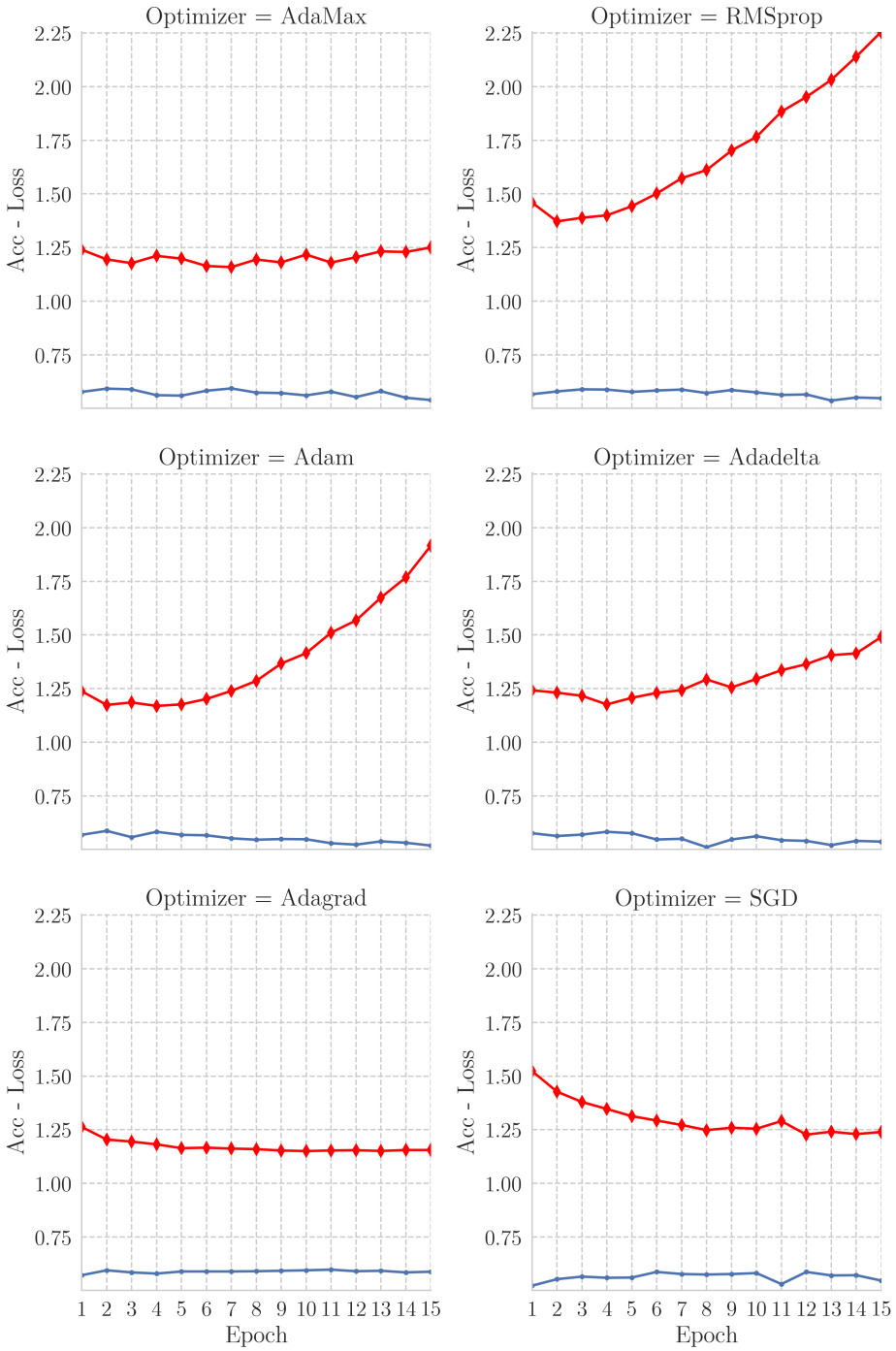


Figure C.1: Validation accuracy (blue) and loss (red) of different optimizers on PCEDT

