Exploring automatic approaches for sentiment lexicon creation for Norwegian

Karianne Kirkeby Amundsen

Thesis submitted for the degree of
Master in Informatics: Language and Communication
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2019

Exploring automatic approaches for sentiment lexicon creation for Norwegian

Karianne Kirkeby Amundsen

Exploring automatic approaches for sentiment lexicon creation for Norwegian

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

The Norwegian language is under-resourced for various Natural Language Processing tasks, including the task of sentiment analysis. However, recent publications of sentiment analysis datasets such as the Norwegian Review Corpus, and the Pros and Cons dataset which is currently being prepared as part of the SANT project, facilitate the creation of new tools and resources. The Norwegian language lacks several tools for sentiment analysis tasks, which have been available for the English language for decades. In this work, we aim to fill some of the gap for sentiment analysis for Norwegian, by performing large-scale experiments on automatic methods for sentiment lexicon creation and expansion. We make use of distributional models as well as resources containing lexical relations. We also experiment with different approaches for improving the quality of word embedding models to use for lexicon creation. Furthermore, we will perform the first binary CNN classification on the reviews with the most extreme positive and negative ratings in the Norwegian Review Corpus, and provide a baseline CNN architecture for this specific task and dataset. The results of this thesis can be used for further research for sentiment analysis for the Norwegian language.

Acknowledgements

I would first and foremost thank my supervisors Samia Touileb and Erik Velldal for your excellent support and guidance. Samia, I am very grateful for your precise feedback, and that you pay attention to the small details. Erik, thank you for steering me in the right direction whenever it was needed. I would also like to thank my family for their continued support throughout my studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sentiment analysis is a field within Natural Language Processing which aims to identify opinions or attitudes present in a text. Subjective information in a text can be used to determine whether the author has a positive, negative, or neutral attitude towards a topic. Different kinds of sentiments can be derived from text, such as emotions, mood, interpersonal stance, attitudes, and personality traits (Jurafsky & Martin, 2018). In this work, the focus is on determining the polarity of a text, whether the text is positive or negative towards a specific topic. Sentiment analysis can be performed as an independent task where the only goal is to determine the sentiment of a text, or it can be used as a tool to measure the performance of sentiment lexicons or other tools created for sentiment analysis.

Sentiment lexicons are resources that contain words or sequences of words, labelled positive or negative. Some lexicons also include the strength of the polarity of each set of words, for example on a scale from 1 to 5. There are many approaches to sentiment lexicon creation, they can be manually annotated by one or multiple people, they can be translated from another language either manually or using a translation software, or extracted from reviews which have been labelled positive or negative. Another widely used approach for lexicon creation is to identify a set of seed words, which is used to increase the size of the lexicon, by for example extracting their neighbours from a word embedding model, or extracting words that co-occur with the seed words in a corpus.

Sentiment lexicons can be used to determine the polarity of a text simply by identifying words from the lexicon that are present in the text being analysed, counting the number of positive and negative labels, and assigning polarity based on the number of positive and negative words in the text. They can also be used as a single feature along with many other features in more complex algorithms for sentiment classification (Jurafsky & Martin, 2018), for example in a convolutional neural network. The information from sentiment lexicons can for example be used in methods involving attention, or encoding the lexicon into the embeddings in neural networks.

Traditional use of sentiment analysis focused on the use of sentiment lexicons, however, methods involving neural models have often been found to outperform these lexicons. Even though sentiment lexicons may not always produce the best results, they have many advantages over neural models. Sentiment lexicons are transparent, the content of the lexicon can easily be interpreted by manual inspection, while neural models are not easily interpretable, and most of the time it is not clear why the model makes a specific prediction. Another advantage of sentiment lexicons over neural models is that sentiment lexicons can be easily altered, and adapted to new domains. Adding or removing words in the lexicon can be done manually, whereas for neural models, alterations and adaptations are much more complicated, and cannot be controlled in the same manner as for sentiment lexicons. However, one of the disadvantages of using only lexicons for the prediction of sentiment, is that it is a primitive resource, which does not take linguistic structures such as negation into account.

The Norwegian language is under-resourced and lacks lexical resources for various Natural Language Processing tasks, including sentiment analysis. While many sentiment lexicons exist for other languages, especially for English, there is a lack of this kind of resource for Norwegian. However, in the recent years, the Sentinent Analysis for Norwegian Texts (SANT)[1] project has provided datasets for sentiment analysis for Norwegian, which facilitates the task of creating more tools for performing sentiment analysis. The Norwegian Review Corpus (Velldal et al., 2017) is a recently published review dataset containing over 35,000 reviews from different domains. The dataset consists of reviews from nine different categories, such as screen, literature, music, and products, and each review is assigned a rating on a scale from 1 to 6. The other dataset, which is currently being prepared as part of the project, is the Pros and Cons dataset, which is a collection of sentences or keyword-based fragments of strong sentiment polarity, published from DinSide.[2] The Pros and Cons dataset consists of 7,300 review-summaries, and is divided into nine categories, such as "data" *(data)*, "bolig" *(home)*, "motor" *(motor)*, and "fritid" *(leisure)*. These sentiment datasets are useful for various tasks of sentiment analysis, and they will also support the creation of new tools. There have been a few previous attempts at creating sentiment lexicons for the Norwegian language. However, the datasets used for evaluation, as well as most of the sentiment lexicons are not freely available, so their results cannot be reproduced.

In this work, we will explore various methods for sentiment lexicon creation for the Norwegian language. We will use and adapt methods that have already been found successful for sentiment lexicon creation for the English language, but also make several novel extensions to these methods. We will explore different variants of two main approaches for sentiment lexicon creation, the first approach involves extracting words with a strong polarity from review datasets, and the second approach involves

[1] https://www.mn.uio.no/ifi/english/research/projects/sant/
[2] https://www.dinside.no/tester

identifying a set of seed words, which will be used for lexicon expansion. We will both create and use existing word embedding models for lexicon expansion. In the final stages of this project, we will explore methods such as retrofitting, and fine-tuning the word embedding models in a CNN, to improve the quality of these models.

1.1 Outline

This thesis is structured as follows.

Chapter 2 provides and overview of previous work on sentiment analysis using sentiment lexicons for classification. Different kinds of sentiment lexicons, algorithms for how they have been generated, and evaluation methods, are discussed in this chapter. This chapter also provides an overview of previous work for the Norwegian language, that involved approaches for creating and evaluating sentiment lexicons. The final part of the chapter covers convolutional neural networks. Research involving a baseline convolutional architecture will be described, as well as how convolutional neural networks can be used to improve the quality of word embedding models used for sentiment lexicon creation.

Chapter 3 presents the existing Norwegian datasets and lexical resources that can be used for sentiment analysis. We will describe the Norwegian WordNet, which is a resource that contains lexical relations. We will also thoroughly describe the Norwegian Review Corpus (Velldal et al., 2017) and the Pros and Cons dataset, which we will later use for various sentiment analysis tasks.

Chapter 4 describes approaches to lexicon creation and expansion using the resources described in Chapter 3, as well as other lexical resources. First, we will present our method for evaluating the sentiment lexicons. Then, we will describe the two main approaches for lexicon creation that we have explored in this work. The first approach consists in extracting words with a strong polarity from labelled text. The second approach involves identifying a set of seed words, and lexicon expansion using various lexical resources. We will also discuss some of the challenges associated with lexicon expansion using automatic methods.

Chapter 5 provides an overview of methods used to improve the quality of word embedding models used for sentiment lexicon expansion. The first sections present convolutional neural networks, and the task of fine-tuning word embedding models. In the second part of the chapter, we will present retrofitting, which is another approach for altering the vector representations in the word embedding models, by combining the information in the models with information from a resource containing lexical relations. In the third part of the chapter, the resulting word embedding models will be evaluated on two standard intrinsic evaluation tasks for word embedding models: analogy prediction and synonym detection. Intrinsic evaluation of word embedding

models is performed to assess the effects of fine-tuning and retrofitting the embeddings. We also investigate the relationship between the intrinsic evaluation of the embeddings that form the basis of lexicon extraction, and the downstream performance of the extracted lexicons.

Chapter 6 describes the top performing sentiment lexicons created in this project. We will first analyse the sentiment lexicons and examine the words with the highest frequencies in development datasets. In the second section, we will perform the final evaluation on the held-out test sets, to find out how the top performing lexicons perform on unseen data.

Chapter 7 summarizes our main findings and presents suggestions for future work within the field of sentiment analysis using sentiment lexicons for the Norwegian language.

Chapter 2

Background

The main focus of this work is on exploring a variety of automatic approaches for sentiment lexicon creation for Norwegian. In this chapter, we will present an overview of methods and algorithms for sentiment lexicon creation and evaluation. Since most of the work in this field has been done for the English language, we will start by presenting related work for sentiment lexicon creation and evaluation in English. We will also present some of the most well-known available English sentiment lexicons. Later in this chapter, we will present the findings from three Norwegian studies on sentiment lexicons. In the final section, we will introduce convolutional neural networks, and describe a method for how they can be used to improve the quality of word embedding models used for sentiment lexicon creation.

The goal of sentiment analysis is to determine opinions, attitudes, or emotions present in a text. Early work on sentiment analysis focused on sentiment lexicons, which were used for review analysis, and the identification of particular attitudes in texts. As pointed out by Pang and Lee (2008), in 2001, the interest for sentiment analysis rose considerably for two main reasons. The first reason for the rise in interest was that people started to realize the vast opportunities and potential applications of sentiment analysis. The second reason was due to the fact that large amounts of data became widely accessible on the web, which in turn made machine learning possible for language technology applications such as sentiment analysis (Pang & Lee, 2008).

2.1 Applications of sentiment analysis

One of the main applications of sentiment analysis is in improving business and marketing strategies. A sentiment analysis system can be used to analyse or categorize information from surveys, in particular open-ended questions, or summarize reviews by extracting the most positive or negative sentences (Turney & Littman, 2003). By categorizing information, or extracting the most important sentences, people can focus on the most important aspects of the review or survey, and do not have to spend time on

reading neutral passages that do not contain relevant information. Based on customer reviews, businesses can tailor their products or services to the needs of the costumers, or adjust their marketing strategies. Political parties can use sentiment analysis to measure how well their campaign is doing, or to determine people's attitudes toward their party (Pang & Lee, 2008) or specific candidates, for example from a set of tweets (Ceron, Curini, Iacus, & Porro, 2014). Another application of sentiment analysis is in the field of chatbots and other systems used for communication (Turney & Littman, 2003). If the system is able to determine whether the incoming text is positive or negative, it will be able to provide a more appropriate answer. Sentiment analysis is a large field with many different approaches, but the focus of this work in on exploring a variety of methods for sentiment lexicon creation. The following sections will cover some of the most well-known English sentiment lexicons, and the methods and algorithms for how they were created.

2.2 Sentiment lexicons

Sentiment lexicons are collections of words associated with a sentiment. The sentiment can be an emotion such as fear or happiness, or a sentiment label (positive or negative). Different sentiment lexicons can have distinct representations of sentiment, they can differ in size, and some may only have labels, whereas others also have assigned a score. Sentiment lexicons can be manually annotated or created using automatic methods. A manually annotated sentiment lexicon is a collection of words labelled by one or more people. It can be created by one individual or it can be a crowd-sourcing effort, in which many people annotate the words in the lexicon, and the inter-rater agreement is calculated. An example of an automatic approach is to extract words from a lexical resource or a corpus, and label the words according to some predefined function. There are advantages and disadvantages to both approaches. Gatti, Guerini, and Turchi (2016) suggest that lexicons that are created in an automatic manner are larger and have better coverage, whereas manually annotated lexicons are usually shorter and have worse coverage because they are very time-consuming to create. They also suggest that manually created lexicons usually have higher precision than the lexicons that have been created in a more automatic manner, because humans are better than computers at understanding ambiguity and distinguishing between synonyms and antonyms.

In machine learning, there is a distinction between supervised and unsupervised learning. In supervised learning, we can make use of labelled data such as reviews with ratings. A review corpus can be used to create a sentiment lexicon, using supervised methods for learning the sentiment of words. Words appearing in the positive reviews can be labelled positive, and words in the negative reviews can be labelled negative. In unsupervised learning, unlabelled data is used for learning, such as a large corpus of raw text. This method for learning relies on finding patterns in the data, so that data with similar characteristics can be clustered together. In sentiment analysis,

unsupervised learning of the sentiment of words can be used to expand a sentiment lexicon after identifying a set of seed words.

Lexicon expansion from a set of seed words can be performed using both supervised and unsupervised methods for learning the sentiment of new words. Jurafsky and Martin (2018) discuss the distinction between supervised and semi-supervised methods for lexicon expansion, and suggest that manually labelling a set of seed words is "minimal human supervision" (p. 10). Therefore, methods that rely on a manually annotated set of seed words for expansion of the sentiment lexicon are considered semi-supervised approaches. In the following sections, we will examine both supervised and semi-supervised approaches for the creation and expansion of sentiment lexicons.

2.3 Sentiment lexicons for English

A widely used approach for sentiment lexicon creation is to first identify a set of positive and negative seed words. The list of seed words can be created in different ways, by manually labelling words positive or negative, or by extracting words that have already been labelled from an existing dataset. The size of the set of seed words can range from just one positive and one negative word, to thousands of words of each polarity (Jurafsky & Martin, 2018). After a set of seed words has been identified, the size of the lexicon can be expanded by adding new words to the lexicon, using resources such as WordNet or collections of raw text. WordNet is a lexical resource for many different languages, which aims to represent lexical relations between words, such as synonyms, antonyms, meronyms, and hyponyms. These lexical relations can be used for lexicon expansion, by adding words that have a relation to the seed words to the sentiment lexicon. This method for lexicon expansion is based on the fact that synonyms have the same polarity whereas antonyms have opposite polarity (Jurafsky & Martin, 2018). New words can also be extracted from collections of text, using methods for pattern matching or statistical measures to identify words of the same polarity. Increasing the lexicon is an iterative process, only the seed words will be used in the first iteration, but as the lexicon grows, so does the number of words used for determining polarity of the new words. The expansion of sentiment lexicons is typically based on distributional methods. In the following paragraphs, we will describe some of the most well-known sentiment lexicons created using the lexical resource WordNet.

2.3.1 Sentiment lexicons from WordNet

Many sentiment lexicons have been generated using the Princeton WordNet[1] (Fellbaum, 1998; Miller, 1995) as a resource for lexicon expansion. Some of the most well-known lexicons are SentiWordNet developed by Esuli and Sebastiani (2006), and SentiWordNet 3.0[2] developed by Baccianella, Esuli, and Sebastiani (2010). These sentiment

[1] https://wordnet.princeton.edu
[2] http://sentiwordnet.isti.cnr.it

lexicons assign a positive, neutral, and negative polarity score to the all the synsets in WordNet. SentiWordNet is based on WordNet version 2.0, whereas SentiWordNet 3.0 is based on WordNet 3.0. WordNet can be used for generating the initial seed words, since some of the words already have a positive or negative label. Lexicon expansion using WordNet or other similar lexical resources also requires a set of seed words. To expand the lexicon, in the first iteration the synonyms and antonyms of the seed words are identified and added to the lexicon. Synonyms are assigned the same label as the seed word, whereas antonyms are assigned the opposite label. In the following iterations, the synonyms and the antonyms to the words found in the previous iteration are added to the lexicon. A disadvantage of this approach is that the lexical resource used to expand the sentiment lexicon has a finite set of words, and a limited amount of relations, so the size of the lexicon can only increase proportional to the amount of relations in the resource.

SentiWordNet (Esuli and Sebastiani, 2006) contains all of the 115,000 synsets from the lexical database WordNet 2.0. Each synset is associated with a positive, neutral, and negative score, which sum up to 1. The lexicon is created by first identifying a set of positive and negative seed words, and adding words to the lexicon by extracting synonyms and antonyms from WordNet, within a certain distance from the original seed words. A list of neutral words is then created and added to the lexicon. All of the synsets in the lexicon are represented by their "gloss", which is the definition of the synset. Multiple classifiers are then trained on these glosses, and in the next step, the classifiers are used to predict the labels of the rest of the synsets in WordNet. Labels are assigned to the new synsets, by averaging the predicted labels across the different classifiers. The new synsets along with the assigned labels are then added to the sentiment lexicon.

Using a similar approach for sentiment lexicon creation, S.-M. Kim and Hovy (2004) investigated the semantic orientation of a text and identified the holder of the semantic orientation, given a topic and texts about this topic. They selected 23 positive and 21 negative verbs, and 15 positive and 19 negative adjectives as seed words, and expanded the lexicon by extracting all of their synonyms (and also the antonyms of adjectives) from WordNet in two iterations. The resulting lexicon consisted of 5,880 positive adjectives, 6,233 negative adjectives, 2,840 positive verbs, and 3,239 negative verbs. They also calculated the strength of the polarity by assuming that if a word has many synonyms occurring in the same class, the strength of the polarity is higher. The sentiment lexicon is then used to predict the sentiment of a text, by identifying all of the individual words in the text that are also present in the lexicon. The individual sentiment-bearing words are used to predict the sentiment of the full text. The lexicon was evaluated in a classifier to identify the positive or negative opinions of a topic holder. The accuracy ranged from 67.0 when the holder of the topic had been detected automatically by a named entity tagger, to 81.0 when the holder of the topic had been manually labeled (S.-M. Kim & Hovy, 2004). They suggest in their paper that sentiment

classification can be done more efficiently when the holder of a topic is close to the topic in the text, which is why they provide the holder when classifying sentiment.

2.3.2 Using methods for pattern matching for lexicon creation

Another approach for sentiment lexicon creation, consists of extracting sentiment-bearing words from a corpus, using methods for pattern matching. Hatzivassiloglou and McKeown (1997) developed an algorithm for creating a sentiment lexicon by extracting adjectives from the 1987 Wall Street Journal corpus.[3] The first step of their method consisted of extracting all of the adjectives that appear more than 20 times in the corpus, and then removing the ones that were not consistently positive or negative in different contexts. The extracted adjectives were manually labelled negative or positive. In the next step, they identified different patterns that represented different lexical relations. Adjectives conjoined by words like "and" are assumed to be of the same polarity, whereas adjectives conjoined by words like "but" are assumed to be of opposite polarity. They also identified two other patterns relevant for classification of polarity, which were the form of the modified noun (singular or plural) and the type of modification of the noun. Then they extracted all of the adjectives that were linked by a conjunction in the corpus. A log-linear regression model predicted the orientation of the extracted adjectives based on the identified patterns, and created a graph with edges between the conjoined adjectives representing either same polarity or opposite polarity.

The set of manually labelled seed words were used to train a classifier to predict the semantic orientation of the adjectives in the corpus. A clustering algorithm was applied on the graph to divide it into two clusters: one positive and one negative. In the last step of their method, they label the clusters positive or negative, assigning the group with the highest average frequency a positive label. This labelling scheme is based on their previous work (Hatzivassiloglou & McKeown, 1995), which found that two groups of opposite adjectives were different in terms of frequencies, and that the group of positive adjectives almost always had higher average frequencies than the group of negative adjectives. The accuracy of the algorithm for correctly labelling the polarity of adjectives ranged from 78.0 to 92.0. The accuracy correlated with the amount of conjunctions in the test set; test sets with few conjunctions achieved the lowest accuracies, whereas test sets with more conjunctions achieved higher accuracies.

2.3.3 Extracting sentiment lexicons from reviews

Sentiment lexicons can be created in a fully supervised manner, using data that has been manually labelled positive or negative. Different kinds of reviews can be used as a resource for lexicon generation, because the author of the review also assigns a label, or gives a numerical rating along with the review (Jurafsky & Martin, 2018). The

[3] https://catalog.ldc.upenn.edu/ldc2000t43

numerical score can be used to measure the strength of polarity of the review. This approach to sentiment lexicons is based on the assumption that positive words are more frequent in positive reviews, and negative words are more frequent in negative reviews (Jurafsky & Martin, 2018). Pang and Lee (2008) suggest that each rating on a numerical scale is associated with a distinct vocabulary. Different algorithms can be used to create sentiment lexicons from reviews. The complexity of these algorithms ranges from calculating raw counts of occurrences of words in the different classes, to more sophisticated methods, such as calculating the probability of a word belonging to each class (Jurafsky & Martin, 2018), or calculating Potts scores for all of the words in the reviews (Potts, 2011).

Potts (2011) used the review databases "imdb.com" and the "experienceproject.com" to study the relationships between words and sentences in the reviews and their ratings. His main goal was to examine how negation was perceived negatively in all contexts, but his methods and the calculations for scoring words can be applied in different domains, for example in the supervised creation of sentiment lexicons. For each word in the reviews, he calculates the normalized likelihood of each word belonging to the different classes of star-ratings in the IMDB reviews, or the reaction categories in the Experience Project. Equation 2.1 presents the probability of a word given a class, whereas Equation 2.2 presents the probability distribution of classes given a word. These equations are from Potts (2011, p. 639–640).[4] In these equations, the letter T represents the corpus, the letter c represents a class, the set of all linguistic types is represented by the letter Π, whereas a specific linguistic type is represented by the letter π.