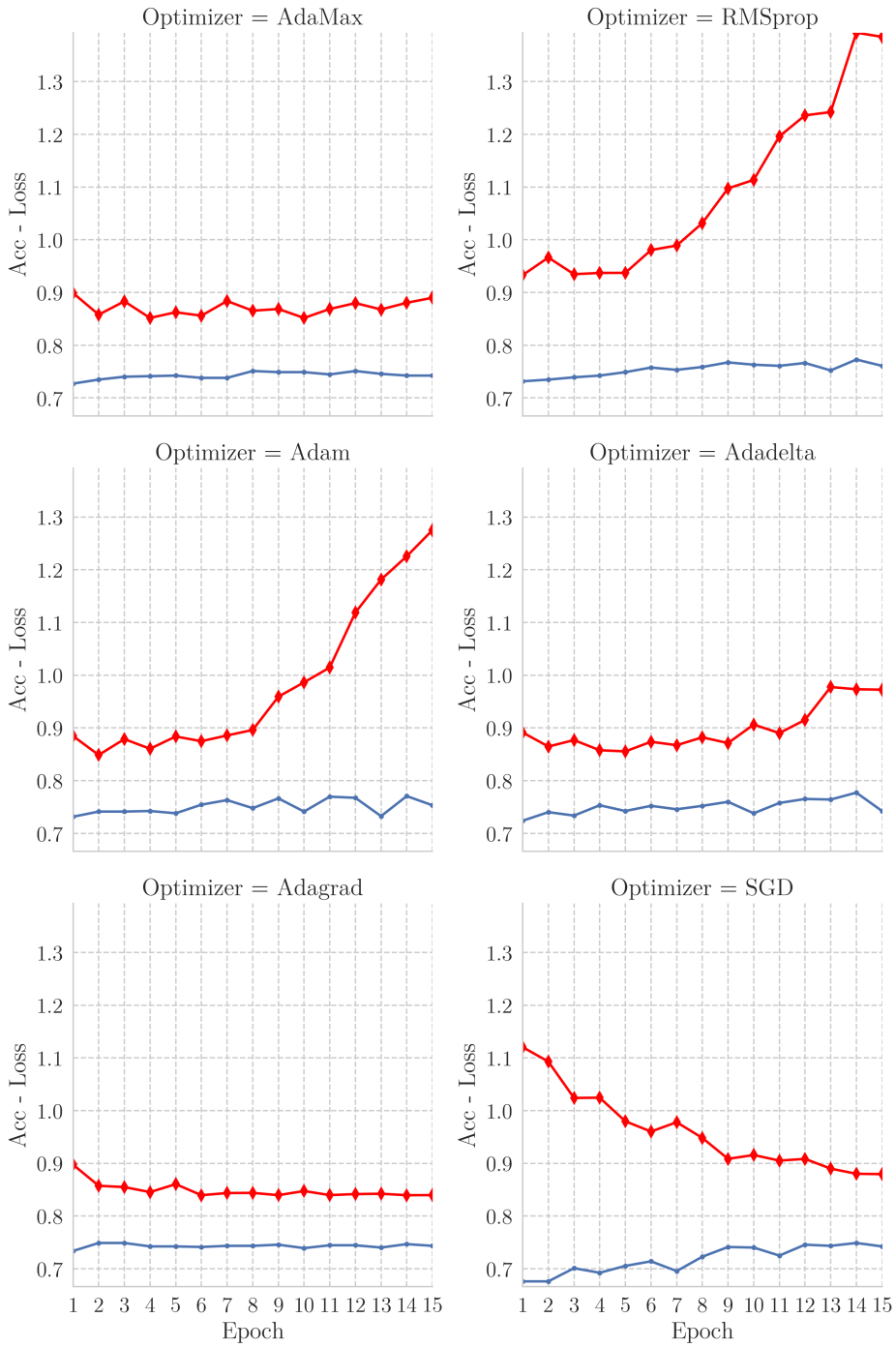


Figure C.2: Validation accuracy (blue) and loss (red) of different optimizers on NomBank

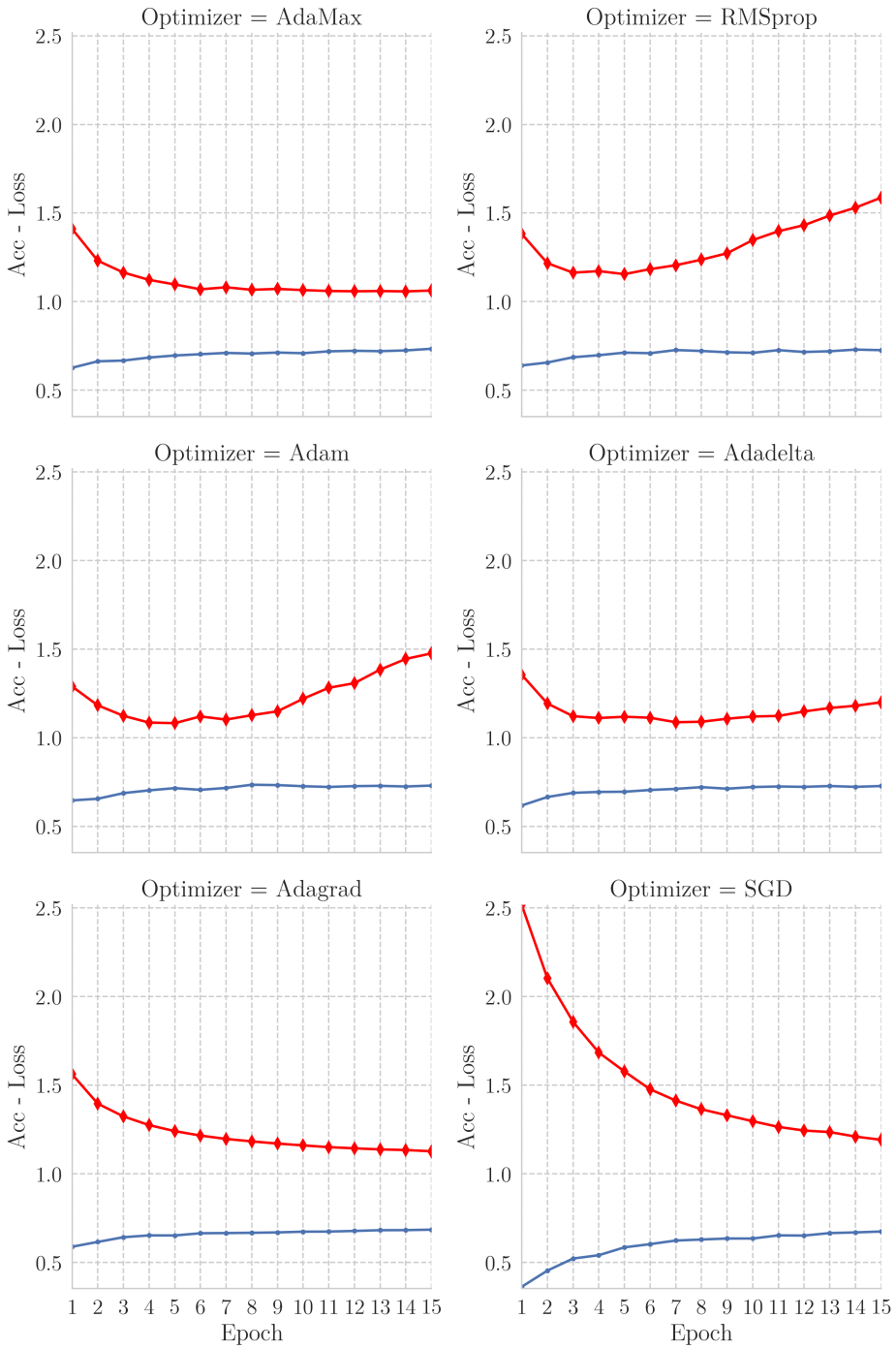


Figure C.3: Validation accuracy (blue) and loss (red) of different optimizers on Tratz