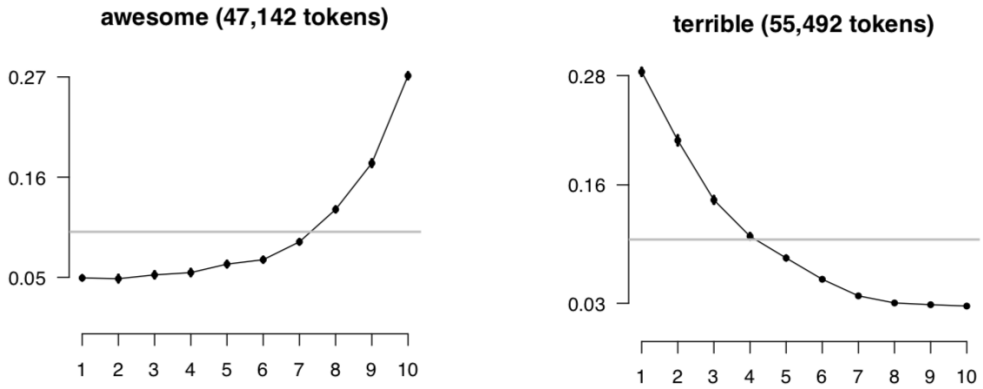
$$Pr_{T,\Pi}(\pi \mid c) = \frac{Count_T(\pi, c)}{Count_{T,\Pi}(c)} \tag{2.1}$$

$$P_{T,\Pi}(c \mid \pi) = \frac{Pr_{T,\Pi}(\pi \mid c)}{\sum_{c^J \in C} Pr_{T,\Pi}(\pi \mid c^J)} \tag{2.2}$$

The distribution of Potts scores representing each word can be visualised in a graph, which has different shapes depending on the degree of positivity and negativity. Figure 2.1 presents the graphs for the words "awesome" and "terrible". The shape of strongly positive words resemble the letter J, whereas the shape of the strongly negative words resemble a reversed J (Potts, 2011).

Another method for sentiment analysis using reviews, was developed by Turney (2002). In this work, he only performs sentiment analysis, and does not create a sentiment lexicon, however, his method can be used for lexicon creation. The algorithm for sentiment analysis developed by Turney (2002) labels movie reviews and auto mobile reviews positive or negative based on the adjectives and the adverbs in the

[4] https://web.stanford.edu/~cgpotts/papers/potts-salt20-negation.pdf



(a) "awesome" (b) "terrible"

Figure 2.1: Visualization of Potts scores for a) the positive word "awesome" and b) the negative word "terrible". The images are from Potts (2011, p. 648).

review. The first part of the algorithm performs part-of-speech tagging on the reviews, and extracts all of the sentences that contain adjectives and adverbs. Bigrams are then created, containing an adjective or an adverb, along with another word that provides contextual meaning. The phrase is assigned a score, which corresponds to the Pointwise Mutual Information (PMI) between the extracted phrase and the two seed words "excellent" and "poor". The PMI measures how often two events occur together, compared to how often they would occur together if they were independent of each other (Church and Hanks 1989). The function for calculating the PMI between two words from Turney (2002), is defined in Equation 2.3.

$$PMI(word_1, word_2) = \log_2 \left(\frac{P(word_1 \,\&\, word_2)}{P(word_1)\, P(word_2}) \right) \tag{2.3}$$

The average score for all the extracted phrases will determine the semantic orientation of the review, and its magnitude. The accuracy of the classification algorithm created by Turney (2002) ranged from 65.83 on movie reviews to 84.00 on auto mobile reviews. The focus of his work is on sentiment analysis using reviews for classification, and not on the creation of sentiment lexicons. Even though he does not create sentiment lexicons in this work, his methods for extracting sentiment-bearing words from reviews can be used for this task.

2.3.4 Creating sentiment lexicons from word embeddings

The distributional hypothesis states that semantically similar words occur in similar contexts (Firth, 1957; Harris, 1954; Rubenstein and Goodenough, 1965). This hypothesis forms the basis for the concept of word embeddings. A word embedding model is

created by examining the nearest words for each word in the corpus, and creating and adjusting their vectors so that similar words have similar representations in the model (Mikolov, Chen, Corrado, & Dean, 2013). In 2003, the concept of word embeddings was introduced by Bengio, Ducharme, Vincent, and Jauvin (2003), but they did not become popular until the release of the tool Word2Vec by Mikolov et al. (2013). There are two main algorithms for creating word embedding models with the Word2Vec tool: CBOW and Skipgram. The CBOW algorithm aims to predict the current word based on the context words, whereas the Skipgram algorithm predicts the context words based on the current word (Mikolov et al., 2013).

Word embeddings are dense representations of words, each word is represented by a n-dimensional vector of floating point numbers in the range between -1 and 1. The most common values for the dimensionalities are 100, 300, and 600, however, values above and below this range can also be selected. A word embedding model with a dimensionality of 100, represents each word by a vector of 100 floating point numbers. The aim of word embedding models is to represent semantically similar words with similar vectors. In addition to capturing similarity between similar words, they also provide a more efficient way to represent textual data, because each word is represented by a dense vector and not by for example a sparse one-hot vector. Using word embeddings instead of one-hot vector representations of data can greatly reduce the dimensional space. Another tool for creating word embedding models is the fastText tool, developed by Bojanowski, Grave, Joulin, and Mikolov (2017). The fastText tool creates vectors representing character n-grams, which allows the model to infer vector representations for unseen words. In the following paragraphs, we will describe two different approaches to sentiment lexicon creation using word embedding models.

Hamilton, Clark, Leskovec, and Jurafsky (2016) created a domain-specific sentiment lexicon called SentProp[5] from word embeddings trained on unlabelled corpora in a label propagation framework. The first part of their method consisted of manually identifying a set of positive and negative seed words. Then they created semantic vectors containing information about word embeddings using a vector space model. In the next step they created a lexical graph, the edges in the graph are represented as the cosine similarity between the node and its K-Nearest Neighbours. To calculate the semantic score of a word, a label propagation algorithm performs a random walk from the set of seed words to each of the words in the graph, and assigns a sentiment score which represents the probability value of each random walk. The random walk is performed on the positive and the negative set of seed words, and the scores from the two walks are combined into a polarity score. The sentiment lexicon created by Hamilton et al. (2016) using the domain-specific word embeddings, outperformed other state-of-the-art lexicons on sentiment classification tasks.

[5] https://github.com/williamleif/socialsent

In a recent study, Wang and Xia (2017) created a sentiment lexicon by combining information from word embeddings with sentiment labels of words and documents using a neural architecture. Their sentiment lexicon is available online.[6] Their method consisted of annotating the polarity of words and documents, then learning the word embeddings, and finally constructing a sentiment lexicon. Word-level sentiment labels are learned by computing the PMI between the words in the labelled documents and the positive and negative class labels. The positive and negative polarity score is calculated by subtracting the PMI for the negative label from the PMI for the positive label. The polarity is represented both as "hard sentiment annotation" which means that the class with the highest score obtains a score of 1, whereas the class with the lowest score obtains a score of 0, and as a "soft sentiment annotation" which is the probability of the score belonging to each class.

The next step of their method consisted of training a neural model to learn sentiment-aware word embeddings. The sentiment lexicon is created by hand-labelling 125 positive and 109 negative seed words, and then extending this set of seed words using a K-Nearest Neighbour clustering algorithm on Urban Dictionary.[7] The extended set of seed words, as well as the sentiment-aware word representations, are used to train a logistic regression classifier to predict sentiment scores. This classifier is used to create the sentiment lexicon, by predicting the sentiment scores of the input words from the corpus Sentiment140.[8] Sentiment140 is a sentiment dataset that contains tweets and associated labels, and can be used as a tool for sentiment analysis. The labels in the dataset were automatically assigned based on emoticons present in the tweets.

The sentiment lexicon created by Wang and Xia (2017) was evaluated on the SemEval 2013–2016 datasets, which are semantic datasets used for the International Workshop on Semantic Evaluation. Supervised classification is performed by extracting pre-defined lexicon features using lexical information from tweets from Sentiment140, such as computing the total count of words in a tweet that has a sentiment score of more than 0 or less than 0. Using pre-defined lexicon features for lexicon evaluation, the classifier obtained an average score of 72.0. Unsupervised sentiment classification is performed by calculating the sum of all the scores present in the sentiment lexicon and in the document. The document is positive if the score is larger than 1, and negative if the score is less than 0. The evaluation on unsupervised sentiment classification achieved an average accuracy of 77.0. Both supervised and unsupervised classification had better performance than other state-of-the-art approaches for creating sentiment lexicons.

In this section, we have presented various methods for sentiment lexicon creation, which have been successful for creating sentiment lexicons for the English language.

[6] https://github.com/NUSTM/HSSWE
[7] www.urbandictionary.com
[8] http://help.sentiment140.com

We have described different approaches, such as manually creating lexicons, extracting sentiment lexicons from reviews, or identifying a set of seed words to use for lexicon expansion. We have also described various resources that can be used to increase the size of the sentiment lexicon, for example word embedding models, or lexical resources containing word relations such as WordNet or a thesaurus.

2.3.5 Available sentiment lexicons for English

In this section, we will describe some of the most well-known English sentiment lexicons created using some of the approaches described in the previous section.

The General Inquirer is the oldest documented sentiment lexicon, created by P. Stone, Dunphy, Smith, and Ogilvie (1968). The system was created to extract information about content in a text, originally used in the field of behavioral science at Harvard, but it also had other applications, such as for sentiment analysis (P. J. Stone and Hunt, 1963). The lexicon is human annotated, and has been extended since it was first published. The most recent version of the lexicon consists of 2,291 negative and 1,915 positive words.

SentiWords is sentiment lexicon created by Gatti et al. (2016). They trained a regression classifier on a manually annotated sentiment lexicon, created by Warriner, Kuperman, and Brysbaert (2013). The classifier was then used to score lemmas from SentiWordNet 3.0. The lemmas and their new scores, along with the manually annotated lexicon, were merged into a new sentiment lexicon called SentiWords. It contains roughly 155,000 words, and each word is assigned a score between -1 (negative) and 1 (positive).

The Affective Norms for English Words (ANEW), by Bradley and Lang (1999) is a sentiment lexicon that was created as a standardized tool for researchers to study emotions. The lexicon was created by first identifying a set of words, and then asking students to manually label the words according to three dimensions, which were pleasure, arousal, and dominance.

The MPQA Subjectivity Lexicon is another well-known sentiment lexicon created by Wilson, Wiebe, and Hoffmann (2005) using the MPQA opinion corpus. The lexicon consists of words that have been manually labelled positive or negative, as well as part-of-speech tags, and information about strength and lemma.

2.4 Sentiment lexicons for Norwegian

The methods that were used to create English sentiment lexicons can also be used to create sentiment lexicons for the Norwegian language, as long as an appropriate lexical resource in Norwegian exists. To the best of our knowledge, there have only been three studies in which sentiment lexicons for Norwegian have been generated. We will first present the methods used to create these lexicons, and then describe the approach used

for the evaluation.

2.4.1 Creating sentiment lexicons from a corpus

One of the methods used for sentiment lexicon creation for Norwegian is based on the method by Turney (2002), which was described in Section 2.3.3. Hammer, Yazidi, Bai, and Engelstad (2014) created sentiment lexicons for the Norwegian language from two different corpora. All of the sentiment lexicons created in this study are available online.[9] The first step of their method consisted of manually labelling a set of 7 positive and 7 negative seed words, which according to the authors were unambiguous across contexts. They used "Aviskorpuset" *(The Norwegian Newspaper Corpus)*,[10] which is a large corpus of news articles with more than 1 billion words, and they also used a crawler to collect posts from forum discussions from "kvinneguiden.no" and "gamer.no". The resulting forum corpus consisted of 130 million words. Instead of using all of the words in each corpus as candidate words for lexicon expansion, they used two different sets of candidate words. The first set was the 10,000 most frequent words in each corpus. To create the second set of candidate words, they first extracted 50,000 adjectives from the full-form word list SCARRIE.[11] The second set of candidate words included the 10,000 adjectives that were present in both SCARRIE and in the corpora. To create the sentiment lexicon, they calculated the PMI between the seed words and the candidate words, and this score represented the magnitude of the sentiment.

2.4.2 Using resources containing lexical relations for lexicon creation

In another study, Hammer, Bai, Yazidi, and Engelstad (2014) created many different sentiment lexicons using different thesauri and dictionaries. The first step of their method consisted of manually identifying 51 positive and 57 negative seed words, that according to the researchers were frequently used in the Norwegian language, consistently positive or negative across contexts, and represented different strengths of positivity and negativity. In the next step, they used a crawler to extract synonyms and antonyms from three different thesauri, and create a graph of these words. The crawler first extracts the synonyms of the seed words, and in the next two iterations it extracts the synonyms of the words found in the previous iteration. A label propagation algorithm is applied on the graph to determine the strength and the polarity of the words. The positive seed words are assigned a score of 1 whereas the negative seed words are assigned a score of -1. The edges are also assigned a score. If the connected nodes are synonyms, the edge is assigned a score of 1, whereas if the connected nodes are antonyms, a score of -1 will be assigned. The non-seed words are then scored by calculating the weighted average of the connected nodes. The algorithm

[9] https://github.com/aleksab/lexicon

[10] https://www.nb.no/sprakbanken/show?serial=sbr-4&lang=nb

[11] https://www.nb.no/sprakbanken/show?serial=oai:nb.no:sbr-9&lang=

iteratively updates these scores until the changes in the scores are below a certain threshold. After the sentiment lexicon has been created, they used the full-form word list SCARRIE to expand the lexicon to include all forms of each word, by assigning the same positive or negative score to each word form. The lexical resources used for creating sentiment lexicons were "Norske Synomymer" *(Norwegian Synonyms)*,[12] "Norsk Ordbok" *(Norwegian Dictionary)*,[13] and "Din Ordbok" *(Your Dictionary)*.[14] They also translated the English sentiment lexicon AFINN[15] which was created by Nielsen (2011) to Norwegian, using Google Translate and manually removed words that they did not think were representing polarity correctly in Norwegian. They used this machine-translated lexicon as a baseline for comparison to the other sentiment lexicons that they created. However, the resources generated in this work (i.e. the sentiment lexicons) are not freely available, which makes the work very difficult to reproduce.

2.4.3 Creating domain-specific sentiment lexicons

In a third study by the same authors, Hammer, Yazidi, Bai, and Engelstad (2015) adjusted scores in a Norwegian sentiment lexicon by joining information from domain-specific corpora. The authors suggest that sentiment scores are highly sensitive to a domain, so a domain-specific sentiment lexicon would outperform a non-domain-specific sentiment lexicon for sentiment prediction. They also attempt to take negation into account, because negation can shift the sentiment of a text. First, the authors created a sentiment lexicon with information from all of the sentiment lexicons they had created in previous studies (Hammer, Bai, et al., 2014; Hammer, Yazidi, et al., 2014). Then they created three different versions of the sentiment lexicon by joining information from a domain-specific corpus with the lexicon, by minimizing the posterior expected loss on a sentiment prediction task on reviews. The loss function consists of two terms, the first term measures the loss from the sentiment lexicon, and the second term measures the loss from the domain-specific corpus. In their first approach, they only use the information from the first term, which only measures the loss from the sentiment lexicon on the evaluation dataset. In the second approach, only the information from the domain-specific corpus was used to calculate the loss on the evaluation set. In their third approach, they use both terms in the loss function, to calculate the weighted average of the loss from the lexicon and from the domain-specific corpus on the evaluation set. The sentiment lexicon that used the weighted average of the loss from both the sentiment lexicon and the domain-specific corpus had a slightly lower mean absolute error than the other two lexicons that they created. The domain-specific corpora used in the study were a collection of 15,118 reviews from "komplett.no" and "mpx.no". Since these reviews were randomly selected for this study, the data is not available for reproduction.

[12] www.ordnett.no
[13] www.ordnett.no
[14] https://www.dinordbok.com
[15] https://github.com/fnielsen/afinn

2.4.4 Evaluation of the Norwegian sentiment lexicons

All of the sentiment lexicons created by Hammer, Yazidi, et al. (2014), Hammer, Bai, et al. (2014) and Hammer et al. (2015) were evaluated on randomly retrieved movie reviews from "filmweb.no" and product reviews from "komplett.no". Since the evaluation was performed on reviews retrieved from websites by a crawler, and not on an existing dataset, the data is not available for reproduction. To evaluate their lexicons, they calculated the sum of the sentiment scores for each sentiment-bearing word in the review, and divided the sum by the number of words in each review. The review was then assigned a rating based on the score obtained in the above calculations. They compared the predicted rating to the actual rating of the review, and calculated the error in classification. The baseline system based on the machine translation of the sentiment lexicon AFINN (Nielsen, 2011) achieved the best average performance across all three studies. Most of the sentiment lexicons and the datasets which were used for evaluation of their lexicons have not been made publicly available, so a comparison between their lexicons and the sentiment lexicons created in our work will not be possible. There is a lack of existing sentiment lexicons in Norwegian, which is why the purpose of this project is to explore different methods for the automatic creation of sentiment lexicons. It is important that our sentiment lexicons, as well as the datasets and resources used for both creation and evaluation of lexicons are freely available.

In this section, we have presented previous research on creating sentiment lexicons using different approaches for Norwegian. We have seen that a machine-translated sentiment lexicon generally outperformed lexicons created using traditional methods for lexicon creation for the Norwegian language. Since there is a limited amount of lexical resources available for sentiment lexicon creation for Norwegian, we are also going to explore methods for improving the existing lexical resources. In the following section, we will present a description of convolutional neural networks, which can be used to improve word embedding models used for sentiment lexicon creation.

2.5 Convolutional neural networks

A Convolutional Neural Network (CNN) is a neural architecture originally developed for image classification. The first modern CNN, called LeNet-5, was developed by LeCun, Bottou, Bengio, and Haffner (1998) and outperformed the state-of-the-art systems for handwritten digit recognition. Collobert and Weston (2008) pioneered the use of convolutional neural networks for Natural Language Processing, using CNN's to perform a variety of NLP tasks, such as semantic role labelling, part-of-speech tagging, and chunking. CNN's became popular for sentiment analysis in 2014, after Y. Kim (2014) and Kalchbrenner, Grefenstette, and Blunsom (2014), demonstrated the performance of these architectures on sentiment analysis tasks. Convolutional neural networks can learn to extract n-grams and detect dependencies over multiple words.

2.5.1 The convolutional neural architecture

Convolutional neural networks can have many applications in NLP, for example in sentiment analysis, named entity recognition and text categorization. A lot of research in the field of convolutional neural networks for sentiment analysis has gone into identifying the optimal configuration of the parameters of the network. In the following paragraphs, we will describe the top configuration by Y. Kim (2014), which we will use as a baseline CNN configuration in Chapter 5.

The convolutional layer An essential feature of convolutional neural networks is the convolutional filter, which can be one or multiple filters that are applied on a subset of the data. These filters produce feature maps from the sliding dot-product between the filter and the data. Many different parameters have to be selected in a convolutional layer. The optimal configuration by Y. Kim (2014) used three different filter sizes, of size 3, 4, and 5. A filter of size 3 is expected to learn representations over 3 words. The CNN by Y. Kim (2014) achieved the best overall results when using 100 filters for each filter size, although this value varied for different kinds of tasks and datasets. The purpose of having many filters, is that each filter learns to recognize a different feature in the data. All of the parameters in the convolutional layer affect the training time of the network. The training time increases with larger filter sizes, and larger number of filters used for each filter size.

The pooling layer Convolutional filters are applied to the data and produce many feature maps which quickly increase the parameter space. A pooling operation is applied on the feature maps to reduce each feature map into a single number, and produce a fixed-size representation of the input. Two main pooling strategies are used in convolutional neural networks: max pooling and average pooling. In average pooling, the average value is calculated from each feature map and passed on to the next layer in the network. Valuable information about extreme values in the feature map can be lost when using average pooling. The pooling strategy used by Y. Kim (2014) is max pooling, which only passes the highest value from each feature map to the next layer. Zhang and Wallace (2017) found that average pooling performed considerably worse than max pooling and that the training time increased when using average pooling in a convolutional neural network for sentiment analysis.