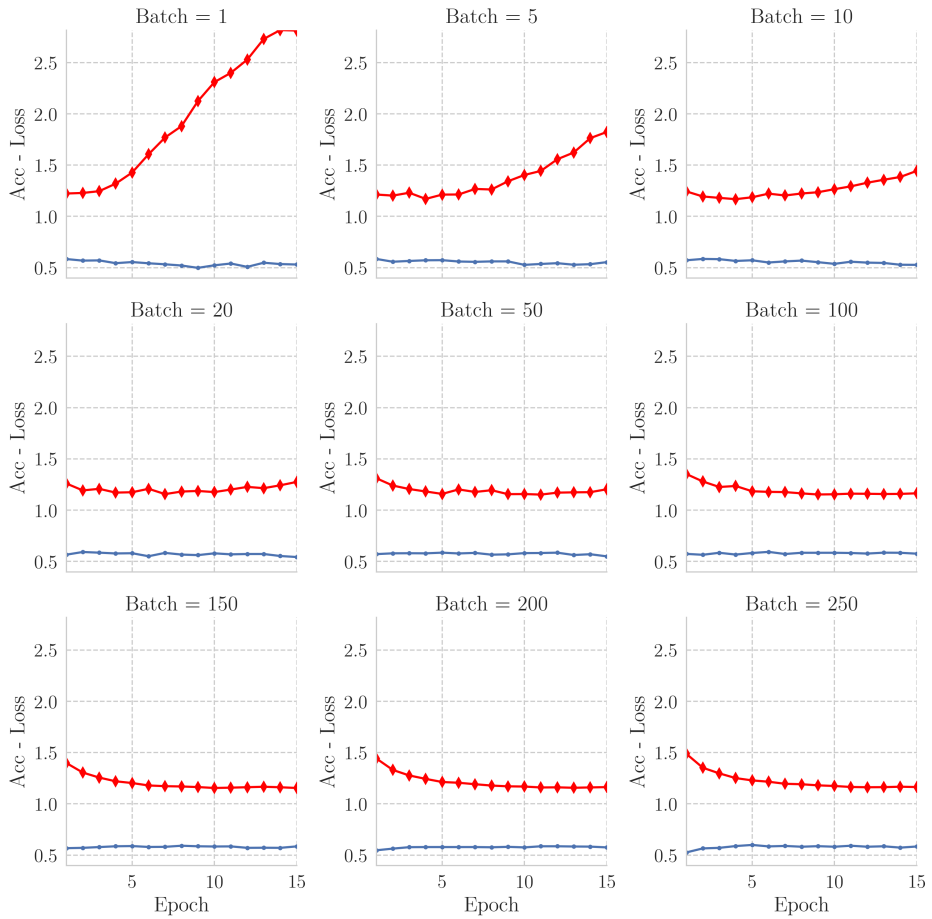


Figure C.4: Loss (red) and validation accuracy (blue) vs. batch size on PCEDT

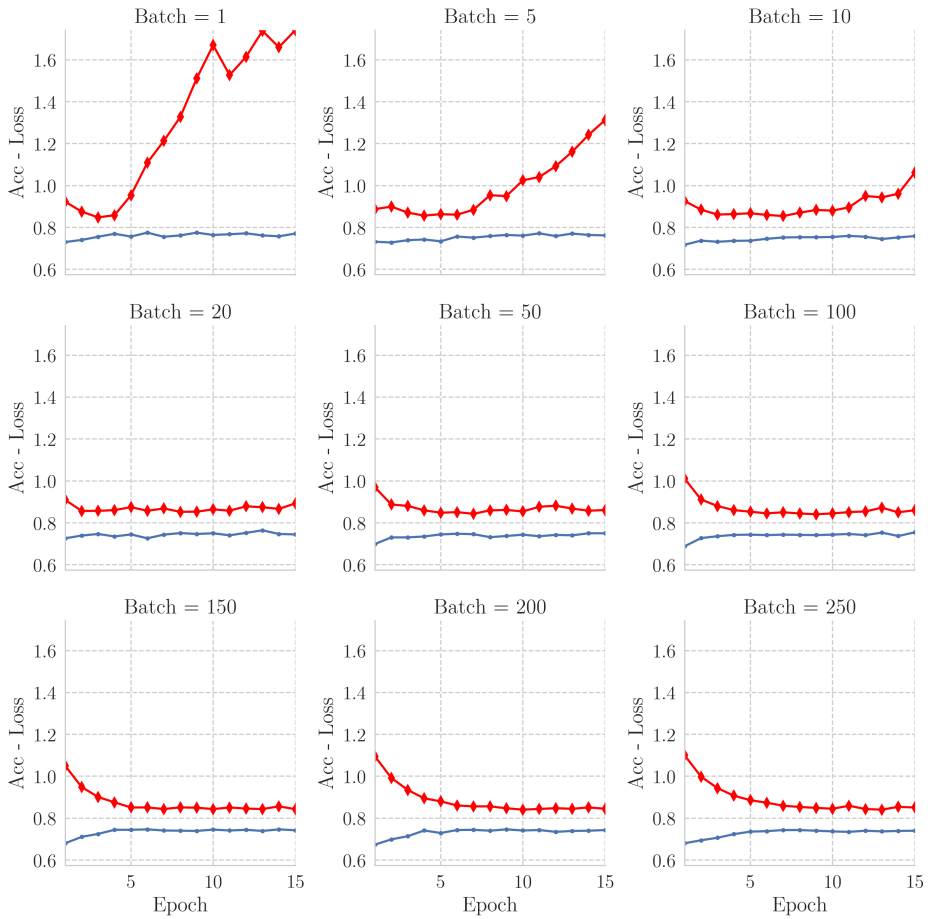


Figure C.5: Loss (red) and validation accuracy (blue) vs. batch size on NomBank

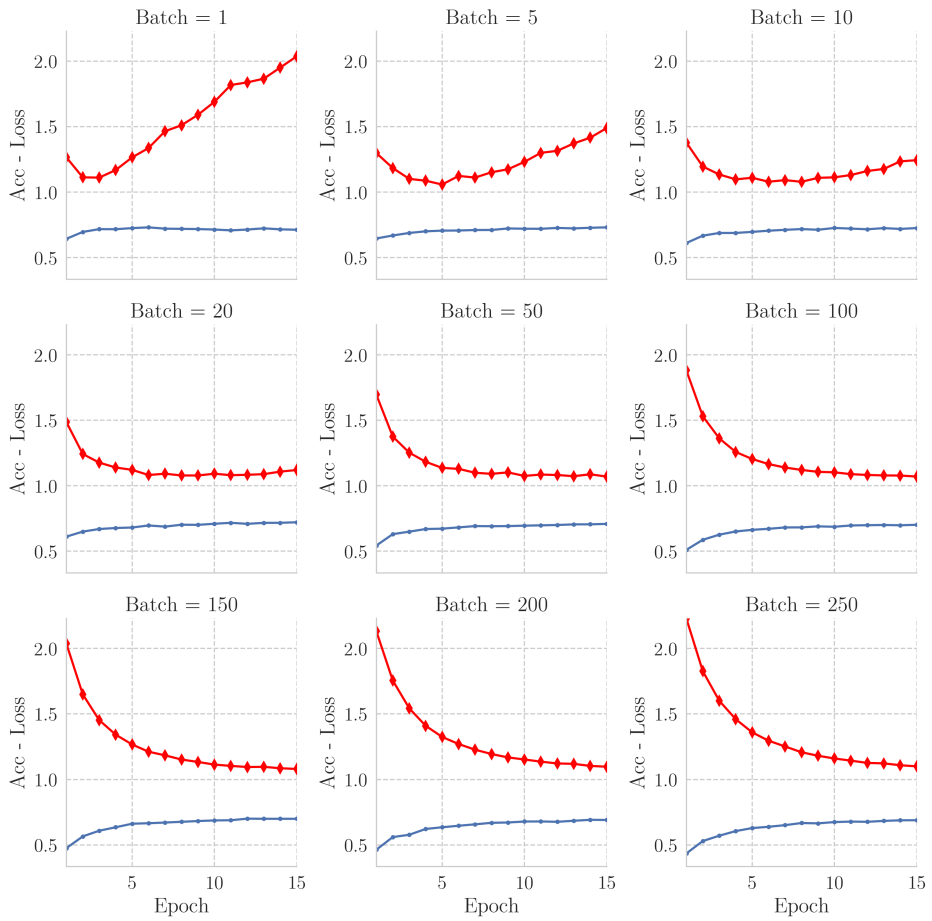


Figure C.6: Loss (red) and validation accuracy (blue) vs. batch size on Tratz

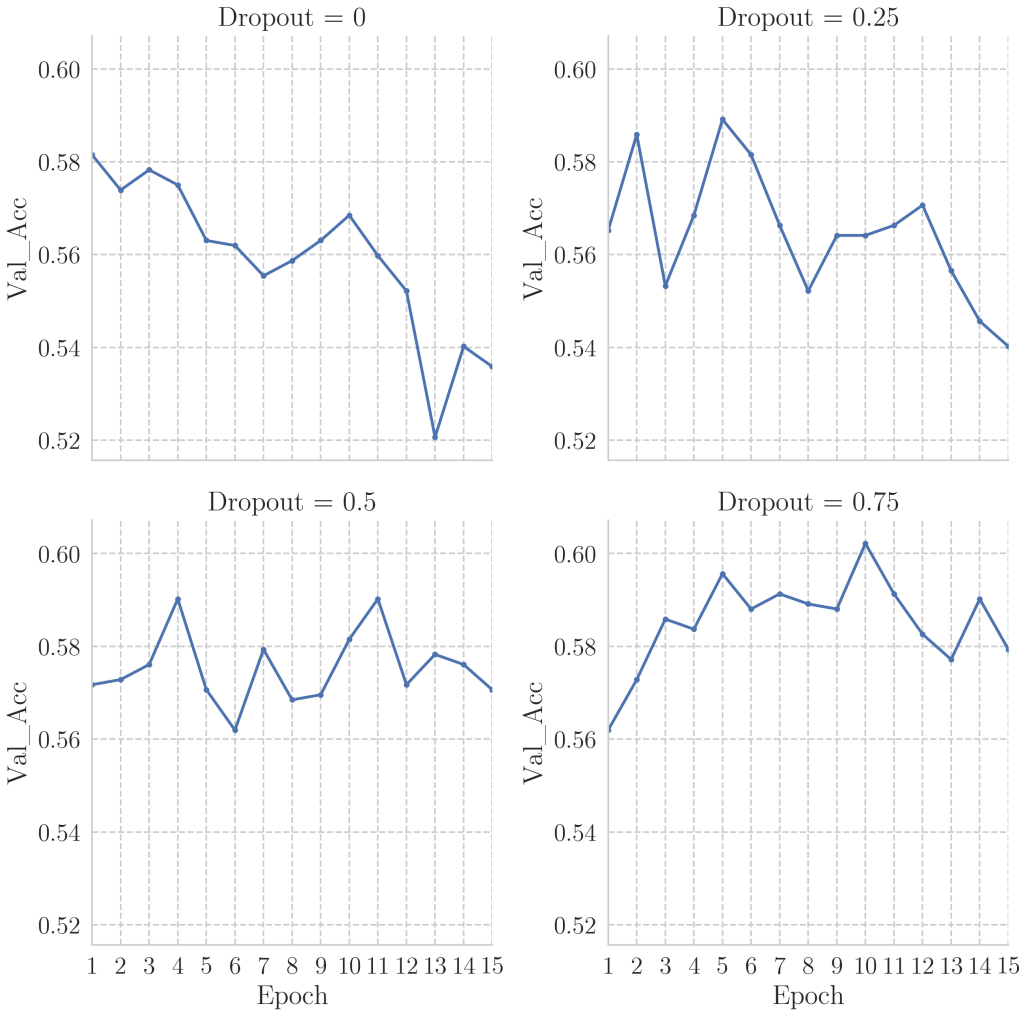