Regularization strategies Dropout is a regularization strategy used in neural networks, which drops random nodes at a certain rate. It was developed by Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov (2014). Y. Kim (2014) used a dropout rate of 0.5 in the penultimate layer. Zhang and Wallace (2017) found that low rates of dropout from 0.1 to 0.5 can have a positive, but very small impact on the performance of the system. They also found that larger dropout rates may have a positive impact on networks with larger number of filters, to prevent them from overfitting.

However, they argue that in general, lower rates of dropout have little effect on the performance of CNN's, whereas higher dropout rates can greatly lower the performance of the network. T

Activation functions One of the main features of neural networks is that they can learn to approximate any function. They can use non-linear activation functions to learn non-linear decision boundaries, in contrast to linear classification algorithms. A neural network achieves this by passing the values in the nodes through a non-linear activation function, which transforms the values into a specific distribution, depending on which activation function is used. The best configuration by Y. Kim (2014) uses two different activation functions, the Rectified Linear Unit (ReLU) and the Softmax activation functions. The ReLU activation function, which is presented in Equation 2.4, is used in all of the layers of the network, except for the output layer.

$$ReLU(x) = max(0, x) \tag{2.4}$$

The ReLU activation function is a simple and widely used activation function that transforms negative numbers into 0, and positive numbers remain unchanged. The other activation function used in the CNN is the Softmax function which is presented in Equation 2.5.

$$Softmax(z)_j = \frac{e_j^z}{\sum_{k=1}^{K} e_k^z} \tag{2.5}$$

The Softmax activation function is often used in the output layer of networks and can handle multi-class, as well as binary prediction. This activation function takes all the input values to the output layer, and transforms them into a categorical probability distribution. All the values in the nodes in the output layer are then represented as probabilities of the input data belonging to the different classes.

Loss function The function for measuring the error in the predictions of the network is the loss function. The loss function used by Y. Kim (2014) is the categorical cross entropy loss, which is presented in Equation 2.6.

$$Loss(p,\, q) = -\sum_{x} p(x)\, \log q(x) \tag{2.6}$$

The cross entropy loss is a calculation of the loss between two probability distributions. In a neural network, the loss is the difference between the distribution of predicted labels, and the distribution of correct labels. Since the Softmax activation function produces a probability distribution over the predicted classes, this activation function works very well with the cross entropy loss function in a neural network. The cross entropy loss function greatly penalises high incorrect probabilities, whereas it only mildly penalises small incorrect probabilities.

Optimizer The optimizing algorithm, which was described in the previous paragraph, is used to update the weights of the network with the objective to minimize the loss. One of the parameters of the optimizer is the learning rate, which is a measure of how quickly the network learns. Small learning rates cause small updates to the weights in the network, and training times can become very long. A large learning rate causes large updates to the weights, which can lead to a network that does not converge. The optimization algorithm used by Y. Kim (2014) is an adaptive optimizer called Adadelta, which was developed by Zeiler (2012). The learning rate used by Adadelta is adjusted throughout the training of the network.

Word embedding models as input to the CNN In sentiment classification tasks, the data that is used for training and prediction in a network is textual data. Neural networks can only process integer inputs, so the texts have to be transformed into an integer representation. There are different methods that allow us to transform text into numbers, such as for example encoding the text into one-hot vectors or transforming the text into its representation in a word embedding model. In a CNN, word embedding models are used in the embedding layer, which is a layer that transforms words into their corresponding vectors. The embedding layer consists of a matrix, in which the index of the rows corresponds to the id of the word, and the rows correspond to the vector representation of a each word. The words in the word embedding model are assigned an id based on their frequency in the corpus the model was trained on.

Embedding layers can either be kept static or dynamic during the training of the network. A static embedding layer is a layer with fixed weights, the weights are not updated during training, whereas in a dynamic embedding layer, the embedding weights are updated and trained along with the network. Dynamic word embeddings can be used to continue training the word vectors, which can be beneficial if we want to train the vectors on a specific dataset. Dynamic word embeddings require longer training times than static word embeddings in a neural network.

2.5.2 Fine-tuning word embeddings in a CNN

Convolutional neural networks can be used to fine-tune word embedding models on sentiment data, which is the main application of CNN's in this work. When a neural network is trained, all of the weights in the network are updated, to minimize the prediction error in the network. Word embedding models that are trained in a dynamic embedding layer, can be saved as a new word embedding model after the network has finished training. The goal of fine-tuning the word embeddings is to get more accurate representations for the sentiment-bearing words in the models. Words in word embedding models have similar vector representations if they appeared in the same contexts in the corpus that was used for creating the word embedding model. Since antonyms often appear in the same contexts (Charles and Miller, 1989), these words

tend to have similar representations in the models. Y. Kim (2014) describes a method for fine-tuning word embedding models in a convolutional neural network, which makes vector representations for certain semantically unrelated words less similar in the model. Figure 2.2 presents the top neighbouring words of the words "good" and "bad" in a word embedding model before and after fine-tuning in a CNN.

	Most Similar Words for	
	Static Channel	Non-static Channel
bad	*good*	*terrible*
	terrible	*horrible*
	horrible	*lousy*
	lousy	*stupid*
good	*great*	*nice*
	bad	*decent*
	terrific	*solid*
	decent	*terrific*

Figure 2.2: The top neighbouring words of antonyms, before and after fine-tuning the word embeddings in a CNN. The table is from Y. Kim (2014, p. 1750).

Y. Kim (2014) performed various classification tasks using a convolutional neural network. He first performs tuning experiments on the parameters of the network to find the optimal configuration for his task, then he experiments with different configurations for the CNN. One of the configurations is a multi-channel convolutional network, which is a CNN with multiple input channels. Each input channel takes a different representation of the input data, which is passed through a convolutional layer and summed before the pooling operation. Y. Kim (2014) created two different channels, using the same word embedding model in two different embedding layers, one static and one dynamic. He found that in the multi-channel set-up, the dynamic layer was able to change the representations for certain antonyms.

In this chapter, we have presented different methods for sentiment lexicon creation both for the Norwegian and the English language. In the following chapter, we will describe the available lexical resources that can be used to create Norwegian sentiment lexicons. In Chapter 4, we will use some of the approaches for lexicon creation that we have described in this chapter along with the lexical resources presented in Chapter 3, to create sentiment lexicons for Norwegian.

Chapter 3

Sentiment resources

As seen in the previous chapter, different kinds of resources for sentiment analysis can be used in the creation of sentiment lexicons. Lexical resources such as WordNet, dictionaries, and review datasets can be used both in the task of creating a sentiment lexicon and to perform sentiment analysis. Datasets containing reviews, such as movie and product reviews, can also be used as a tool for evaluating sentiment lexicons. Many studies have used online reviews, retrieved by a crawler (Hammer et al., 2015; Hammer, Bai, et al., 2014; Hammer, Yazidi, et al., 2014; Potts, 2011), but the reviews have to be freely available in order to reproduce the results. To be able to create sentiment lexicons for Norwegian, and for the evaluation results to be reproducible, we have to use datasets and lexical resources that are freely available. In this chapter, we will describe some of the available resources for sentiment lexicon creation and evaluation in Norwegian. We will also provide an overview of how these resources will be used in this work.

3.1 WordNet

The Norwegian WordNet can potentially be used as a resource for lexicon creation because it contains words that have been labelled positive or negative. The first version of the Norwegian WordNet was developed by Kaldera Språkteknologi AS on behalf of the Norwegian National Library, and is available online.[1] It is a collection of files in the .rdf file format, each file represents a different word relation, such as synsets, hypernyms, synonyms, and word senses. The Norwegian WordNet is based on the translation of DanNet[2] which is the Danish WordNet, developed by Pedersen et al. (2009). It contains roughly 65,000 synsets and 250,000 named entities as reported by Sand (2016). There is a lack of documentation on the original version of Norwegian WordNet, however, there is a newer version developed by Sand, Velldal, and Øvrelid (2017)[3] which is far better documented. This version only consists of 51,258 synsets,

[1] https://www.nb.no/sprakbanken/show?serial=sbr-27
[2] http://wordnet.dk/menu%3Fitem=2.html
[3] https://github.com/heisand/NWN

all of the named entities from the first version have been omitted. Since the Norwegian WordNet is based on DanNet, resources for DanNet are also relevant for the Norwegian WordNet, and have been consulted to gain an understanding of this lexical resource.

3.1.1 Sentiment labels in WordNet

A paper by Braasch and Pedersen (2010) examines the positive and negative connotations in DanNet and how these were labelled. Their focus is on the annotated nouns in DanNet, of which there are 506 negative and 144 positive. They have chosen to label word senses, i.e. synsets, and not individual words as positive or negative, because a word can have different labels in different contexts. The positive and negative labels in DanNet are manually labelled, and do not contain a strength of the polarity. The label assigned to a word sense is primarily based on how the word is defined in "den Danske Ordbog" *(the Danish Dictionary)*.[4] Some of the words in the dictionary contain information about whether the word is positive or negative, whereas for other words, the authors had to look for other cues, such as the polarity of the adjectives used in the definition of the word, in order to determine the polarity. If they still were not able to determine the polarity of the word, they searched the web and manually labelled the word as positive or negative according to the online opinions on the word. In DanNet, there are almost twice as many negative labels as there are positive labels. Approximately 78% of the labelled words are negative, whereas only approximately 22% of the labelled words are positive.

Among the synsets in the latest version of the Norwegian WordNet by Sand et al. (2017), 390 are labelled positive or negative. Sand et al. (2017) did not modify the file containing the positive and negative labels, so we had to extract the synsets that had been assigned a label, which were also present in the latest version of the Norwegian WordNet. Since the latest version contains fewer synsets than the original version, some of the labelled synsets (260) from the old version have been omitted. The distribution of labels in the Norwegian WordNet is similar to the distribution of labels in DanNet, there are 111 positive synsets (28.4%), and 279 negative synsets (71.6%). Most of the synsets in the Norwegian WordNet have been tagged with part-of-speech tags, and the majority of the labelled synsets are nouns, as in DanNet. Of the 390 labelled synsets in the latest version of the Norwegian WordNet, there are 279 nouns, 29 verbs, and 13 adjectives. There are 69 synsets that do not have a part-of-speech tag.

3.2 The Norwegian Review Corpus (NoReC)

The Norwegian Review Corpus (NoReC)[5] is a Norwegian corpus for document-level sentiment analysis, created by Velldal et al. (2017) as part of the ongoing SANT project.

[4] https://ordnet.dk/ddo
[5] https://github.com/ltgoslo/norec

It consists of 35,194 reviews, published between 1998 and 2017, which have been assigned a score between 1 and 6 by the author of the review. The rating distribution in NoReC is presented in Figure 3.1. Most of the reviews have a rating of 3–5, whereas few reviews are assigned the most extreme ratings (1, 2, and 6).

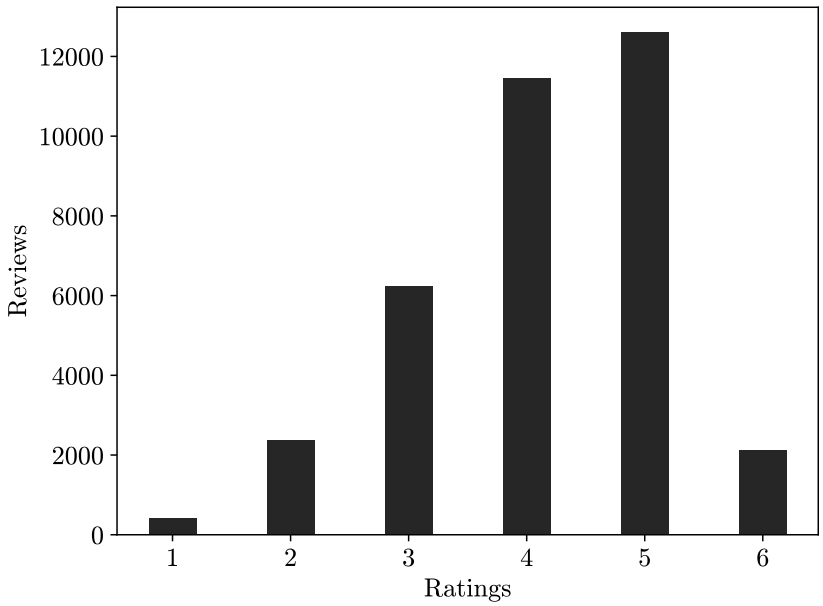


Figure 3.1: The distribution of ratings in the NoReC dataset.

NoReC was created by first identifying reviews provided by Schibsted Media Group, Aller Media, and NRK. The reviews were then converted to HTML in order to preserve the content of the review, but also to mark text that was not relevant. Each review was then pre-processed further and represented in the CoNLL-U format. Then they identified the language in the review, whether it was written in Nynorsk or Bokmål. NoReC is freely available and is distributed in both the CoNLL-U and HTML file format.

The category distribution is made up of nine different categories, which is presented in Figure 3.2. Since the focus of this work is on lexicon-based binary sentiment classification, reviews with similar ratings in the dataset are grouped together. The uneven distribution of positive and negative reviews can present problems for our project. Certain classifiers can learn domain-specific words and assign these words more weight even though they are not sentiment-bearing. If there is much more data in one of the classes, the classifier may be able to learn more features of the documents in that particular class, and make better and more accurate predictions for documents belonging to that class. There is also an uneven distribution of categories in the dataset,

many reviews belong to categories such as music and screen, whereas categories such as restaurants and stage have few reviews.

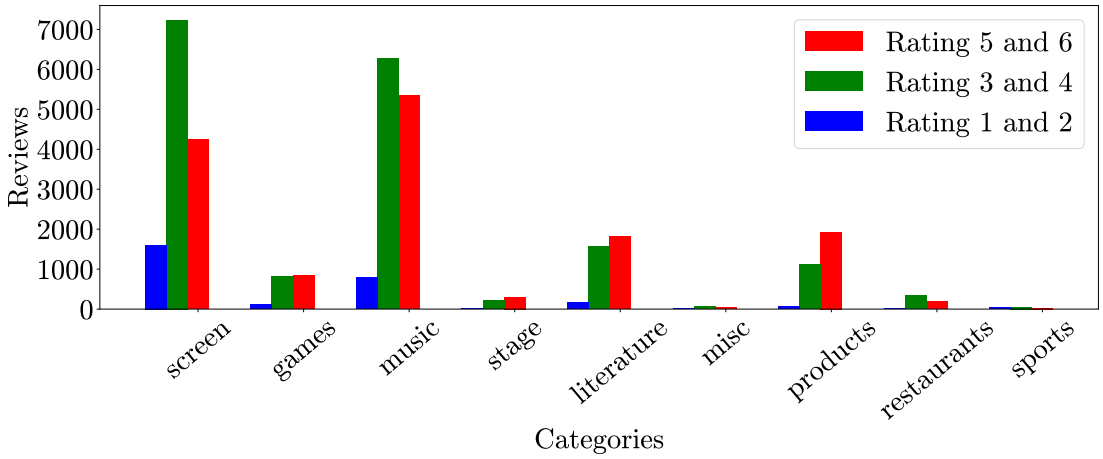


Figure 3.2: The distribution of categories in the NoReC dataset.

The reviews in NoReC have been split into a training set, a development set, and a test set, based on the publication dates of the reviews. The splits were created to facilitate reproduction of results from experiments on these reviews, and to avoid having reviews about the same movie or product in two different splits. The training split consists of 80% of the reviews with the oldest publication dates in the dataset. The development split contains 10% of the reviews in the dataset, which were published after the reviews in the training split. The third split is the test split, which consists of the 10% of the reviews in the dataset with the newest publication dates. NoReC can have a variety of applications in different sentiment analysis tasks. It can be used for sentiment lexicon creation using methods by Turney (2002) or Potts (2011), or it can be used in the evaluation of sentiment lexicons, using the same approach as Hammer, Yazidi, et al. (2014), Hammer, Bai, et al. (2014) and Hammer et al. (2015). NoReC can also be used for sentiment analysis in a convolutional neural network, using the approach by Y. Kim (2014) or Zhang and Wallace (2017).

We will use the different splits of NoReC for different tasks. The training split will be used to create sentiment lexicons using the approach by Potts (2011). It will also be used to create word embedding models, and to fine-tune word embedding models in a convolutional neural network. The development split will be used for the evaluation of sentiment lexicons. We will use the test split for the final evaluation of the top performing sentiment lexicons. Table 3.1 presents statistics of the NoReC.

In this work, we will focus on the classification of very positive and negative reviews, and not on mildly polarized reviews. For some of the tasks in this project, such as

the evaluation of sentiment lexicons, we will only consider reviews with the most extreme ratings. To get a fairly even distribution of positive and negative reviews, we have chosen to only include reviews with a rating of 1, 2, and 6, for these tasks. We have labelled the splits of NoReC, containing only reviews with a rating of 1, 2, and 6, NoReC$^{\pm}$ train, NoReC$^{\pm}$ dev, and NoReC$^{\pm}$ test. Table 3.2 presents statistics of the NoReC$^{\pm}$ dataset.

Split	# Pos	# Neut	# Neg	Tokens	Lemmas	Word forms	Avg. length
Train	11,597	14,235	2,326	11,484,648	373,657	436,602	408
Dev	1,569	1,719	230	1,737,928	98,858	120,963	494
Test	1,559	1,723	231	1,776,091	97,497	120,120	506
Total	14,725	17,677	2,787	14,998,667	438,306	511,150	426

Table 3.1: Statistics of the NoReC. The rows represent the different data splits in NoReC. For each split, we report the number of reviews with a rating of 5 and 6 (# Pos), a rating of 3 and 4 (# Neut), and a rating of 1 and 2 (# Neg), as well as the total amount of reviews of each rating. The tokens represent the total amount of tokens in the split, the lemmas represent the amount of unique lemmas in the split, and the word forms represent the amount of unique word forms in the split. The avg. length represents the average length of the reviews in each datasplit.

Split	# Pos	# Neg	Tokens	Lemmas	Word forms	Avg. length
NoReC$^{\pm}$ train	1,692	2,326	1,675,582	108,651	131,484	417
NoReC$^{\pm}$ dev	226	230	236,354	27,176	34,497	518
NoReC$^{\pm}$ test	204	231	224,811	25,802	33,011	517

Table 3.2: Statistics of the NoReC$^{\pm}$ dataset (ratings 1, 2, and 6). The rows represent the different data splits. For each split, we report the number of reviews with a rating of 6 (# Pos) and a rating of 1 and 2 (# Neg). The tokens represent the total amount of tokens in the split, the lemmas represent the amount of unique lemmas in the split, and the word forms represent the amount of unique word forms in the split. The avg. length represents the average length of the reviews in the datasplit.

3.2.1 The NoReC training split

The training split of NoReC will have various applications in this project. First, it will be used for sentiment lexicon creation using both NoReC and NoReC$^{\pm}$ for this task. Then, we will use the full NoReC training dataset in the creation of word embedding

models. Later in this work, we will use the training set of NoReC$^{\pm}$ in a convolutional neural network. We had to split the NoReC$^{\pm}$ training dataset into two new splits, a training set and a development set, since the original development and test sets were going to be used for evaluating the sentiment lexicons. Since the original NoReC splits depend on the dates of publication for the different reviews, the same approach was taken to split the training split into a training set and a development set. The 10% of the most recent reviews from each category of positive reviews, and the 10% of the most recent published reviews from each category of negative reviews were removed from the training split to form the development split. Table 3.3 presents some statistics of the new training and development splits extracted from the original NoReC$^{\pm}$ training split. We have called the new splits NoReC$^{\pm}$ CNN train and NoReC$^{\pm}$ CNN dev.

Split	# Pos	# Neg	Tokens	Lemmas	Word forms	Avg. length
NoReC$^{\pm}$ CNN train	1,522	2,093	1,478,684	100,187	121,419	409
NoReC$^{\pm}$ CNN dev	170	233	196,898	24,513	31,148	488

Table 3.3: Statistics of the NoReC$^{\pm}$ CNN training and NoReC$^{\pm}$ CNN development splits. The rows represent the different data splits. For each split, we report the number of reviews with a rating of 6 (# Pos) and a rating of 1 and 2 (# Neg). The tokens represent the total amount of tokens in the split, the lemmas represent the amount of unique lemmas in the split, and the word forms represent the amount of unique word forms in the split. The avg. length represents the average length of the reviews in the datasplit.