Figure C.7: Validation accuracy vs. dropout rate on PCEDT

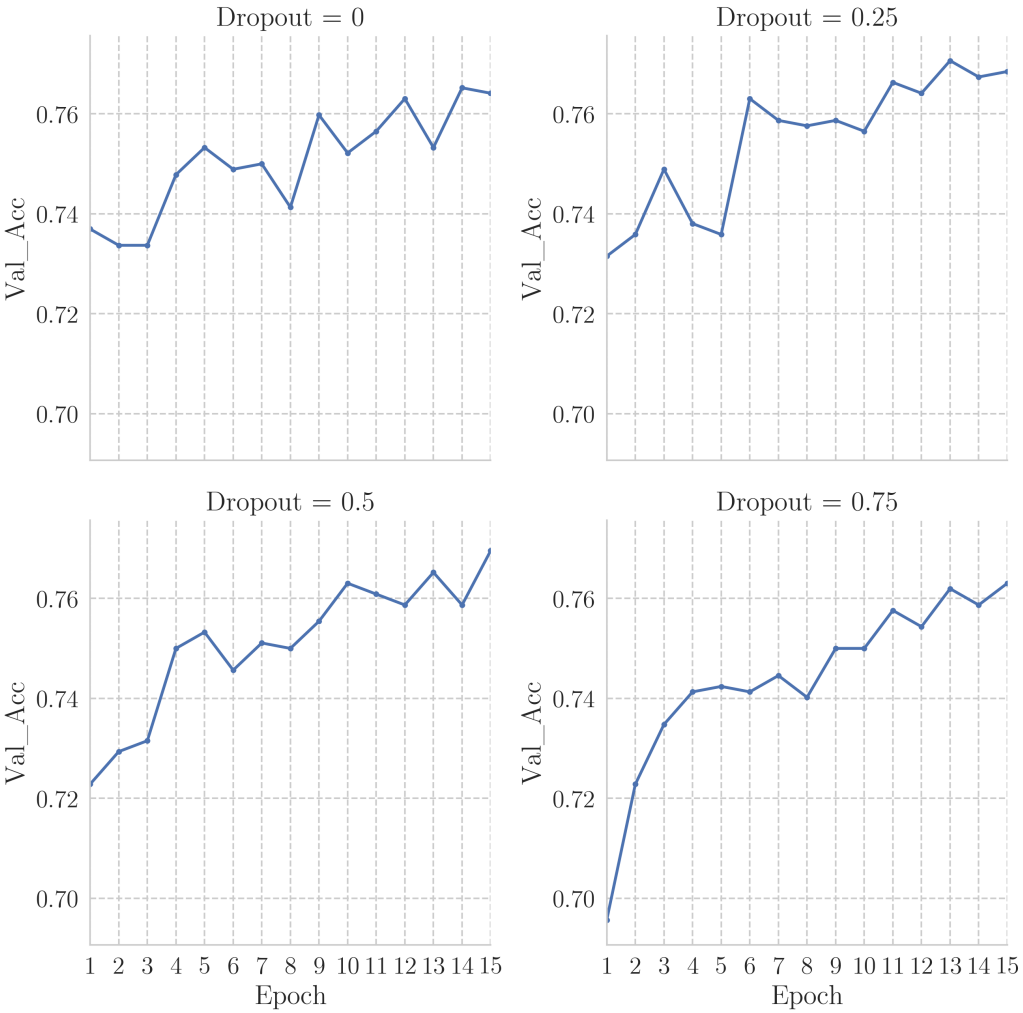


Figure C.8: Validation accuracy vs. dropout rate on NomBank

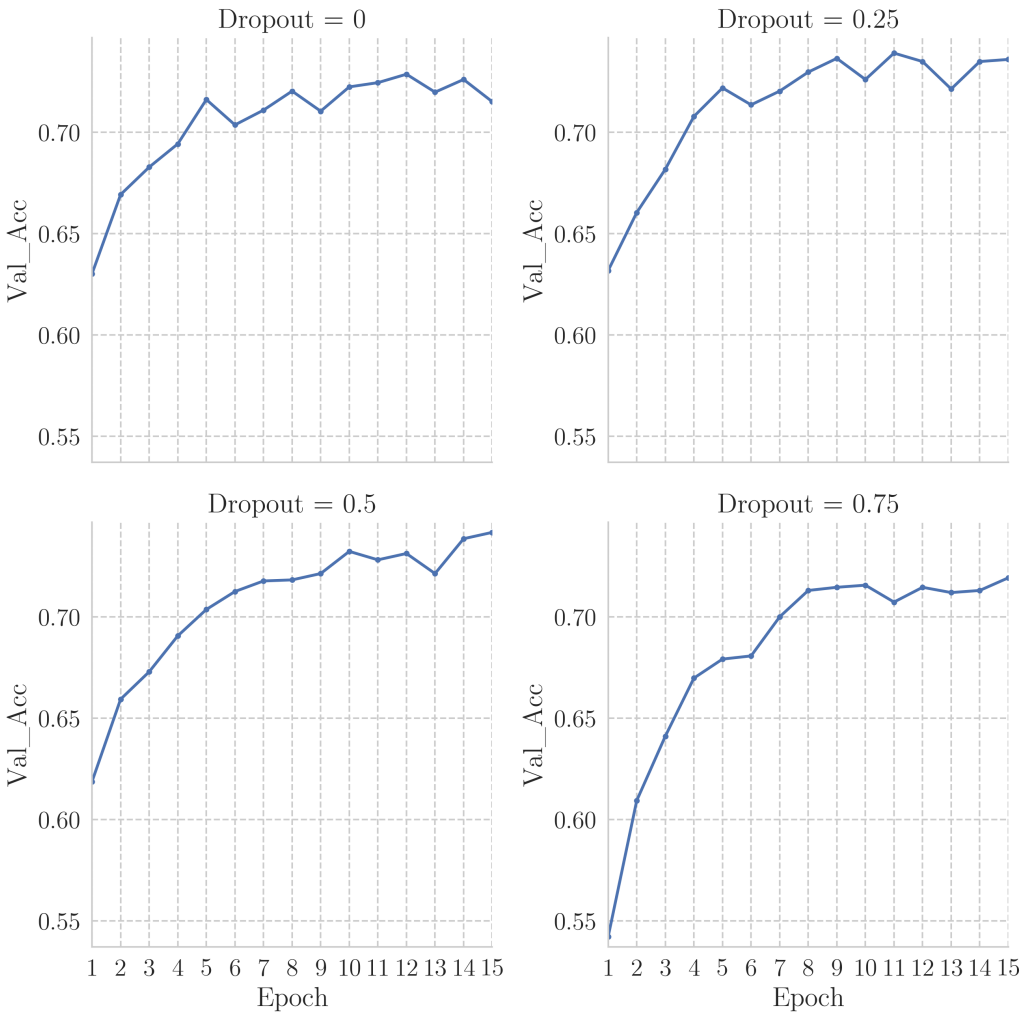


Figure C.9: Validation accuracy vs. dropout rate on Tratz

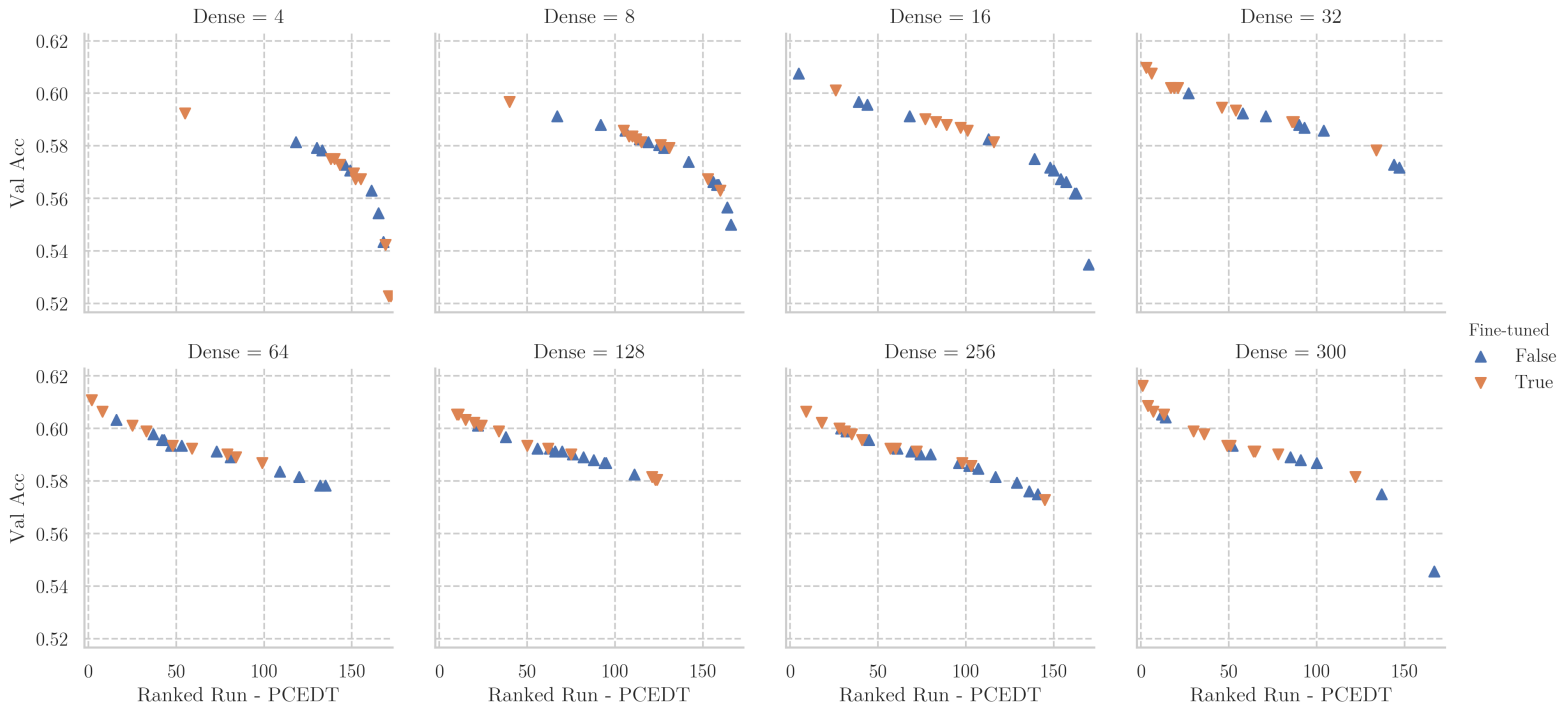


Figure C.10: PCEDT validation accuracy vs. size of the dense layer in the hyperparameter random search experiments