To test the performance of the new splits compared to the original splits, a convolutional neural network was implemented using the baseline architecture and the hyper-parameters by Y. Kim (2014), which was described in Section 2.5.1 in Chapter 2. First, the network was trained on NoReC$^{\pm}$ CNN train and tested on NoReC$^{\pm}$ CNN dev. Then, it was tested on the reviews in NoReC$^{\pm}$ dev. Finally, the network was trained on the reviews in NoReC$^{\pm}$ train and tested on the reviews in NoReC$^{\pm}$ dev. The results are presented in Table 3.4. The two first tests were done to see whether the new splits achieved similar results to the original splits, and to determine whether the new splits could be used for further experiments in a convolutional neural network. The three tests achieved similar results, and the differences can simply be attributed to the variance in the network. We will discuss variance in neural networks further in Section 5.1.1 in Chapter 5. Training the network using NoReC$^{\pm}$ CNN train, achieves almost the same accuracies on NoReC$^{\pm}$ CNN dev and NoReC$^{\pm}$ dev. In the final test, the entire NoReC$^{\pm}$ train was used for training the network, which contains 10% more training data than NoReC$^{\pm}$ CNN train. There is a slight increase in the performance of the classifier when 10% more training data is added, however, the increase in accuracy is small, and can be attributed to the variance in the network.

Training dataset	Testing dataset	CNN Accuracy
NoReC$^{\pm}$ CNN train	NoReC$^{\pm}$ CNN dev	92.05
NoReC$^{\pm}$ CNN train	NoReC$^{\pm}$ dev	92.32
NoReC$^{\pm}$ train	NoReC$^{\pm}$ dev	93.42

Table 3.4: Testing the NoReC$^{\pm}$ datasets in a CNN, using a pre-trained fastText Skipgram word embedding model of dimensionality 100 in the embedding layer.

There are several reasons why the classifier achieves such high accuracies. This is a binary classification problem, we only predict whether the review is positive or negative, and we do not predict the score in the review as in multi-class prediction. We are also only using the most extreme reviews for classification, the most positive and negative reviews. It should be much easier for a classification algorithm to predict the labels of reviews that are extremely positive or negative, then for example reviews with a less extreme rating, such as reviews with a rating of 3 or 4. Since we have chosen to create sentiment lexicons containing only positive and negative words, and not include neutral words, we also want to focus other aspects of this work on the same form of binary classification. We will use convolutional neural networks later in this work to try to improve the vector representations in word embedding models, which will be used for sentiment lexicon creation. The goal is that the extremely positive and negative reviews in NoReC$^{\pm}$ will contain strong signals that will improve the embeddings of the sentiment-bearing words in the models, which will hopefully lead to better sentiment lexicons.

3.2.2 The NoReC development split

The development split of NoReC$^{\pm}$ will be used to evaluate the performance of sentiment lexicons. There is almost an even distribution of positive and negative reviews in NoReC$^{\pm}$ dev, however, the distribution of categories in this subset of the dataset is uneven. The distribution of positive and negative labels across the different categories in the development split of NoReC$^{\pm}$ is presented in Figure 3.3. Because of the rating distribution in NoReC$^{\pm}$, and the uneven distribution of categories in the different classes, it is very difficult to obtain both an even distribution of positive and negative reviews, and an even distribution of categories. In the evaluation of the sentiment lexicons, we are going to count the frequencies of the positive and negative words from the sentiment lexicon present in the review that is being analysed, which is a very simple approach for lexicon evaluation. A review is labelled positive if there are more words from the positive side of the sentiment lexicon, and negative if there are more negative words. The uneven distribution of categories will therefore probably not have a large impact on the classification results. Other classifiers, such as neural classifiers, will to a larger extent be able to learn specific features from different categories, which

could influence the results to a larger degree than in our primitive evaluation method.

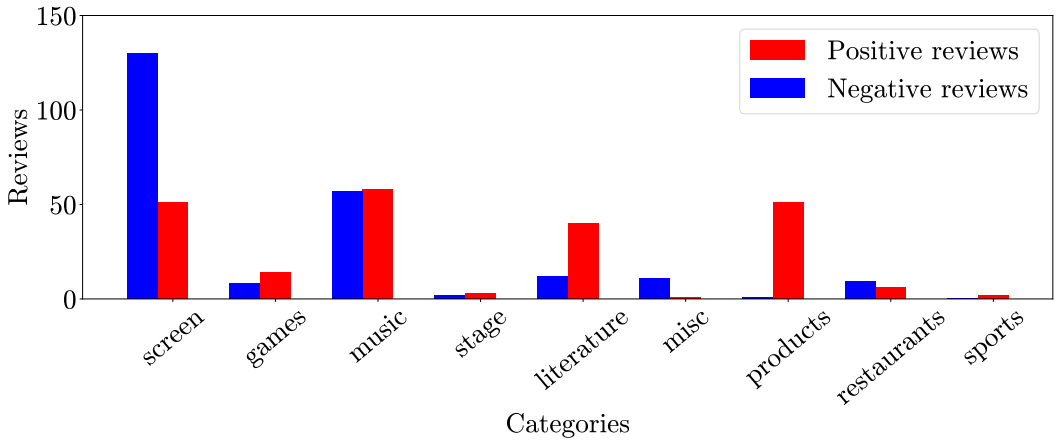


Figure 3.3: The distribution of categories in the development split of NoReC$^{\pm}$.

3.2.3 The NoReC test split

The reviews in the test split of NoReC$^{\pm}$ will be used for evaluation of the top performing lexicons in the final stages of this project. We will perform the final evaluation on the top performing lexicons created using different strategies, to measure how well they perform on unseen data. Since the lexicons are continuously evaluated on the development split of NoReC$^{\pm}$, and the lexicons are adapted and transformed based on these results, there is always a risk that we are overfitting the lexicons to this particular subset of the dataset. By evaluating the sentiment lexicons on a held-out test set, we can measure how well they perform on unseen data.

3.3 The Pros and Cons dataset

The Pros and Cons dataset is a collection of sentiment-bearing summaries from DinSide, currently being prepared as part of the SANT-project. All of the summaries in the dataset were published in the period between 2002 and 2017. Most of the summaries in this dataset are also in NoReC. NoReC contains the full version of the reviews, whereas the texts in the Pros and Cons dataset are short summaries which form part of these reviews, and contain polarized and compressed versions of the reviews, with the most important sentiment-bearing sentences. The summaries were published together with the reviews, and usually have one or a few sentences, or keyword-based fragments, associated with a thumbs up and a thumbs down. The positive sentences make up the positive part of the dataset (Pros) and the negative sentences make up the negative part of the dataset (Cons). The reviews in NoReC can therefore have 2 corresponding reviews in the Pros and Cons dataset. The dataset has

almost the same amount of positive and negative summaries, there are 3,703 positive and 3,658 negative summaries. Each summary has a title, corresponding to the item that is being reviewed, and a dice corresponding to the rating. The positive characteristics of the item are presented below the thumbs up icon and the negative characteristics are presented below the thumbs down icon. Figure 3.4 presents a summary from a computer review from DinSide.[6]

Figure 3.4: A summary from DinSide.

The category distribution of the Pros and Cons dataset consists of nine different categories, which are presented in Figure 3.5. The category distribution of the Pros and Cons dataset is quite different from the category distribution of NoReC. Most of the summaries in the Pros and Cons dataset are taken from product reviews from NoReC. The product category in the Pros and Cons dataset has been divided into different subcategories, which are the original categories that were assigned by the author of the review. A few of the summaries have not been assigned a category by the author, and therefore just been placed in a category by the creator of the dataset ("autofil" and "dinside"). Since each summary belongs to a single category, and the summary is divided into the Pros and the Cons parts of the dataset, there is an even distribution of Pros and Cons labels within each category. Some of the categories, such as "bolig" *(home)* and "data" *(data)* contain the majority of the summaries, whereas other categories only have a few, such as "reise" *(travel)* and "fritid" *(leisure)*.

The Pros and Cons dataset is much smaller than NoReC, however, NoReC$^{\pm}$ and the Pros and Cons dataset contain roughly the same number of labelled examples. Since most of the summaries in the Pros and Cons dataset are in NoReC, the original

[6] https://www.dinside.no/data/en-av-arets-aller-beste-pc-er/69983250

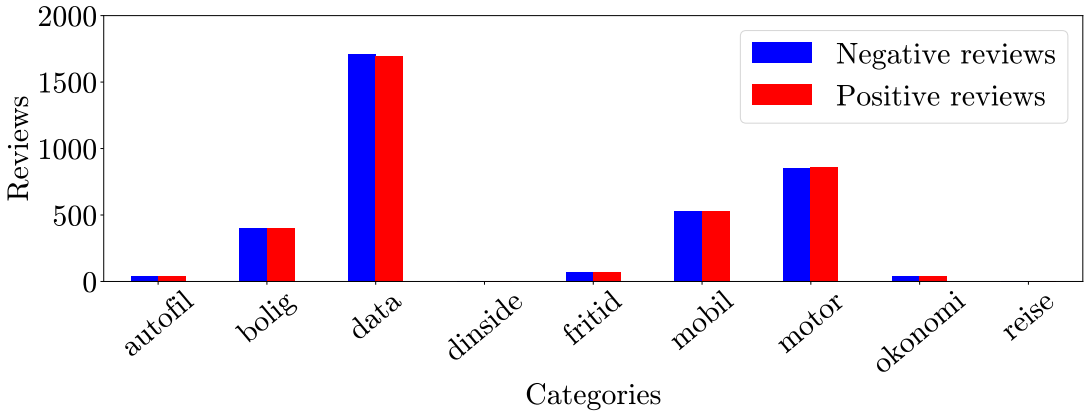


Figure 3.5: The distribution of categories in the Pros and Cons dataset. The categories are "autofil" *(cars)*, "bolig" *(home)*, "data" *(data)*, "dinside" *(dinside)*, "fritid" *(leisure)*, "mobil" *(mobile)*, "motor" *(motor)*, "okonomi" *(economics)*, and "reise" *(travel)*.

training, development, and test splits from NoReC can also be used for the Pros and Cons dataset. There are 1,546 summaries in the Pros and Cons dataset that are not in NoReC, which we have labelled the rest. Statistics of the Pros and Cons dataset are presented in Table 3.5. The average length of the summaries in this dataset is very different from NoReC, since the majority of these summaries consists of one or two sentences. Another main difference between the datasets is that there are almost 15 million tokens in NoReC, whereas only roughly 100,000 tokens in the Pros and Cons dataset.

Split	# Pros	# Cons	Tokens	Lemmas	Word forms	Avg. length
Train	2,362	2,360	64,267	6,868	8,448	14
Dev	259	263	9,221	1,971	2,419	18
Test	255	255	8,727	1,837	2,255	17
Rest	774	772	18,144	3,183	3,947	12
Total	3,650	3,650	100,359	9,261	11,385	14

Table 3.5: Statistics of the Pros and Cons dataset. The rows represent the different data splits. For each split, we report the number of summaries labelled Pros and the number of summaries with a Cons label. The tokens represent the total amount of tokens in the split, the lemmas represent the amount of unique lemmas in the split, and the word forms represent the amount of unique word forms in the split. The avg. length represents the average length of the summaries.

3.3.1 The Pros and Cons training split

The Pros and Cons training split will be used for two different tasks in this project. We will first use the dataset to create sentiment lexicons by extracting sentiment-bearing words from positive and negative summaries. Later in this work, the training split will be used to fine-tune word embedding models in a convolutional neural network. The training split consists of 4,646 summaries, and the distribution of categories is very similar to the distribution of categories in the entire dataset. Using the same approach for splitting the NoReC$^{\pm}$ training split in the previous section, we have separated Pros and Cons training dataset into a new training and a development split. The development split was created by extracting 10% of each category of positive and negative summaries from the training set, based on the publication date. We have labelled the new training split CNN train and the new development split CNN dev. We have also added the 1,546 summaries that were not in the NoReC dataset to the Pros and Cons CNN training split, to have more training data in the convolutional neural network. We will use the superscript $^{+}$ for the dataset that contains the reviews that were not in NoReC. Statistics for the Pros and Cons datasets used in a CNN are presented in Table 3.6.

Split	# Pros	# Cons	Tokens	Lemmas	Word forms	Avg. length
CNN train^{+}	2,898	2,893	74,179	7,659	9,418	13
CNN dev	238	237	8,232	1,725	2,126	17

Table 3.6: Statistics for the new Pros and Cons training and development split used in a CNN. The rows represent the different data splits. For each split, we present the number of summaries labelled Pros, and the number of summaries with a Cons label. The tokens represent the total amount of tokens in the split, the lemmas represent the amount of unique lemmas in the split, and the word forms represent the amount of unique word forms in the split. The avg. length represents the average length of the summaries.

The new splits were tested in a convolutional neural network using the hyperparameters by Y. Kim (2014), which were the same parameters that were used to test the NoReC$^{\pm}$ splits. For each of the training datasets, we tested both with and without adding the 1,546 summaries that were not in NoReC. The results are presented in Table 3.7. All of the configurations produce similar CNN accuracies. Two of the configurations benefited from adding more data to the training set, however, for one of the configurations, the accuracy decreased after adding more training data. The differences in the CNN accuracies are quite small, and can be attributed to the variance in the network, which we will describe in Section 5.1.1 in Chapter 5.

Training dataset	Testing dataset	CNN Accuracy
CNN train	CNN dev	96.42
CNN train$^+$	CNN dev	97.05
CNN train	Dev	97.50
CNN train$^+$	Dev	96.16
Train	Dev	95.00
Train$^+$	Dev	97.31

Table 3.7: The results of testing the different splits in the Pros and Cons dataset in the baseline CNN, using a pre-trained fastText Skipgram word embedding model of dimensionality 100 in the embedding layer.

3.3.2 The Pros and Cons development split

The Pros and Cons development split will be used in the evaluation of sentiment lexicons. All of the lexicons created in this work will be evaluated on the development splits of both NoReC$^{\pm}$ and the Pros and Cons. The development split of the Pros and Cons dataset consists of 259 positive and 263 negative summaries. The distribution of labels and categories in the Pros and Cons development set is presented in Figure 3.6. Only the five largest categories from the full dataset are present in the development split. The other four categories, "autofil" *(cars)*, "dinside" *(dinside)*, "okonomi" *(economics)*, and "reise" *(travel)*, only contain a few summaries each, and all of these summaries are in the training split of the dataset.

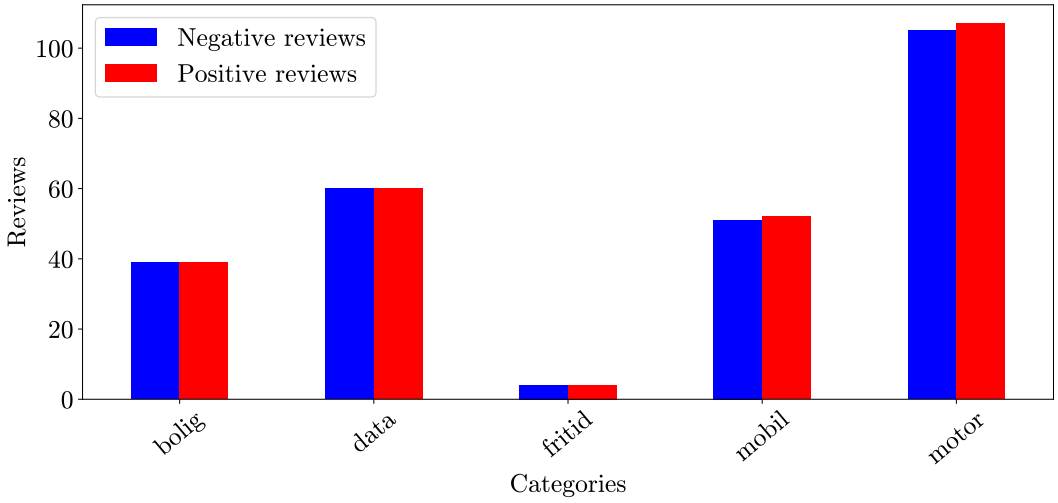


Figure 3.6: The distribution of categories in the development split of the Pros and Cons. The categories are "bolig" *(home)*, "data" *(data)*, "fritid" *(leisure)*, "mobil" *(mobile)*, and "motor" *(motor)*.

3.3.3 The Pros and Cons test split

The summaries in the test split of the Pros and Cons are the summaries corresponding to the reviews in the test split of NoReC. We are saving these summaries for the final evaluation of the sentiment lexicons. The distribution of categories in this subset of the Pros and Cons is very similar to the distribution of categories in the development split presented in Figure 3.6. The development split and the test split, which are presented in Table 3.5, are similar in terms of the number of reviews, the average length of summaries, and the number of tokens and types.

In this Chapter, we have presented different resources for sentiment lexicon creation and evaluation for the Norwegian language. The Norwegian WordNet, which is a resource containing lexical relations, will be used for the task of sentiment lexicon creation. The Norwegian Review Corpus and the Pros and Cons dataset will be used for various tasks, including the creation and expansion of sentiment lexicons, as well as the evaluation of sentiment lexicons. These datasets will also be used for sentiment analysis in a convolutional neural network.

Chapter 4

Developing sentiment lexicons

In this chapter, we will describe the methods for creation and evaluation of sentiment lexicons used in this work. Since there is not a standard method for sentiment lexicon evaluation, we have chosen a simple method that we believe is suitable for this project. We will also describe the two different approaches that we have experimented with for sentiment lexicon creation. First, we will describe a method for automatically extracting sentiment lexicons from labelled text. Later, we will explore methods for creating sentiment lexicons based on an initial set of seed words. The seed words are then used for lexicon expansion, to increase the size of the sentiment lexicon. We have explored various methods for lexicon expansion from the set of seed words, such as extracting their neighbours from a word embedding model, or extracting their synonyms from a lexical resource. We will discuss both manual and automatic methods for lexicon creation.

4.1 Evaluation of sentiment lexicons

There is not a standard method for evaluating sentiment lexicons, but the goal of the evaluation is often to measure how well they agree with human judgement. Evaluation of sentiment lexicons can be performed in various ways, for example using task-based methods such as measuring the accuracy on a document classification task using sentiment lexicons, or using the lexicon as a part of a larger evaluation system for example in a neural network. In this work, we are using an evaluation method that is similar to the unsupervised evaluation method by Wang and Xia (2017), described in Section 2.3.4 in Chapter 2. In this evaluation method, the classifier predicts the label of a review based on the number of words in the review that are present in the sentiment lexicon. The review is positive if there are more positive words, from the sentiment lexicon present in the review, than negative words, and vice versa.

$$Polarity_{review} = \sum_{P_i \in Positive} - \sum_{P_i \in Negative} \tag{4.1}$$

In our evaluation method, if there is an equal amount of positive and negative words present in the text, the classifier will assign a label depending on the id of the review in the dataset. Reviews with an even-numbered id will be assigned a positive label, whereas reviews with an odd-numbered id will be assigned a negative label. The goal is to assign labels in a seemingly random manner when the classifier is not able to make a prediction. This method for assigning labels based on the id of the review will hopefully ensure a fair distribution of positive and negative labels in the cases when the classifier is not able to assign a label. The reviews used for evaluation of the sentiment lexicons in this work are the lemmatized versions of the development sets of NoReC$^{\pm}$ and the Pros and Cons, which were described in Chapter 3. There are many possible variations of the evaluation method we are using, such as taking negation into account using a sentiment shifter function as reported by Hammer et al. (2015) and Hammer, Bai, et al. (2014), or using different weighting schemes for the words in the reviews and the sentiment lexicons as reported by Nielsen (2011). In this work, we will only use the evaluation method in its simplest form, which is a very primitive approach to sentiment analysis. The classification algorithm will for example not take negation in a review into account when predicting the label of the review. Example 4.2 presents a sentence from the Pros and Cons dataset, which demonstrates the problem of negation.

(4.2) "Ikke den beste innsynsvinkelen"
Not the best viewing angle

The only sentiment-bearing word in Example 4.2 is the word "beste" *(best)*. Since there is only one word in the text that would be present in a sentiment lexicon, it would be assigned a positive label. However, the word "beste" *(best)* is negated by the word "ikke" *(not)*, which in this case shifts the sentiment of the text from positive to negative. Taking negation into account in the evaluation method would require a more sophisticated algorithm for classification because of the complexity of negation. The two datasets that we will use in the evaluation of sentiment lexicons in this work are very different, and may also differ in terms of negation. An evaluation method that accounts for negation in the Pros and Cons dataset may not be able to detect negation in NoReC. In this work, we want to focus on the creation of sentiment lexicons and explore various approaches for this task, so developing a variety of evaluation methods that account for negation is outside the scope of this thesis.

The goal of this work is to explore a variety of automatic methods for sentiment lexicon creation, and use our simple evaluation approach to compare the lexicons. We have chosen this method for evaluation, because we want to examine the isolated effects of the sentiment lexicons. The goal is that this simple form of evaluation primarily uses information from the sentiment lexicons, and does not rely on cues from review categories or other possible features that can be learned by a more sophisticated classifier. We hope that the evaluation method that we have chosen can give us a good indication of the coverage of the lexicon, by identifying the number of hits each lexicon

has generated in the evaluation datasets. Later in this chapter, we will look at the correlation between the number of hits in the evaluation dataset and the accuracy.

In the following sections, we will explore different methods for the creation of sentiment lexicons. In Section 4.2, we will present a method for automatically extracting sentiment lexicons from labelled text. This approach is based on the assumption that each rating is associated with a distinct vocabulary (Pang and Lee, 2008), so positive words have a much higher probability of appearing in positive reviews than negative, and vice versa. We have used both NoReC and the Pros and Cons dataset for this task, since these datasets consist of reviews that are associated with a rating. In Section 4.3, we will explore another approach for creating sentiment lexicons, which is based on a set of seed words. We will then use the seed words to expand the sentiment lexicons, using resources containing lexical relations, and word embedding models. The evaluation method that we have have described in this section will be used to evaluate and compare the sentiment lexicons.

4.2 Extracting sentiment lexicons from labelled text

The first approach for sentiment lexicon creation that we will explore in this work, is the extraction of sentiment lexicons from labelled text. Using the methods by Turney (2002) or Potts (2011) as described in Section 2.3.3 in Chapter 2, sentiment lexicons can be created from labelled text, by extracting words from positive and negative reviews. We have decided to use the method by Potts (2011) for this task, because of the similarities between the dataset he uses (IMDB) and our own dataset (NoReC), and because his method does not involve identifying a set of seed words, which will be the focus of sentiment lexicon creation later in this chapter. The main focus of the paper by Potts (2011) is on negation and how it affects sentiment. He creates probability distributions of Potts scores for the words in the dataset, to show the distribution of ratings for each word. Since he identifies words that are very frequent in either positive or negative reviews, and very infrequent at the opposite end of the distribution, we can use his method, and adapt it to fit our task of sentiment lexicon creation.

Since the method by Potts (2011) only creates probability distributions of Potts scores for words, and does not involve sentiment lexicons, we had to adjust the method to fit our task and datasets. We have used the equations presented in Equation 2.1 and 2.2 in Chapter 2, to obtain a probability distribution for each word over the classes of ratings in our datasets. After obtaining the probability distributions for each word, we had to decide how to use these distributions to create sentiment lexicons. We decided to add words with high Potts scores for either positive or negative ratings to the lexicon. We assume that neutral words are likely to occur in both positive and negative reviews, so they will have roughly the same probability of belonging to the positive and negative classes, and therefore are not added to the lexicon. To create lexicons from the Potts

distributions, we had to decide on different threshold values for the words that are added to the lexicons. These threshold values, along with other decisions we had to make, will be described in the following sections. We will first present our approach for extracting words from the Pros and Cons dataset, and analyse the resulting lexicon in Section 4.2.1. In Section 4.2.2, we will describe the methods used to create sentiment lexicons from NoReC. Table 4.1 presents the evaluation results on the NoReC$^{\pm}$ and the Pros and Cons development sets by the resulting lexicons.

Dataset used for lexicon creation	Potts score threshold	Frequency threshold	Lexicon size	Accuracy NoReC$^{\pm}$	Accuracy Pros and Cons
Pros and Cons	0.9	5	294	53.07	87.52
NoReC$^{\pm}$	0.9	5	1,063	77.85	55.85
	0.8	55	252	64.03	55.47
Both	0.9	5	1,357	59.21	85.98

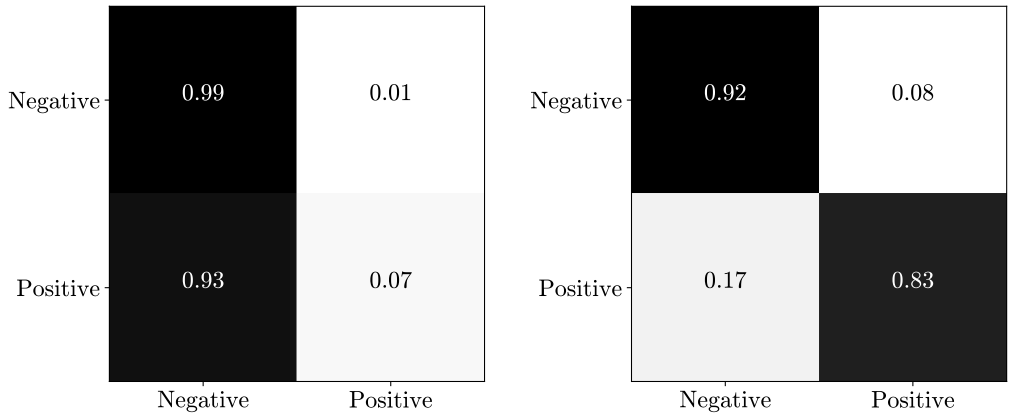
Table 4.1: Evaluation results of the Potts lexicons created from the training sets of NoReC and the Pros and Cons. The Potts score represents the Potts score threshold for including words to the lexicon. The lexicon created from both datasets is the concatenation of the Potts lexicon from the Pros and Cons dataset and the Potts lexicon from NoReC$^{\pm}$, both using a frequency threshold of 5, and a Potts score threshold of 0.9.

4.2.1 Creating a Potts lexicon from the Pros and Cons dataset

The probability distributions for the words in the Pros and Cons dataset only consist of two values, since there are only two classes (i.e. Pros and Cons). We only considered words that appear five times or more in the dataset, to avoid adding very infrequent words to the sentiment lexicon. Higher frequency thresholds were tested, but very few words were added to the lexicon when the threshold was above five for this dataset. To create the sentiment lexicon, we extracted the adjectives, adverbs, nouns, and verbs from the training set of the Pros and Cons. These words were used to make probability distributions of Potts scores. Different probability values for adding words to the lexicon were also tested, and a probability threshold of 0.9 seemed appropriate based on a manual inspection of the resulting lexicons. When setting the threshold higher than 0.9, very few words were added to the lexicon. Lower values for the Potts score threshold resulted in very large lexicons that mainly consisted of non-sentiment-bearing words. The resulting lexicon, created from the Pros and Cons dataset with a frequency threshold of five, and a minimum Potts score of 0.9 for each word, consists of 137 positive and 157 negative words.

The lexicon has quite good performance on the Pros and Cons development set, however, it performs poorly on the NoReC$^{\pm}$ development set. It achieves an accuracy

of 53.07 on the evaluation set of NoReC$^{\pm}$, and an accuracy of 87.52 on the Pros and Cons evaluation dataset. Figure 4.1 presents the accuracies across labels obtained by the lexicon, on the two evaluation datasets. The lexicon created from the Pros and Cons dataset has an accuracy of 92.00 when predicting negative reviews, and an accuracy of 83.00 when predicting positive reviews in the Pros and Cons development dataset.



(a) The NoReC$^{\pm}$ development set.

(b) The Pros and Cons development set.

Figure 4.1: Confusion matrices on the evaluation datasets by the Pros and Cons Potts lexicon.

The majority of the predictions for the reviews in the development split of NoReC$^{\pm}$ are negative, even though there is an even distribution of positive and negative reviews in the dataset. The lexicon has 1,552 positive and 6,940 negative hits in the evaluation dataset, which is why it almost always makes a negative prediction. We have examined the top negative hits, and found the words with the highest frequencies in the dataset, which are "for" *(for)*, with a frequency of 2,207, and "ikke" *(not)*, with a frequency of 1,644. Both of these words obtain more hits individually, than the entire set of positive words in the lexicon. Among the top negative hits for this lexicon, we also find words such as "litt" *(little)*, "kun" *(only)*, "igjen" *(again)*, "finne" *(find)*, and "synes" *(think)*. None of these words would be labelled positive or negative by a human annotator, because they are not sentiment-bearing.

Since these words have been added to the Potts lexicon, they mostly occur in negative summaries in the Pros and Cons dataset, which means that they may be indicative of negative sentiment in this dataset. We have examined some of the negative summaries in the development set of the Pros and Cons, and found many cases where the author uses a positive word together with a negation, instead of just using the negative antonym, for example describing a product as "not good" instead of using the word "bad". Two other interesting negative words in the lexicon are the words "finne"

(find) and "synes" *(think)*. These words indicate that negative summaries contain more subjective opinions than positive summaries, since the Potts score for the positive class for these words is less than 0.1. Subjective opinions can make the reviews seem less strict, and neutralize the negativity.

The lexicon achieved good results on the Pros and Cons development set, but wrongly labelled almost all of the positive reviews in NoReC$^{\pm}$ dev because of functional words and non-sentiment-bearing words in the lexicon. We have removed stopwords from the lexicon, to examine how it affects the accuracies on the evaluation datasets. For this task, we have used the NLTK list of stopwords, available from Github.[1] The NLTK list of stopwords is a list of 176 words in Bokmål and Nynorsk, which are mostly prepositions, pronouns, and modal verbs. After removing the stopwords from the lexicon, the accuracy on the development set of NoReC$^{\pm}$ increased from 53.07 to 60.52, whereas the accuracy on the Pros and Cons development set decreased from 87.52 to 81.76. It seems that the stopwords are beneficial for labelling summaries in the Pros and Cons dataset, whereas removing these words from the lexicon increases the accuracies on the NoReC$^{\pm}$ evaluation dataset.

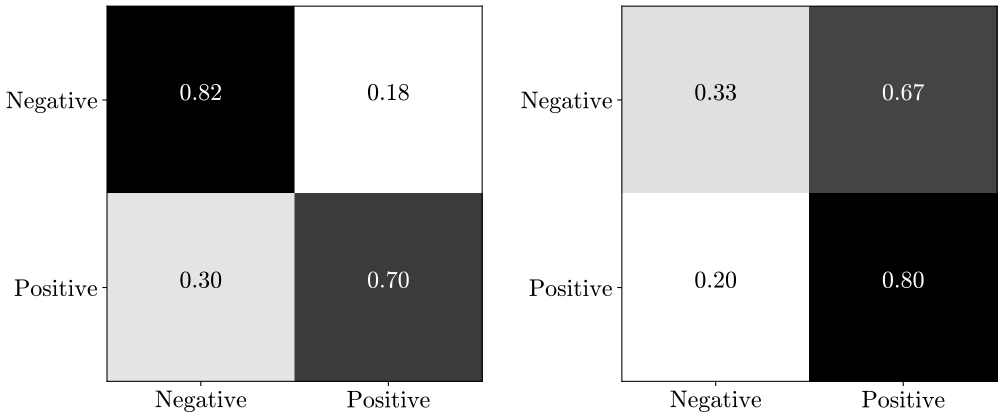
4.2.2 Creating Potts lexicons from NoReC

We have also explored two different methods for extracting adjectives, adverbs, nouns, and verbs from the training set of NoReC$^{\pm}$. First, we used the same approach as we used for the Pros and Cons dataset to extract positive and negative words with a frequency of five or more, from the training set of NoReC$^{\pm}$. We created distributions of Potts scores for each of these words, and added words with a Potts score of 0.9 or higher to the sentiment lexicon. The resulting lexicon consists of 1,284 positive and 975 negative words. The lexicon achieved an accuracy of 77.85 on the NoReC$^{\pm}$ development set, and an accuracy of 55.85 on the Pros and Cons development set. Figure 4.2 presents the accuracies across labels obtained by this NoReC$^{\pm}$ Potts lexicon.

We have examined the lexicon to find out why the majority of the predictions on the Pros and Cons dataset are positive, even though there is an even distribution of positive and negative summaries. Many of the negative words in the lexicon can be used in movie descriptions, such as the words "film" *(movie)*, "dansefilm" *(dance movie)*, and "actionkomedie" *(action comedy)*. Among the positive words, there are words such as "cpu" *(cpu)*, "audio" *(audio)*, "glidelås" *(zipper)*, "høyttalere" *(speakers)*, and "maskinvare" *(hardware)*, which can be used in product descriptions. Figure 3.2 in Chapter 3 presents the category distribution of NoReC, which is the corpus that was used to create this lexicon. The majority of the movie reviews in NoReC are negative, whereas the majority of product reviews are positive, which can explain why these words were added to the positive and negative side of the sentiment lexicon. The Pros and Cons dataset is a collection of the most sentiment-bearing sentences associated

[1] https://github.com/xiamx/node-nltk-stopwords/blob/master/data/stopwords/norwegian

with product reviews from NoReC. Therefore, it is not surprising that the words from the positive sentiment lexicon, which can be used in product descriptions, obtain many hits in the Pros and Cons development set, and are responsible for many of the incorrect positive predictions.



(a) The NoReC$^{\pm}$ development set.

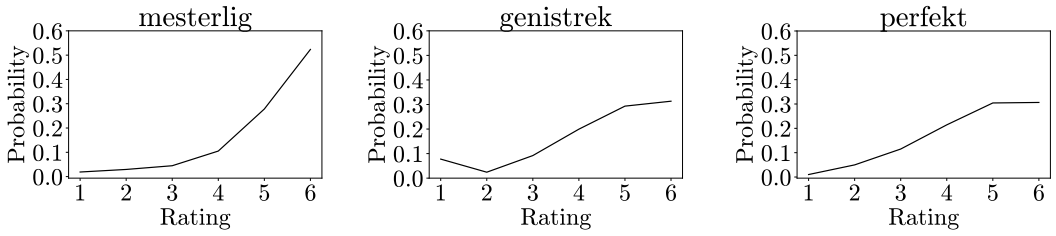
(b) The Pros and Cons development set.

Figure 4.2: Confusion matrices of class accuracies of the NoReC$^{\pm}$ lexicon created from Potts scores.

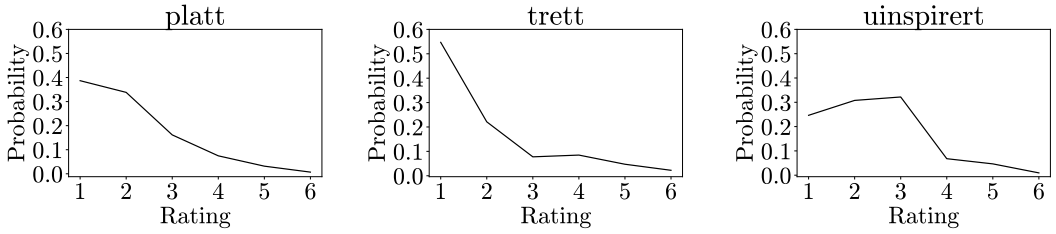
We removed the stopwords from the lexicon and achieved the exact same accuracies as before removing the stopwords. None of the words on the NLTK list of stopwords were present in this lexicon. This indicates that the stopwords are not frequent in reviews with a specific polarity, neither are they infrequent in reviews with the opposite polarity in the NoReC dataset. It can explain why the accuracy on the NoReC$^{\pm}$ development set increased after removing stopwords from the Pros and Cons Potts lexicon in the previous section.

The second approach for creating a Potts lexicon from NoReC$^{\pm}$ uses information from all of the reviews in the training set of NoReC, but only includes words that are frequent in NoReC$^{\pm}$. Very few words have Potts scores of more than 0.2 for a single rating, since the probability distributions are over all the ratings in the dataset (1–6). Figure 4.3 presents the distribution of Potts scores for some of the words in NoReC$^{\pm}$ with the highest Potts scores for a single rating. These distributions are similar to the distributions of the positive word "awesome" and the negative word "terrible", presented in Figure 2.1 in Chapter 2.

The second NoReC$^{\pm}$ Potts lexicon was created from the distributions of Potts scores for all the adjectives, adverbs, nouns, and verbs in the training set of NoReC. We



(a) "mesterlig" *(masterful)* (b) "genistrek" *(stroke of genious)* (c) "perfekt" *(perfect)*



(d) "platt" *(flat)* (e) "trett" *(tired)* (f) "uinspirert" *(uninspired)*

Figure 4.3: Visualization of Potts distributions of words from the NoReC$^{\pm}$ reviews.

experimented with different values for the frequency threshold, and the Potts score cut-off, to find a configuration that was suitable for our task and dataset. We first divided the reviews into a negative class (ratings 1, 2, and 3) and a positive class (ratings 4, 5, and 6). We found that using a cut-off probability score of 0.8 for either the positive or negative class, ensured that words that were added to the lexicon had the majority of their distribution in the positive or negative class. To make sure that the words that were added to the lexicon were actually present in NoReC$^{\pm}$, we set the Potts score threshold for an individual class to 0.2. Since the negative words in the lexicon are from reviews with a rating of 1 or 2, their Potts score threshold was set to 0.4. The Potts score threshold was set to 0.2 for positive words since these words were extracted from reviews with a single rating. A frequency threshold of five, which was used to create the previous lexicons, resulted in a lexicon of 13,598 negative and 818 positive words. A manual inspection of the lexicon showed an extreme amount of non-sentiment-bearing words, and the difference in size between the positive and negative lists was extremely large. We experimented with different values for the frequency threshold, and found that a threshold of 55 ensured two classes of roughly the same size.

The resulting lexicon consists of 140 positive and 112 negative words. It achieves an accuracy of 64.03 on the NoReC$^{\pm}$ development set and an accuracy of 55.47 on the Pros and Cons dataset. This lexicon achieves almost the same results as the first NoReC$^{\pm}$ Potts lexicon on the Pros and Cons dataset, however, it has much worse performance on the NoReC$^{\pm}$ dataset. We removed the stopwords from this lexicon, and found that the accuracy on NoReC$^{\pm}$ increased to 64.69 and the accuracy on the Pros and Cons

evaluation dataset decreased to 54.89. Similar to the first Potts lexicon from NoReC$^{\pm}$, very few stopwords were present in this sentiment lexicon as well.

4.2.3 Combining the Potts lexicons

We have created a sentiment lexicon by combining the information in the Potts lexicons extracted from NoReC$^{\pm}$ and the Pros and Cons. Only the NoReC$^{\pm}$ lexicon with the best performance on the evaluation datasets was used for this task. The new lexicon is a combination of the 1,063 words from the NoReC$^{\pm}$ lexicon, and the 294 words from the Pros and Cons lexicon. This lexicon achieves a slightly lower accuracy on the Pros and Cons development set than the Potts lexicon created from just the Pros and Cons training set. However, the accuracy on the development set of NoReC$^{\pm}$ is much lower than the accuracy achieved by the Potts NoReC$^{\pm}$ lexicon. There are more than three times as many words from the NoReC$^{\pm}$ lexicon than the Pros and Cons lexicon in the concatenated lexicon, so it is surprising that the accuracy on the development set of NoReC$^{\pm}$ was significantly lower for the combined Potts lexicon than for the Potts NoReC$^{\pm}$ lexicon.

After removing stopwords from the combined lexicon, the accuracy on the development set of the Pros and Cons decreased from 85.98 to 79.84, whereas for the development set of NoReC$^{\pm}$, the accuracy increased from 59.21 to 73.90. This lexicon has the best overall performance among the lexicons extracted from NoReC$^{\pm}$ and the Pros and Cons training sets. However, we do not know it the lexicon has been overfit to both of our datasets, or if it would produce similar results on a different unseen dataset. Ideally, we should have had a third evaluation dataset to test the lexicons extracted from NoReC$^{\pm}$ and the Pros and Cons training set, to find out how well they perform on unseen data. Unfortunately, these two datasets are the only existing datasets for sentiment analysis in Norwegian.

The lexicons created from NoReC$^{\pm}$ and the Pros and Cons, have good performance when they are evaluated on the same dataset that was used for creation of the lexicons, and quite poor results when evaluated on the other dataset. It seems that they are not able to generalize to different kinds of reviews. The two datasets are very different, especially in terms of average length, and the category distribution. The reviews in the training set of NoReC$^{\pm}$ has an average length of 417 words, whereas the summaries in the Pros and Cons training set has an average length of 14 words. The summaries in the Pros and Cons dataset are from product reviews, whereas the category distribution in NoReC is more diverse. The differences in the datasets can explain why the lexicon created from one of the datasets does not perform well on the other dataset.

In the following section, we will explore an approach for sentiment lexicon creation based on a set of seed words. We will first explore two different methods for identifying the sets of seed words. We will then use the seed words to expand the size of the

sentiment lexicons, using various resources, such as word embedding models and resources containing lexical relations.

4.3 Lexicon creation based on a set of seed words

The second approach for creating lexicons in this work, is sentiment lexicon creation based on an initial set of seed words. We have created different sets of seed words, using two separate approaches. The first approach for creating the seed words was manually labelling a set of selected words positive or negative. The second set of seed words was created using an automatic approach, by extracting words from the lexical resource WordNet. Since WordNet contains words that are labelled positive or negative, it should be suitable for the task of creating a set of seed words. Later in this chapter, the sets of seed words will be used for lexicon expansion.