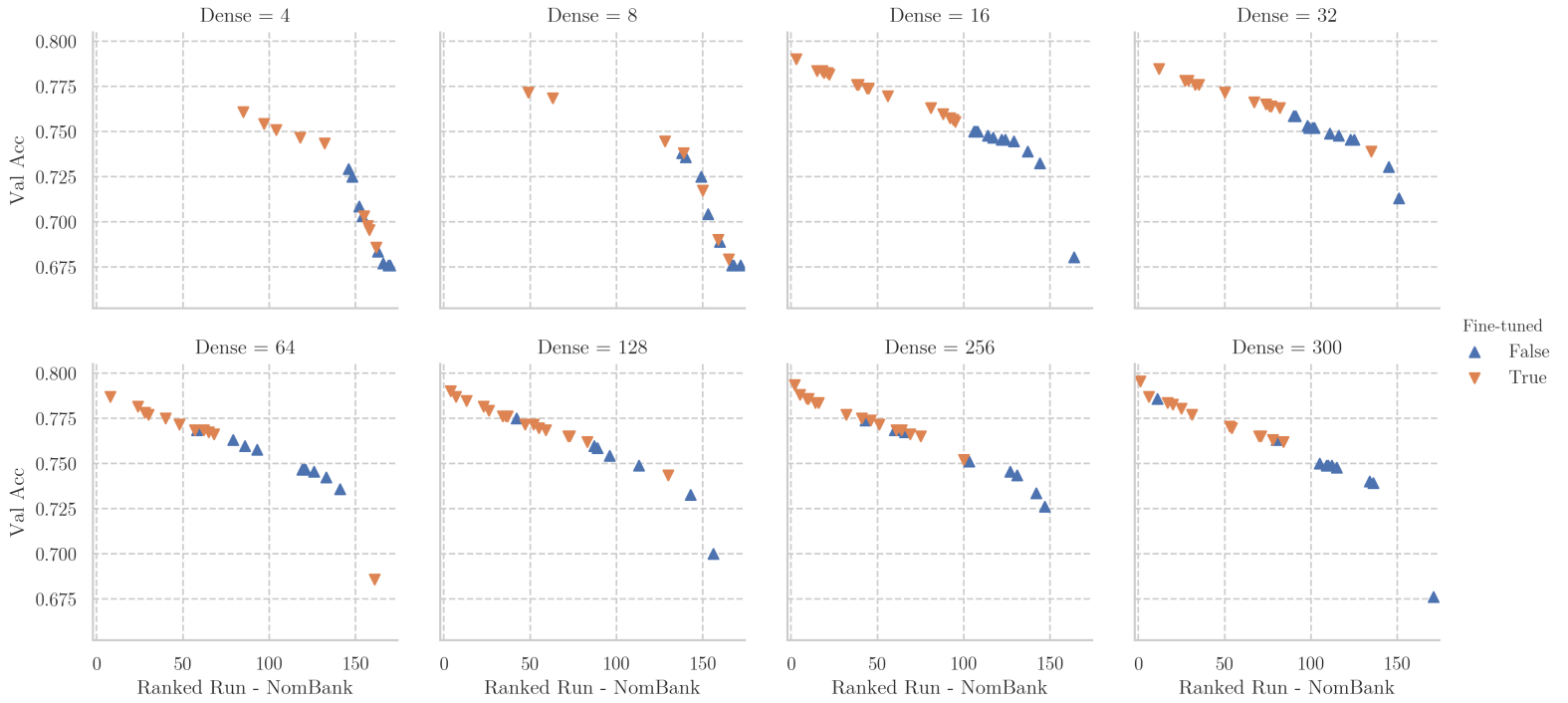


Figure C.11: NomBank validation accuracy vs. size of the dense layer in the hyperparameter random search experiments

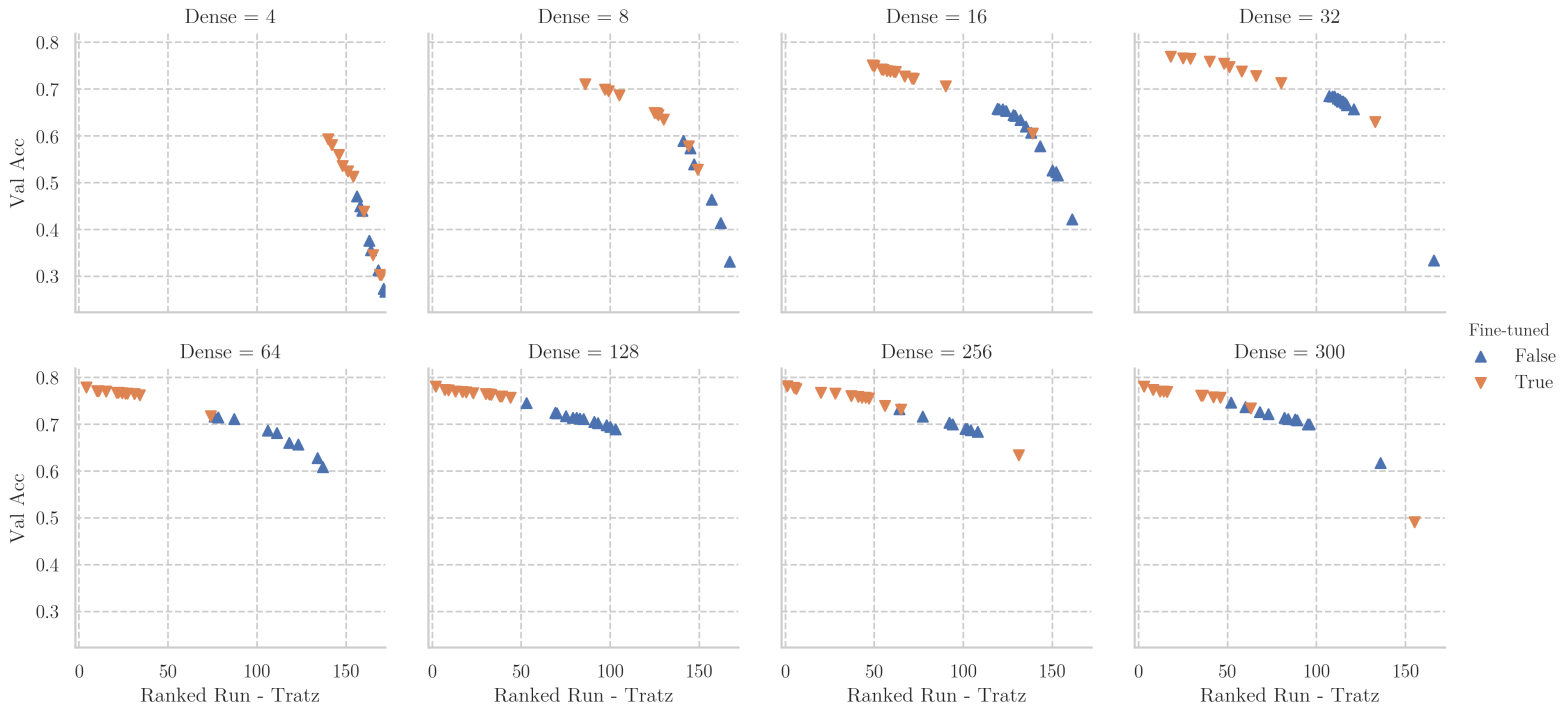


Figure C.12: Tratz validation accuracy vs. size of the dense layer in the hyperparameter random search experiments