4.3.1 The manually annotated seed words

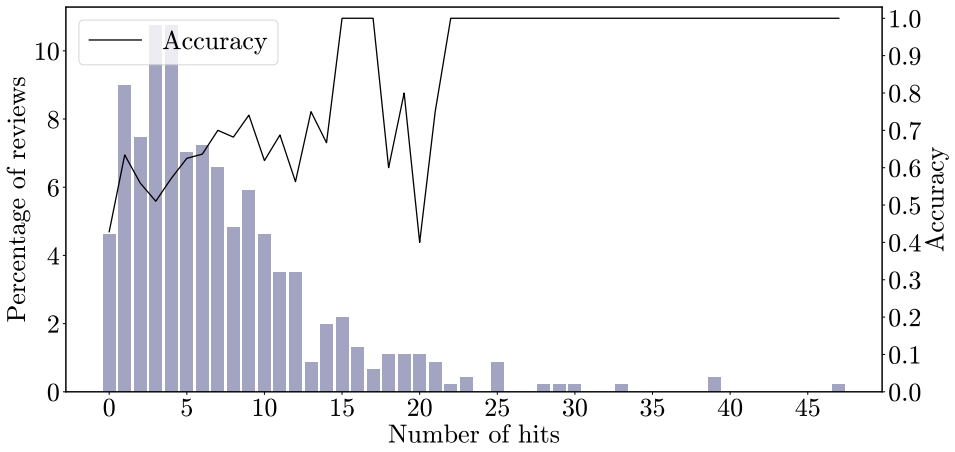
The first set of seed words is a manually annotated list of 47 positive and 50 negative adjectives and adverbs, which was created without consulting any lexical resources. The manually annotated seed words are presented in Table 4.2, whereas the English translation is presented in Table A.1 in Appendix A.

Positive seed words	Negative seed words
fin, bra, hyggelig, pen, trygg, god, nydelig, genial, morsom, imponerende, ærlig, herlig, vakker, fantastisk, positiv, utmerket, enestående, sterk, flink, ekte, koselig, kul, gøy, enkel, kjærlig, elskverdig, varm, gledelig, intelligent, glimrende, suksessfull, høflig, ren, heldig, søt, vennlig, gavmild, perfekt, super, strålende, flott, lys, lekker, elegant, praktisk, moderne, solid	dårlig, stygg, grusom, slem, fæl, makaber, negativ, skitten, sint, syk, voldelig, sur, feil, tåpelig, klumpete, treg, falsk, rar, streng, gal, vanskelig, kjip, redd, sørgelig, smertefull, teit, ekkel, dum, misunnelig, svak, ille, vond, mørk, kjedelig, sær, sliten, farlig, doven, merkelig, tung, lat, gammel, kaotisk, hard, elendig, slapp, farlig, vulgær, dyr, kald

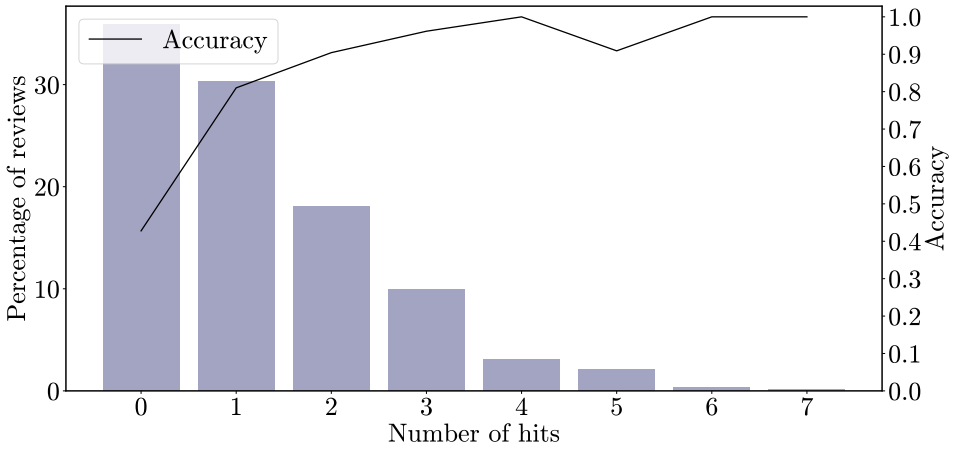
Table 4.2: The manually annotated seed words.

The goal was to ideally create a list of words that had an unambiguous polarity sentiment label across all contexts. However, after the creation of the list, we have realized that not all of the words in the list of seed words are unambiguous across all contexts. The word "enkel" *(easy/simple)* can be used in a positive context for something that is easy, for example "an easy installation", but it can also be used in a negative context, for example to describe a design that is too simple.

The seed words will be used for sentiment lexicon expansion using various resources, such as word embedding models and resources containing lexical relations. Since we are going to create new lexicons based on the set of seed words, we want to measure the coverage of the seed words on the evaluation datasets. All of the lexicons that are created from seed words also include all of these words, so we can expect the same or better coverage from the lexicons that are generated through lexicon expansion. The coverage of the manually annotated list of seed words on the development splits of NoReC$^{\pm}$ and the Pros and Cons is presented in Figure 4.4.



(a) The development set of NoReC$^{\pm}$.



(b) The development set of the Pros and Cons.

Figure 4.4: Coverage and accuracy of the manually annotated seed words.

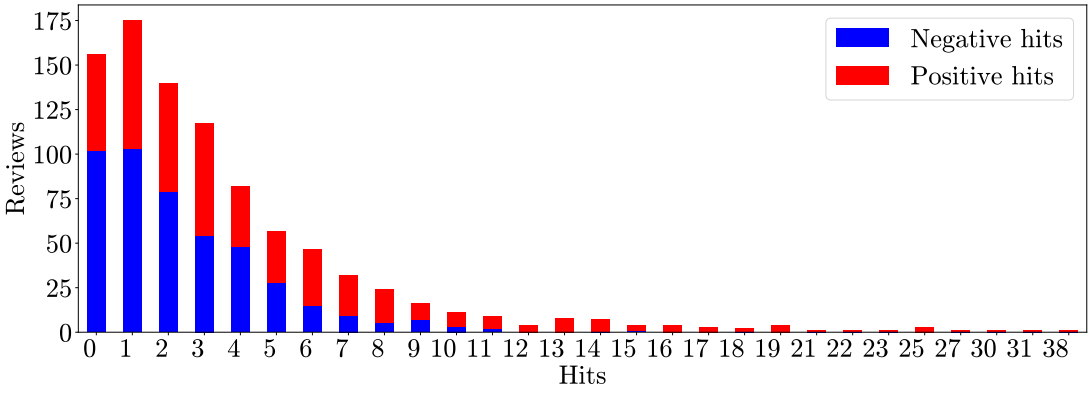
Around 95% of the reviews in the development split of NoReC$^{\pm}$ have one or more words present from the set of manually annotated seed words, and more than half of the reviews have five words or more. The set of seed words has fewer hits in the

Pros and Cons dataset, around 65% of the summaries have one or more words present from the set of seed words. The reviews in the development split of NoReC$^{\pm}$ are long reviews, with an average length of 518 words per review, whereas the summaries in the Pros and Cons dataset are much shorter, with an average length of 18 words. The set of seed words has much better coverage on the development set of NoReC$^{\pm}$ than on the Pros and Cons development set, but since the average lengths are so different, a hit in a summary in the Pros and Cons dataset may be more relevant for predicting the sentiment, than a hit in a review in NoReC$^{\pm}$.

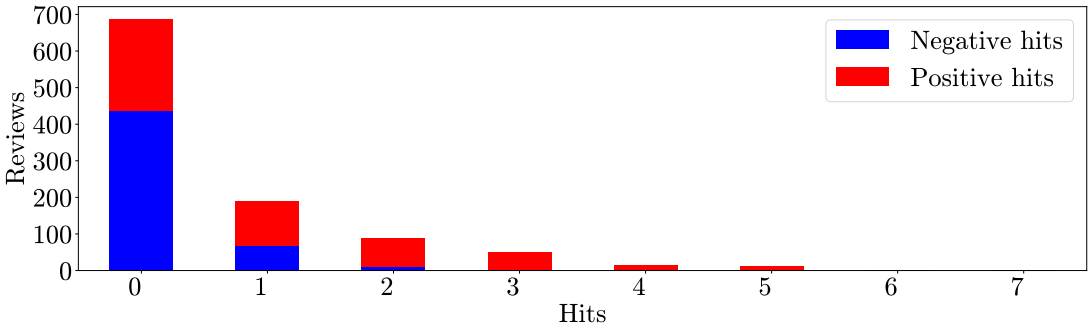
The accuracy of the classifier generally increases as the number of words present in the evaluation text from the lexicon increases. For the development set of NoReC$^{\pm}$, there is a large decrease in accuracy for reviews with 20 words present from the lexicon. However, only 1% of the reviews, which is a total of five reviews in the dataset, had 20 hits from the lexicon, and three of these reviews were incorrectly labelled. All of the reviews were labelled positive by the classifier, however, three of the reviews were negative. The three reviews that were wrongly labelled, consisted of many negated words from the positive list of seed words such as "ikke spennende" *(not interesting)*, and "ikke morsom" *(not funny)*, which can explain why the reviews were wrongly labelled positive. All of the reviews with 22 or more hits in the development set of NoReC$^{\pm}$ are correctly labelled by the classifier. Later in this chapter, we will expand the size of the sentiment lexicons using various lexical resources. The goal of the expansion is to increase the vocabulary in the lexicon, which hopefully leads to more hits in the evaluation dataset and better accuracies.

We have also examined the difference between the positive and negative hits in the evaluation datasets, which is presented in Figure 4.5. Since a review can have both positive and negative hits, these charts represent frequency counts and not percentages of reviews. For each review in the dataset, we count the number of positive and negative hits, and report both of these numbers individually. It means that a review with zero negative hits and one positive hit, will contribute to the negative bar representing zero hits and the positive bar representing one hit. We wanted to represent the positive and negative hits in the same chart, to allow for a better comparison between the positive and negative hit frequencies.

The majority of negative hits in the development split of NoReC$^{\pm}$ are in reviews with just a few hits. All of the reviews with 13 or more hits, only have hits from the positive seed words. The manually annotated set of seed words has 2,188 positive and 1,078 negative hits in the development split of NoReC$^{\pm}$, which is over twice as many positive than negative hits. The distribution of hits in the Pros and Cons dataset is similar, with a majority of positive hits. The seed words have 540 positive and 100 negative hits in the Pros and Cons development set.



(a) The development set of NoReC$^{\pm}$.



(b) The development set of the Pros and Cons.

Figure 4.5: The distribution of positive and negative hits from the manually annotated set of seed words.

The manually annotated list of seed words has good coverage on both datasets, with hits in over 95% of the reviews in the development set of NoReC$^{\pm}$, and in over 65% of the summaries in the Pros and Cons development set. The set of seed words only contains adjectives and adverbs, and it seems that many of these words are present in both positive and negative texts, using negation to shift the sentiment. By only including adjectives and adverbs in the list of seed words, lexicon expansion may be restricted to primarily adding words from the same word classes. To obtain a very different set of seed words, we have extracted the labelled words from WordNet, which are primarily nouns, and some verbs, adjectives, and adverbs. We wanted to see how the list containing adjectives and adverbs differed from the list containing mostly nouns, on the evaluation task of this work. Another motivation for creating a second set of seed words was to explore both a manual method and an automatic method for seed word creation. In the following paragraphs, we will describe the set of seed words extracted from WordNet.

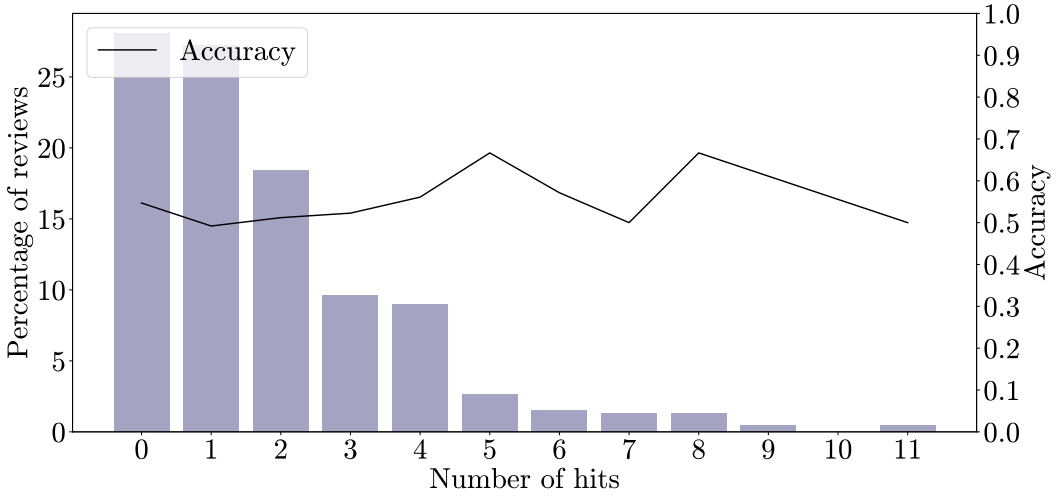
4.3.2 The WordNet seed words

We created a different set of seed words by extracting all of the synsets with a positive or negative label from the Norwegian WordNet. For this task, we used the latest version of WordNet, modified by Sand et al. (2017). The WordNet is a collection of individual files, and each file represents a different lexical relation between the synsets. The version by Sand et al. (2017) is a modification of some of the original WordNet files, and the file containing the connotations is not among the modified files. We therefore had to extract the connotations from the original version of WordNet, and compare the connotated synsets to the synsets modified by Sand et al. (2017) to ensure that only the connotated synsets present in the new version were added to the list of seed words. To match the word format in our datasets, we separated each synset into individual words. The set of seed words extracted from WordNet consists of 131 positive words and 330 negative words.

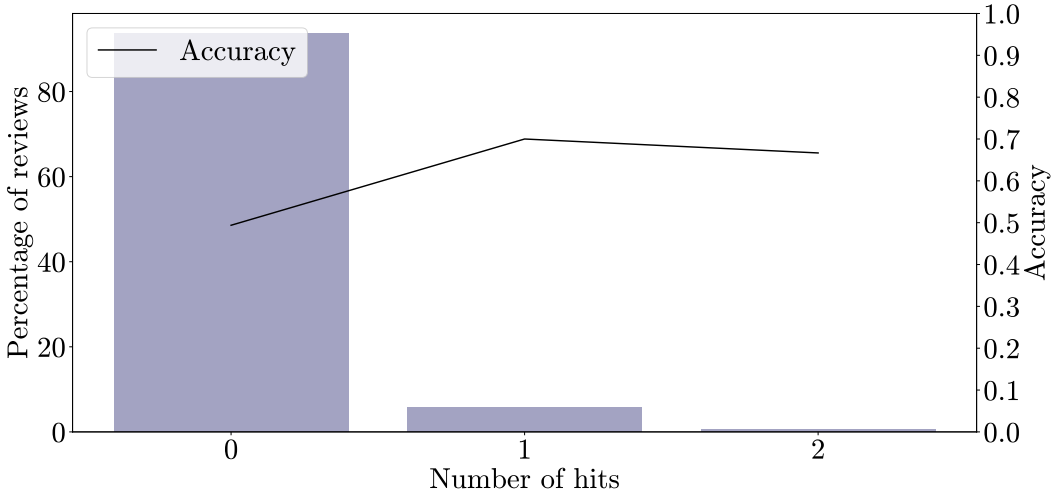
We have also examined the coverage of this set of seed words on the evaluation datasets, which is presented in Figure 4.6. Over 25% of the reviews in the development set of NoReC$^{\pm}$, and around 90% of the summaries in the development set of the Pros and Cons did not have any hits from the WordNet seed words. One possible explanation for the low coverage, is that the synsets and their connotations have been translated from Danish to Norwegian. Even though the two languages are similar, positive and negative words in the Danish language may not necessarily be indicative of sentiment in Norwegian.

Another possible explanation for why the coverage is low, is that the WordNet seed words are not present, or obtain very few hits in the evaluation datasets. Among the positive WordNet seed words, we find words such as "hanndyr" *(male animal)*, which is not present in NoReC$^{\pm}$ dev, and "due" *(pigeon)*, and "hai" *(shark)* which both have one hit each in the dataset. The positive WordNet seed word with the highest frequency in NoReC$^{\pm}$ dev is the non-sentiment-bearing word "ønske" *(wish)*, with a frequency of 81. Among the negative WordNet seed words, there are many sentiment-bearing words such as "hykler" *(hypocrite)*, "gærning" *(lunatic)*, "avskyelig" *(disgusting)*, "fjolle" *(silly)*, and "feiging" *(coward)*, but they each only have one hit in the NoReC$^{\pm}$ development set, whereas the word with the most hits is the non-sentiment-bearing word "dukke" *(doll)*, with a frequency of 65. The WordNet seed words obtain 36 hits in the Pros and Cons development set (16 positive and 20 negative), and 820 hits in NoReC$^{\pm}$ dev (485 positive and 335 negative).

Figure 4.6 shows that there is not a clear relationship between the coverage and the accuracy for this lexicon, the accuracy is within the range of 40.0–60.0 for all the numbers of hits. Predicting whether a review is positive or negative is a binary classification problem, if the two classes are roughly the same size, we can expect the classifier to achieve an accuracy of roughly 50.0 just by guessing the label. When



(a) The development set of NoReC$^{\pm}$.



(b) The development set of the Pros and Cons.

Figure 4.6: Coverage of the WordNet seed words.

there are no hits in the review, the classification algorithm randomly assigns a label depending on the id of the review. There are many reviews that do not have any hits from the list of WordNet seed words, especially in the Pros and Cons dataset, so this could be an explanation for why the classifier obtains almost the same results as if the labels were randomly assigned. In the following section, we will use the evaluation task described in Section 4.1 for a comparison between the seed words from WordNet and the manually annotated ones.

4.3.3 Seed word evaluation

We are using the simple approach for sentiment lexicon evaluation described in Section 4.1 for evaluating the two sets of seed words. The evaluation results are presented in Table 4.3. As we already saw in the previous section, the accuracies obtained by the WordNet seed words are slightly above 50.0 for both datasets, which are results that the classifier could have obtained by randomly predicting a label. The manually annotated seed words performed better than chance on both datasets. It had better performance on the Pros and Cons than on NoReC$^{\pm}$ dev, which is not very surprising considering the differences in the datasets. NoReC is a dataset containing full reviews, whereas the Pros and Cons dataset contains short summaries with the most polarized sentiment-bearing sentences. Since the manually annotated set of seed words obtained the best results on both datasets, it will be used as the main set of seed words used for lexicon expansion. In the following sections, we will explore different methods for increasing the size of sentiment lexicons.

Lexicon description	Lexicon size	Accuracy NoReC$^{\pm}$	Accuracy Pros and Cons
Seed words adj/adv	100	**63.81**	**71.45**
Seed words WordNet	461	53.50	50.76

Table 4.3: A comparison of the WordNet seed words and the manually annotated seed words containing only adjectives and adverbs.

4.4 Increasing the size of the sentiment lexicon

We saw in the Section 4.3.1, that for the best performing set of seed words, the evaluation accuracy increased as the number of words from the lexicon present in the text increased. In this section, we are going to use the seed words to expand the size of the sentiment lexicons. We expect the larger lexicons to have better coverage on the evaluation datasets, and therefore also achieve higher accuracies on the evaluation task. We will present different methods for lexicon expansion and discuss the resulting lexicons. The main approach that we have explored is lexicon expansion from word embedding models. In this project, we aim to use automatic approaches for creating sentiment lexicons with large vocabularies, which have high precision on the evaluation datasets. Using automatic methods for lexicon expansion, we can create sentiment lexicons with a large vocabulary in a relatively small amount of time. In the following paragraphs, we will describe some of the automatic approaches for lexicon expansion we have explored in this work.

4.4.1 Sentiment lexicon expansion using word embedding models

Word embedding models can be used to expand sentiment lexicons, because they contain information about word similarity. The models can be used to find the n-nearest neighbours to each seed word, and add these words to the sentiment lexicon together with the label of the seed word. Each word in the word embedding model is represented by a vector, which can be used to calculate the cosine similarity between the seed words and other words in the model. The cosine similarity between two vectors is defined in Equation 4.3. We can experiment with the number of neighbours extracted from the model, and the number of iterations for extracting neighbours. For example, if we chose to extract three neighbours in two iterations, the top three neighbours of the seed words are extracted in the first iteration, and in the second iteration, the top three neighbours of the previously extracted words are added to the sentiment lexicon.

$$\cos(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{AB}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} \mathbf{A}_i \mathbf{B}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{A}_i)^2}\sqrt{\sum_{i=1}^{n} (\mathbf{B}_i)^2}} \tag{4.3}$$

Word embedding models for sentiment lexicon expansion

The word embedding models used for this task were created by Fares, Kutuzov, Oepen, and Velldal (2017), and can be downloaded from the NLPL repository.[2] All of these models were trained on "Aviskorpuset" *(The Norwegian Newspaper Corpus)*, which is available from Språkbanken.[3] The corpus used to train the word embedding models is a collection of news articles with more than 1 billion words in Bokmål and 60 million words in Nynorsk, as reported by Stadsnes (2018). Only models containing lemmatized words were used, since we are using the lemmatized versions of the evaluation datasets. All of the models provided by Fares et al. (2017), used in this work are trained using either the CBOW or the Skipgram algorithm. The Skipgram models are trained using fastText (Bojanowski et al., 2017), whereas the CBOW models are trained using Word2Vec (Mikolov et al., 2013).

Our approach for sentiment lexicon expansion is similar to the approach by Hammer, Yazidi, et al. (2014). They identified a set of seed words and calculated the PMI between the seed words and words in "Aviskorpuset" *(The Norwegian Newspaper Corpus)* to expand their lexicon. We are going to use word embedding models trained on the same corpus to expand the sentiment lexicons in this work. Even though the sentiment lexicons by Hammer, Yazidi, et al. (2014) are available online, the identifiers in the paper do not match the identifiers in the repository, so we will not be able to perform a comparison between their top performing lexicon and our lexicons.

In word embedding models, different values for the various parameters can be selected by the creator of the model. One of the parameters is the window size,

[2] http://vectors.nlpl.eu

[3] https://www.nb.no/sprakbanken/show?serial=oai%3Anb.no%3Asbr-4&lang=nb

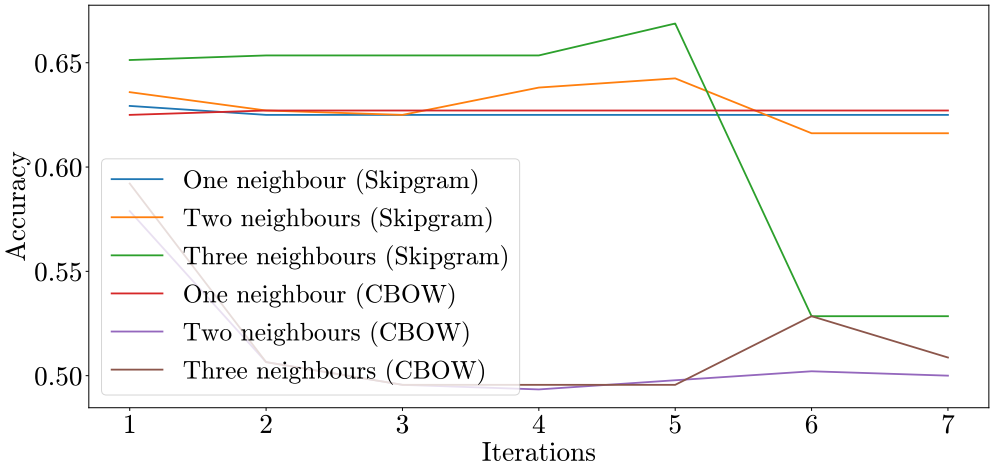
which represents the number of words to the left and right of the candidate word to consider when predicting the candidate word or the context words. All of the models we use in this project have a window size of five. Another parameter of the model is the dimensionality, which represents the length of the word vectors. We will explore various word embedding models of different dimensionalities. The word embedding model which was reported to have the best performance on the task of synonym detection using the Norwegian Synonymy Test Set by Stadsnes (2018), was the CBOW model of dimensionality 600. Models with such high dimensionalities are very time-consuming to create, and they also require long training times in a neural network and when used for sentiment lexicon expansion.

In the following paragraphs, we will explore methods that we have used for lexicon expansion using word embedding models trained on "Aviskorpuset" *(The Norwegian Newspaper Corpus)* provided by Fares et al. (2017). The manually annotated seed words containing adjectives and adverbs will be used for this task. We will explore various configurations for creating the sentiment lexicons, to find the optimal configuration for our task. We have used models with a dimensionality of 100 for these experiments, since they are less time-consuming, and therefore allow us to test a larger amount of various parameters for lexicon creation. We will then use the optimal configuration from these experiments to create new lexicons from different word embedding models of larger dimensionalities, which will be described later in this chapter.

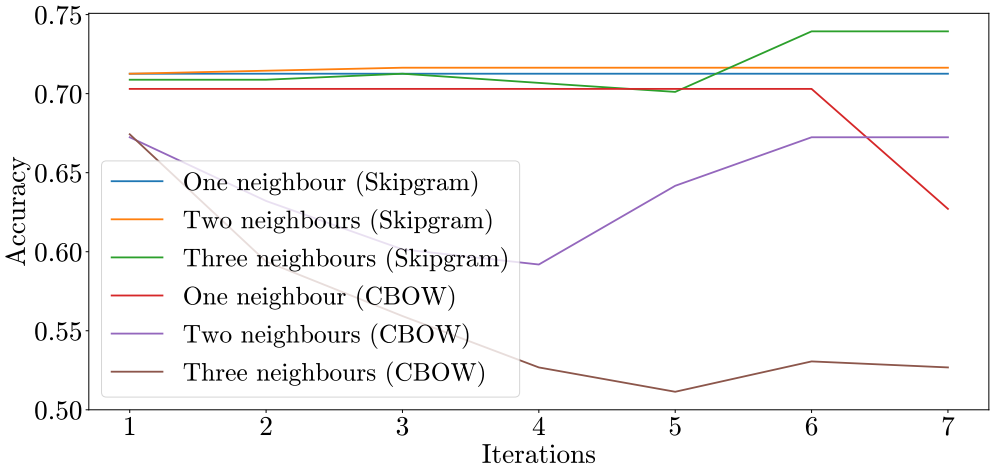
Parameter tuning for lexicon expansion

We are performing the tuning of the parameters on a Word2Vec CBOW model and a fastText Skipgram model of dimensionality 100, since the models trained using these two algorithms can be very different. We have experimented with different values for both neighbours and iterations for extracting words from the models. Lexicon expansion using only a few iterations for extracting neighbours, is the best way to ensure that all of the words in the lexicon are similar to the seed words. The probability of adding words that are not sentiment-bearing to the lexicon increases as the number of iterations increases. If one non-sentiment-bearing word is added to the lexicon, all of its neighbours will be added to the lexicon in the following iteration. If the original word is non-sentiment-bearing, it is likely that its neighbours are non-sentiment-bearing. The lexicon grows exponentially as the number of iterations increases which greatly affects the time it takes to create the lexicon.

Varying the number of iterations We have tested 1–7 iterations for lexicon expansion, and in each iteration we extracted 1–3 neighbours. The resulting sentiment lexicons were then evaluated on the development sets of NoReC$^{\pm}$ and the Pros and Cons. The results are presented in Figure 4.7.



(a) The development set of NoReC$^{\pm}$.



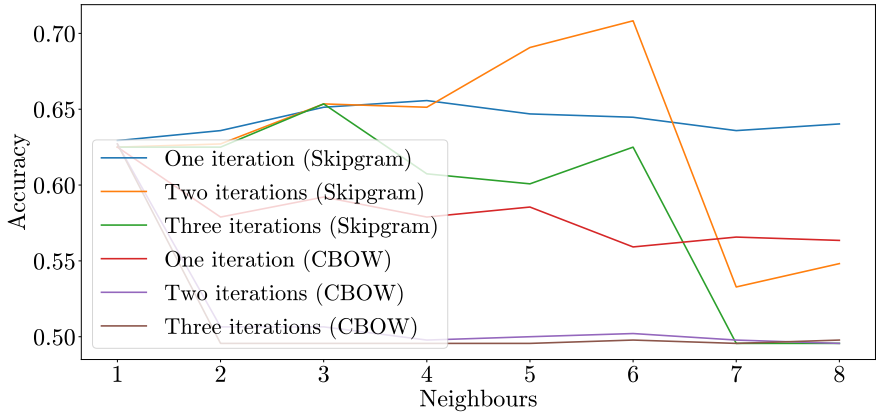
(b) The development set of the Pros and Cons.

Figure 4.7: Testing the effects of varying the number of iterations for lexicon expansion. The accuracies in the graphs represent the accuracies obtained by the resulting lexicons.

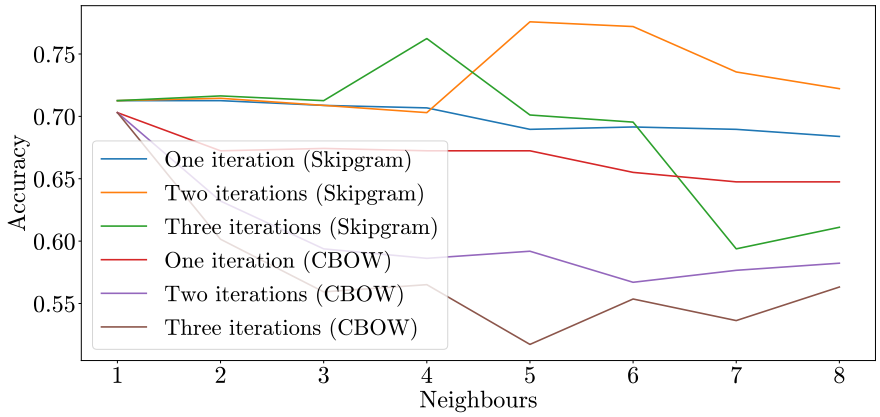
The lexicons from the fastText Skipgram model, which were created by extracting three neighbours in each iteration for 1–5 iterations have the best performance on the NoReC$^{\pm}$ dev. These are the only lexicons that achieve better results than the manually annotated set of seed words, which achieves an accuracy of 63.81 on this evaluation dataset. The lexicons created from the fastText Skipgram model have much better overall performance than the lexicons created from the Word2Vec CBOW model, on both evaluation datasets. The sentiment lexicon with the best performance on NoReC$^{\pm}$, also has the best performance on the Pros and Cons dataset. In the following

paragraphs, we will examine the effects of experimenting with different values for the number of neighbours.

Varying the number of neighbours extracted from the model We have also experimented with the number of neighbours that are extracted from the model in each iteration. Since the size of the lexicon grows exponentially larger in each iteration, we only extracted 1–8 neighbours, in 1–3 iterations from the model. The lexicons were then evaluated on the evaluation datasets. The results are presented in Figure 4.8.



(a) The development set of NoReC$^{\pm}$.



(b) The development set of the Pros and Cons.

Figure 4.8: Testing the effects of varying the number of neighbours for lexicon expansion. The accuracies in the graphs represent the accuracies achieved by the resulting sentiment lexicons.

All of the lexicons created from models trained using the Word2Vec CBOW algorithm had the same or worse performance than the set of manually annotated seed words. It

seems that the performance of all the lexicons decreases when extracting seven or more neighbours from the model. Some of the lexicons created using the Skipgram model have much better performance on the evaluation datasets than the set of seed words, especially the lexicons created in two iterations by extracting five and six neighbours in each iteration. These two lexicons achieved the highest accuracies on both of the evaluation datasets. Since the two configurations that produced the best results were very similar, we have decided to only use the configuration that achieved the best overall results for further experiments. The configuration that extracts the top six neighbours of each word in two iterations, will be the standard configuration for lexicon expansion from word embedding models. In the following section, we will use this configuration to create sentiment lexicons from word embedding models with different dimensionalities.

Testing different word embedding models for sentiment lexicon expansion

In this section, we are going to use the manually annotated seed words to create sentiment lexicons from word embedding models with dimensionalities of 100, 300, and 600. For each of the dimensionalities, we will create lexicons from a model trained using the fastText Skipgram algorithm, and from a model trained using the Word2Vec CBOW algorithm. The evaluation accuracies achieved by the resulting lexicons are presented in Table 4.4.

Lexicon description	Lexicon size	Accuracy NoReC$^{\pm}$	Accuracy Pros and Cons
Seed words (adj/adv)	100	63.81	71.45
Skipgram (100)	1,639	**70.83**	**77.20**
Skipgram (300)	1,879	62.28	71.07
Skipgram (600)	1,940	61.40	69.92
CBOW (100)	2,179	50.21	56.51
CBOW (300)	1,739	67.98	61.87
CBOW (600)	1,709	65.57	61.87

Table 4.4: Evaluation results by lexicons created from Word2Vec CBOW and fastText Skipgram word embedding models of dimensionalities of 100, 300, and 600.

The lexicon created from the fastText Skipgram model of dimensionality 100 is the only lexicon that has better performance on the Pros and Cons development set than the set of manually annotated seed words. All of the lexicons created using the Word2Vec CBOW models have significantly worse performance than the manually annotated seed words, and than the lexicons created using the Skipgram models on the Pros and Cons development set. However, the lexicons from models of dimensionality

300 and 600 have decent performance on the development set of NoReC$^{\pm}$. The best results on the NoReC$^{\pm}$ development set are achieved by the lexicon created from the 100-dimensional model trained using the Skipgram algorithm. For the lexicons created using the Skipgram models, the accuracies on both datasets decrease as the dimensionality of the model increases.

We have tested the optimal configuration from the lexicon parameter-tuning experiments on word embedding models of dimensionalities 100, 300, and 600. None of the lexicons created from the models with higher dimensionalities achieved better results than the lexicons created from the models used in the parameter-tuning experiments. It is possible that the configurations we found were specific to the model used for lexicon creation, and that other models have different optimal configurations. Finding the optimal configuration for each of the models is beyond the scope of this thesis.

Lexicon expansion using the WordNet seed words

The word embedding model with the best performance in the previous section was used to create a sentiment lexicon from the WordNet seed words. The lexicon was created by extracting six neighbours in two iterations from the fastText Skipgram model of dimensionality 100, which resulted in a lexicon of 2,669 positive and 6,111 negative words. The evaluation results are presented in Table 4.5.

Lexicon description	Lexicon size	Accuracy NoReC$^{\pm}$	Accuracy Pros and Cons
Seed words (adj/adv)	100	**63.81**	**71.45**
Seed words (WordNet)	461	53.50	50.76
Seed words (WordNet) & Skipgram (100)	8,780	49.78	50.95

Table 4.5: Results on Sentiment classification using the sentiment lexicons created from the WordNet seed words. The first two lexicons are the manually annotated seed words and the WordNet seed words. The third lexicon is created from the WordNet seed words and expanded using the Skipgram model of dimensionality 100.

The lexicon created from the WordNet seed words, and expanded using the fastText Skipgram word embedding model, has worse performance than both sets of seed words on both of the evaluation datasets. It achieves accuracies of around 50.0, which can be attributed to a random prediction of labels. We will therefore not use the WordNet seed words for any further experiments. In the following section, we will experiment with domain-specific word embedding models trained on NoReC.

Lexicon expansion using word embedding models trained on NoReC

We hope to achieve better performance on the evaluation datasets by lexicons from a domain-specific corpus trained on reviews, instead of using a neutral corpus trained on news articles. Because of the small number of tokens and lemmas in the Pros and Cons dataset, it was too small for this task. We have trained our own CBOW and Skipgram word embedding models on the training set of NoReC. The word embedding models were trained using the Word2Vec tool created by Mikolov et al. (2013) and implemented in Gensim.[4] Both models use a window size of five, and a dimensionality of 100. The resulting models are quite small compared to the models trained on "Aviskorpuset" *(The Norwegian Newspaper Corpus)*. The NoReC models have a vocabulary of 64,963 words, whereas the models trained on "Aviskorpuset" *(The Norwegian Newspaper Corpus)* contain almost 1,5 million words. We have created sentiment lexicons from the manually annotated set of seed words, and expanded the lexicons using the NoReC word embedding models by extracting six neighbours in two iterations. The evaluation results of the sentiment lexicons created from the NoReC embeddings are presented in Table 4.6.

Lexicon description	Lexicon size	Accuracy NoReC$^{\pm}$	Accuracy Pros and Cons
Seed words (adj/adv)	100	63.81	**71.45**
Seed words (adj/adv) & NoReC Skipgram	2,206	49.78	54.78
Seed words (adj/adv) & NoReC CBOW	1,534	60.74	67.81

Table 4.6: Evaluation results of the sentiment lexicons from the word embedding models trained on NoReC.

Both of the sentiment lexicons created from the NoReC word embedding models have worse performance than the set of manually annotated seed words on both of the evaluation datasets. We cannot attribute the low accuracies to the small vocabulary, since the resulting lexicons are similar in size to the lexicons created from word embedding models trained on "Aviskorpuset" *(The Norwegian Newspaper Corpus)*. We have examined the lexicon created from the NoReC Skipgram model to see if there were any obvious words in the lexicon that could be affecting the evaluation results. The words "også" *(also)*, "men" *(but)*, "de" *(they)*, "ikke" *(not)*, and "det" *(it)*, were added to the positive sentiment lexicon in the first iteration, because they are the top six neighbours of the word "god" *(good)*, which is one of the seed words. In the second iteration, the top six neighbours of these words are added to the lexicon, and none of these new words are sentiment-bearing either. Since the word "god" *(good)* has co-occurred with these words in the corpus, they have high cosine similarity scores in the

[4] https://radimrehurek.com/gensim/

word embedding models.

We have created two new word embedding models containing only verbs, nouns, adjectives, and adverbs from the NoReC training corpus to avoid functional words in the sentiment lexicons. The performance of the resulting lexicons is presented in Table 4.7. The lexicon created from the CBOW model is the only lexicon that outperformed the set of seed words on the development set of NoReC$^{\pm}$. We have also experimented with creating word embedding models from the NoReC training corpus with higher dimensionalities (300 and 600), but we did not see an improvement in the accuracies by the resulting lexicons on the evaluation datasets. In Chapter 5, we will try to enhance these word embedding models using two different approaches.

Lexicon description	Lexicon size	Accuracy NoReC$^{\pm}$	Accuracy Pros and Cons
Seed words (adj/adv)	100	63.81	**71.45**
Seed words (adj/adv) & NoReC Skipgram	2,689	53.07	62.64
Seed words (adj/adv) & NoReC CBOW	2,146	64.03	63.98

Table 4.7: Evaluation results of sentiment lexicons created from word embedding models trained on adjectives, adverbs, verbs, and nouns in the training set of NoReC.

In this section, we have used the seed words to expand the sentiment lexicons using various word embedding models. The lexicon with the best performance was created from the fastText Skipgram model of dimensionality 100, which achieved an accuracy of 70.83 on the development set of NoReC$^{\pm}$, and 77.20 on the development set of the Pros and Cons. In the following section, we will explore a different lexical resource for sentiment lexicon expansion.

4.4.2 Lexicon expansion using the Norwegian Synonymy Test Set

The Norwegian Synonymy Test Set[5] is a lexical resource that also has been consulted for sentiment lexicon expansion in this work. Hammer, Bai, et al. (2014) also used a version of this lexical resource for lexicon creation, however, their lexicons are not available so we will not be able to compare their lexicons to the lexicons created in this work. The Norwegian Synonymy Test Set was collected from the digital version of "Norske synonymer blå ordbok" *(Norwegian dictionary of synonyms)* by Kunnskapsforlaget, and published by Stadsnes, Øverlid, and Velldal (2018). It contains 24,649 headwords and their synonyms. Among the synonyms, there are 106,749 tokens and 30,756 types (Stadsnes, 2018). This resource was created as a standard evaluation dataset for

[5] https://github.com/ltgoslo/norwegian-synonyms

Norwegian word embedding models. We will use this lexical resource for sentiment lexicon expansion, but also in Chapter 5, for retrofitting in Section 5.3, and for the task of intrinsic evaluation of word embedding models in Section 5.4.

We have used the Norwegian Synonymy Test Set to create a sentiment lexicon, by extracting all of the synonyms of the positive and negative words from the manually annotated set of seed words. The evaluation results are presented in Table 4.8. The sentiment lexicon performs worse than the manually annotated seed words on both datasets. We performed a manual error analysis to find out why the lexicon achieves poor results on the evaluation task on the NoReC$^{\pm}$ development set. The sentiment lexicon created from the Norwegian Synonymy Test Set consists of 760 positive and 948 negative words. Since the negative side of the sentiment lexicon is much larger than the positive side, it could mean that the lexicon has more negative hits and favours a negative prediction, even though the distribution of positive and negative reviews is even in the dataset. However, in the development split of NoReC$^{\pm}$, we found that there were 12,303 hits from the positive words, and 7,066 hits from the negative words in the lexicon, so it is not likely that the larger negative vocabulary in the lexicon is responsible for the poor evaluation performance.

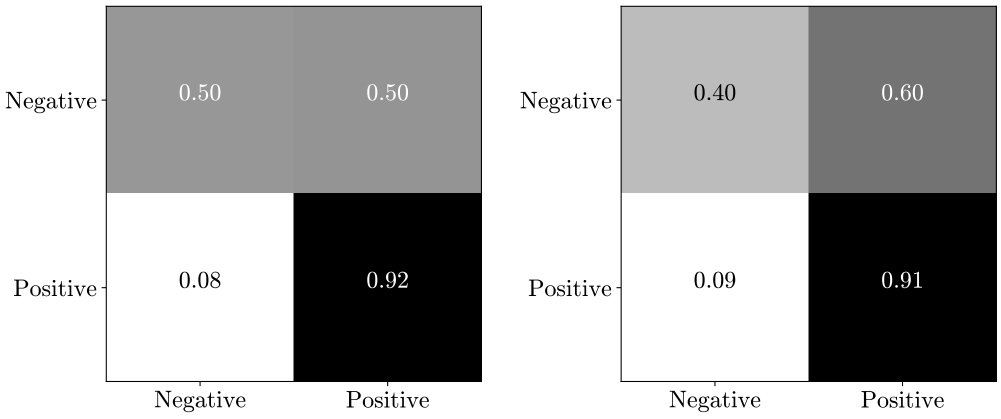
Lexicon description	Lexicon size	Accuracy NoReC$^{\pm}$	Accuracy Pros and Cons
Seed words (Adj/adv)	100	**63.81**	**71.45**
Seed words (Adj/Adv) & Synonyms	1,708	57.67	70.49

Table 4.8: Evaluation results of the manually annotated seed words and the sentiment lexicon created from the Norwegian Synonymy Test Set.

Another possible explanation for these results is that words that are non-sentiment-bearing are added to the lexicon, which was the case for the lexicons created from the NoReC word embedding models. We reviewed the words in the lexicon, and found the word "ikke" *(not)* in the list of negative words, because it is a synonym of the negative seed word "gal" *(crazy)*. In Section 4.2.1, we saw how the word "ikke" *(not)* was used to negate positive sentiment-bearing words in the Pros and Cons dataset, and that it was present mostly in negative summaries, however, it is not used with the same frequency in the NoReC$^{\pm}$ reviews. The word "ikke" *(not)* is present in 388 of the 456 reviews in the development set of NoReC$^{\pm}$, so there is a negative hit in almost 85% of the reviews in the dataset, even though there is an even distribution if positive and negative reviews.

We used the NLTK stopwords to filter out all of the stopwords from the sentiment lexicon. After removing the stopwords, the lexicon achieves even lower accuracies, the accuracy on the development set of NoReC$^{\pm}$ decreased from 57.67 to 56.57, and the

accuracy on the Pros and Cons development set decreased from 70.49 to 65.45. Since the accuracies decreased after removing stopwords from the lexicon, we cannot attribute the low accuracy on the evaluation task to the word "ikke" *(not)* being present in the lexicon. Figure 4.9 presents the confusion matrices on the Pros and Cons evaluation dataset by the sentiment lexicon both with and without stopwords.



(a) Lexicon with stopwords.

(b) Lexicon after removing stopwords.

Figure 4.9: Confusion matrices on the Pros and Cons development set by the lexicons created from the Norwegian Synonymy Test Set.

Both of the lexicons achieve good results for labelling positive summaries, however, the accuracies for the negative predictions are very low. The lexicon which contains the stopwords achieves an accuracy of 50.00, but after removing the stopwords, it decreases to 40.00. The word "ikke" *(not)* was removed from the negative lexicon, and was responsible for the decrease in accuracy along with the other stopwords. In Section 4.2.1, we saw how important the word "ikke" *(not)* and other functional words were for predicting the sentiment of summaries in the Pros and Cons development set. In the following section, we will describe some of the problems that can arise from the automatic methods for sentiment lexicon expansion.

4.5 Challenges in the automatic creation of sentiment lexicons

In the previous sections, we encountered different challenges that can arise from the automatic creation of sentiment lexicons. A sentiment lexicon created from a word embedding model relies on the information in the model. A word embedding model is created from a corpus, and is based on the distributional hypothesis which states that similar words occur in similar contexts (Firth, 1957; Harris, 1954; Rubenstein and Goodenough, 1965). Antonyms and synonyms are often used in the same context

(Charles and Miller, 1989), which results in similar vector representations in the word embedding model. If a word is replaced by its antonym in a sentence, the sentence is often still syntactically and semantically correct, but the meaning of the sentence has shifted in the opposite direction. When creating sentiment lexicons from word embedding models, antonyms are often extracted and added to the lexicon, since they have high cosine similarity scores. For example, in Table 4.9, we show some examples of words and their neighbours in the fastText Skipgram word embedding model of dimensionality 100. Each of these words have one of their antonyms among their top five neighbours.

Word	Neighbours	Cosine similarity
Positiv	"positiv" *(positive)*	90.49
	"positiv/negativ" *(positive/negative)*	86.54
	"positiv." *(positive.)*	83.54
	"negativ" *(negative)*	82.89
	"positivt" *(positive)*	82.15
Lav	"høy" *(tall)*	88.76
	"lave" *(short)*	84.99
	"lavere" *(shorter)*	82.94
	"laveste" *(shortest)*	79.52
	"lavest" *(shortest)*	78.41
Sen	"sein" *(late)*	84.69
	"før" *(before)*	76.20
	"allerede" *(already)*	73.58
	"først" *(first)*	71.75
	"seinere" *(later)*	71.47
Lys	"lys-mørk" *(light-dark)*	80.62
	"lys-mørke" *(light-dark)*	78.73
	"lyse" *(shine)*	75.66
	"mørk" *(dark)*	74.85
	"lyskasterlys" *(floodlight-light)*	74.21

Table 4.9: The five nearest neighbours of the words "positiv" *(positive)*, "lav" *(short)*, "sen" *(late)*, and "lys" *(bright)* in the Skipgram word embedding model of dimensionality 100.

Including words that are non-sentiment-bearing, or mislabelling words in the lexicon, can have a great impact on the evaluation results. Semantically neutral words or named entities are sometimes added to the sentiment lexicon even though they are non-sentiment-bearing. When a lexicon is created from a review dataset, neutral words can be added to the positive or negative part of the sentiment lexicon if they

appear in very positive or negative reviews. It can be difficult for a classifier to distinguish between neutral and sentiment-bearing words occuring in the same labeled text (Taboada, Brooke, Tofiloski, Voll, & Stede, 2011). Neutral words may also be added to the lexicon when using word embedding models for lexicon expansion, because two words can be semantically similar even if they do not have the same sentiment orientation. The word "dukke" *(doll)* from the WordNet seed words, and the word "film" *(movie)* from the Potts lexicon extracted from NoReC$^{\pm}$, are examples of words that are not sentiment-bearing, and influenced the evaluation results. Removing stopwords such as the word "ikke" *(not)* is relatively simple, but it is much more difficult to remove for example non-sentiment-bearing nouns or verbs from the lexicons. These words can be removed manually after the creation of the lexicon, however this approach can be very time-consuming and is outside the scope of this thesis.

In this chapter, we have explored various methods for sentiment lexicon creation. We have extracted several lexicons from NoReC$^{\pm}$ and the Pros and Cons, using the method by Potts (2011). Two different sets of seed words were identified and later used for sentiment lexicon expansion. We have performed experiments on extracting words from word embedding models, and found a configuration which consisted of extracting six neighbours in two iterations, which produced the best results on our evaluation task. We were also faced with some of the challenges associated with automatic approaches for sentiment lexicon creation. In the following chapter, we will explore two different approaches for enhancing word embedding models used for sentiment lexicon creation.

Chapter 5

Improving word embeddings

We have shown in Chapter 4 that some of the sentiment lexicons created using information from word embedding models had better performance than the set of sentiment-bearing seed words. Even though some of the lexicons had better performance than the seed words, we would like to enhance the models in order to achieve even better results on our evaluation task. In this chapter, we will describe two different approaches for improving the quality of word embedding models. The first approach is fine-tuning the word embeddings in a convolutional neural network. The second approach for fine-tuning the word embedding models is a method called retrofitting, which involves combining information from a word embedding model and a resource containing lexical relations.

Each fine-tuned model will be evaluated using two different approaches. We will create sentiment lexicons from each model, which will be evaluated using the standard evaluation approach described in Section 4.1 in Chapter 4. The models will also be evaluated on two different intrinsic evaluation tasks: synonymy detection and analogy prediction, which are standard evaluation tasks for word embedding models. We will discuss these tasks further in Section 5.4. For all of the models that are fine-tuned in a CNN, we also report the CNN accuracies.

To be able to use CNN's to fine-tune the word embeddings, we first have to perform preliminary tuning experiments on the network to find the optimal configuration for our task and dataset. In the following section, we will provide detailed descriptions of the baseline configuration of our network. We will also assess the variance in the network, and perform experiments on tuning the hyperparameters of the CNN. We will describe different approaches for fine-tuning word embedding models in a CNN in Section 5.2. In Section 5.3, we will use retrofitting to combine the information from various word embedding models, with the information contained in the Norwegian Synonymy Test Set. In the final section of this chapter, in Section 5.4, we will present the two standard methods used for the intrinsic evaluation of word embedding models.

5.1 The convolutional neural architecture

In this work, the main application of convolutional neural networks is to fine-tune the word embedding models used for sentiment lexicon creation. The goal is to get more robust and accurate vector representations for sentiment-bearing words, and that the models contain less similar vector representations for antonyms. Word embedding models are fine-tuned in the embedding layer in a convolutional neural network. Since we want to find the optimal configuration of the network, and this configuration is task-specific and also specific to the dataset that is being used, we have to perform tuning experiments on the parameters in the network. To the best of our knowledge, these are the first sentiment analysis experiments using only very positive and negative reviews from NoReC in a CNN, so we have to tune the parameters of the network to find the optimal configuration for our task. The system by Y. Kim (2014) which we will use as a baseline for our CNN, and the tuning of the hyperparameters will be presented in the following sections. Since improving the accuracy of the CNN on sentiment analysis is not the main objective of this project, we will only perform preliminary experiments on tuning the hyperparameters.

Y. Kim (2014) found that the top performing hyperparameter configuration in a CNN is task-specific. Since it is task-specific, his best configurations may not necessarily be the optimal configuration for our task. We will use the top configuration from Y. Kim (2014), which is described in Section 2.5.1 in Chapter 2, as a baseline for our CNN, and perform hyperparameter tuning experiments using random search (Bergstra & Bengio, 2012), to find the best values for the parameters of our network.

We are performing hyperparameter tuning experiments on the network by testing different values for various parameters, and then comparing the CNN configurations in terms of the achieved CNN accuracies for sentiment classification. Unfortunately, there is no guarantee that the configuration that achieves the best CNN accuracy also produces the best fine-tuned word embedding models for sentiment lexicon creation. To find the optimal configuration for fine-tuning the word embedding models, we would have to train the embedding layer, and create sentiment lexicons from the resulting models for each of the parameters that is tested. Using this approach, we would probably find a good configuration of the CNN for fine-tuning word embedding models for sentiment lexicon creation, but it would be extremely time-consuming and it is beyond the scope of this thesis. We will perform experiments to find the best configuration of the CNN for sentiment analysis on the NoReC$^{\pm}$ training dataset, and hope that this is also a good configuration for fine-tuning the word embedding models used to create sentiment lexicons. In the following section, we will assess the variance in the network.

5.1.1 Variance in the network

Each time a neural network is trained, the weights are initialized to a small random number. Because of the random initialisation, a network can be trained on the same data multiple times, and each time produce different results on the same test data. If we are able to control the randomness in the network, it will produce the same results each time the network is trained, and we would be able to replicate our results. The main reason for the variance in the network is the random initialization of the weights in the network. The standard weight initialization in Keras is the Xavier uniform initialization, introduced by Glorot, Xavier and Bengio, Yoshua (2010). All the weighs in the network are randomly initialized from a Gaussian distribution with a mean and variance of zero.

There are several approaches for controlling randomness in the network. We can train and test the network a number of times and calculate the average accuracies over all the runs. This is costly in terms of training times, but can lead to robust results. We can also use k-fold cross validation and run the network k times, each time training on $k-1$ folds and testing on the last fold. Another approach for reducing the variance is to fix the random seed generator. All the weights in the network are initialized to a random number based on a random seed, so if the random seed generator is fixed, the weights would be initialized to the same numbers each time. The random seed also has to be fixed for all the third party libraries.

To reduce the variance in a network, we can also remove parallelisation in Tensorflow.[1] When running parallel jobs on multiple threads, we cannot control the work flow of the individual threads. Each time the network is trained, the weights are updated by different threads, which complete their jobs at different times. Since the calculations of each layer depend on the calculations in the previous layers, the weight updates will be different every run if the threads finish at different times. To eliminate this problem, we can run Tensorflow using only one thread, while at the same time fixing the random seed generator for all libraries. This approach should almost completely eliminate randomness and produce the same results every time, however it is extremely time-consuming. We have tested some of the approaches for controlling randomness to reduce the variance in the CNN. The results are presented in Table 5.1.

We ran each test ten times, and for each measure, we report the minimum, maximum, and average accuracy, as well as the average training time. When not implementing any measure for controlling the variance in the network, the accuracy ranges from 90.57 to 92.55. Fixing the random seed generator in Python and all third party libraries decreased the variance, and the training time was roughly the same as when not implementing any strategies for controlling the variance. However, to almost completely remove the variance in the network, we can fix the random seed in addition

[1] https://www.tensorflow.org

to removing parallelisation in Tensorflow. When implementing this measure for controlling the variance, the network will achieve almost the same accuracy every time the network is trained, but it greatly increases the training time. Training the network using the first two strategies took roughly 23 minutes, but for the third strategy, the training time took more than two hours.

Strategy	Minimum accuracy	Maximum accuracy	Average accuracy	Average training time in seconds
No measure	90.57	92.55	91.36	1385
Fix random seed	91.31	92.30	91.77	1383
Remove parallelisation	91.06	91.56	91.23	8305

Table 5.1: Measuring the variance in a CNN using three different strategies for controlling the variance. For all three strategies, the network was trained and tested 10 times. Removing parallisation in Tensorflow also included fixing the random seed generator.

The baseline configuration of the network was implemented for these experiments, which uses relatively small values for the filter sizes and the number of filters used for each filter size. We will test larger values for the parameters in the configuration of the network, which can increase training time. It is therefore not feasible to implement all of these measures for controlling the variance, because it is too time-consuming. We will however fix the random seed, since it reduces the variance in the network and it does not require more training time than when not implementing any measures for controlling the variance. Knowing that there is variance in a network, we can attribute small differences in the performance to the variance. Small differences in accuracies between two configurations could simply be due to the variance in the network, and does not necessarily mean that one of the configurations is better than the other.

5.1.2 Testing the baseline configuration

We are using the configuration by Y. Kim (2014), which is described in Section 2.5.1 in Chapter 2, as a baseline configuration. To be able to use the configuration for our task and dataset, we had to make some decisions about the parameters that are specific for our datasets. We will describe these parameters in the following paragraphs.

Implementation environment Our convolutional neural network is implemented in Keras,[2] which is a high-level deep learning library for Python, developed by Chollet et al. (2015). Our installation uses Tensorflow as a back-end machine learning framework,

[2] https://keras.io

however, Keras also has support for other frameworks such as Theano[3] and CNTK.[4] We are using Keras in this project because it is a framework that allows us to focus on more high-level concepts.

Batch size and the number of epochs The network in the baseline configuration is trained using batches of input. When updating the network in batches, all of the training samples in the batch are passed through the network and the loss is updated after each sample. After the loss is calculated over the whole batch, it is propagated back trough the network, and all of the weights are updated. The batch sizes that we have chosen are 72 for NoReC$^{\pm}$ CNN train and 36 for the Pros and Cons^{+} CNN train. We tested different values for the batch size for the two datasets, and these batch sizes achieved the highest accuracies. The baseline configuration of the network that we have implemented is trained in 36 epochs. However, when testing different optimizers later in this chapter, we had to increase the number of epochs significantly, for the network to converge. A measure for early stopping is used to prevent the network from overfitting, which is based on the loss in the network. The network will stop training if the loss is less than 0.01 over five consecutive epochs.

Input length All of the input samples in the network have to be of the same length. Since we are using CNN's to fine-tune the words in the word embedding models, we wanted to make sure that the input lengths covered the majority of the words in the reviews, in order for the classifier to be trained using as many words as possible from the dataset. If the input is longer than the chosen input length, the rest of the text in the input is removed and not passed through the network. If the input is shorter, it will be padded at the end of the input sequence. The same vector representation is used for all of the input words that are not present in the word embedding model. For the reviews in the NoReC$^{\pm}$ CNN train, we have chosen an input length of 2,048 words, because this covers the majority of reviews in the training set (99.98%). The longest review in the training set consists of 4,373 words, whereas the average length of reviews in the training set is 408 words. If we were to include all the words in the reviews in the input length, the average review would need padding equivalent to ten times the input length, which greatly increases the training time. For the Pros and Cons^{+} CNN train, we have chosen an input length of 100 words which covers all of the words in the summaries in the dataset.

Results of testing the baseline configuration

The results of testing the baseline configuration on both datasets are presented in Table 5.2. The baseline configuration has a better performance on the Pros and Cons CNN dev than the NoReC$^{\pm}$ CNN dev. These results are likely due to the fact that the Pros and Cons dataset is more similar to the datasets used by Y. Kim (2014) than NoReC. The

[3] http://deeplearning.net/software/theano/
[4] https://www.microsoft.com/en-us/cognitive-toolkit/

Pros and Cons dataset and the datasets used by Y. Kim (2014) mostly consist of short sentiment-bearing sentences, whereas NoReC consists of long reviews with an average length of 426 words. The average length of reviews is much smaller in the Pros and Cons dataset than in NoReC, so the relatively small filter sizes may be more beneficial for shorter texts. Another explanation is in the difference in the classification tasks. The Pros and Cons dataset contains the most sentiment-bearing sentences from the reviews from DinSide, whereas NoReC contains the full reviews. Therefore, it might be easier to predict the correct labels of the summaries in the Pros and Cons dataset than of the reviews in NoReC.

CNN Configuration	Accuracy NoReC$^{\pm}$ CNN dev	Accuracy Pros and Cons CNN dev
Baseline	92.05	97.05

Table 5.2: Testing the baseline configuration of hyper-parameters from Y. Kim, 2014 on the NoReC$^{\pm}$ CNN dev and the Pros and Cons CNN dev.

5.1.3 Fine-tuning the hyperparameters of the CNN

Only the NoReC$^{\pm}$ training dataset was used for tuning the values of the hyperparameters of the CNN, because it was not feasible to perform the tuning experiments on both datasets. We chose to use NoReC for this task because it is larger than the Pros and Cons dataset, and because of the results in Table 5.2. The baseline configuration has worse performance on NoReC$^{\pm}$ than the Pros and Cons dataset, so there is more room for improvement for NoReC$^{\pm}$.

In the experiments, we wanted to test different values for some of the hyperparameters that we thought could benefit from using other values more targeted towards the specific tasks in our project. We chose to experiment with filter sizes, the number of filters for each size, and the dropout rate. The tuning of the hyperparameters was done using a random search over a list of defined values for filter sizes, the number of filters for each filter size, and dropout. When tuning the hyperparameters using random seach, a selected number of random samples are chosen from a predifined search space (Bergstra & Bengio, 2012). We have tested 250 out of the 960 possible configurations, from the values used in the random search in Table 5.3.

All of the parameters in a neural network are dependent, so we cannot know exactly how a single parameter affects the performance of the network. However, we can try to interpret the results of the tests and see if there are any obvious patterns. We have plotted each of the tested parameters against the accuracy of the network, using the

framework Matplotlib.[5] The boxes represent the data between the upper and lower quartile, whereas the hard lines inside the boxes represent the medians, and the dashed lines represent the means. The lines with the whiskers at the ends represent the ranges of the data points, and the circles outside the boxes are the extreme outliers. We also report the 10 best performing configurations, which are presented in Table 5.4. The results of tuning each of the parameters will be discussed individually in the following paragraphs.

Feature	Value
Filter sizes	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 15, 20, 30, 50
Number of filters	10, 25, 50, 75, 100, 200
Dropout	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9

Table 5.3: Values used for tuning the CNN parameters in a random search.

Filter size	Number of filters	Dropout rate	CNN accuracy
2, 5, 8	200	0.5	93.31
2, 3, 6	200	0.5	93.15
2, 5, 8	200	0.7	93.05
2, 6, 7	200	0.5	93.05
3, 4, 9	100	0.6	93.05
1, 3, 9	100	0.8	93.04
1, 3, 10	200	0.6	92.81
2, 3, 4	200	0.2	92.81
2, 4, 8	200	0.5	92.80
2, 5, 15	200	0.5	92.80

Table 5.4: The top ten configurations from the random search for tuning the hyperparameters of the CNN.

Filter sizes The best configuration by Y. Kim (2014) was task-specific and dataset-specific. In our experiments, we always use a combination of three different filter sizes. We wanted to experiment with larger filter sizes since the documents in NoReC are much longer than the documents (sentences) used in his work. We also wanted to test larger filter sizes, since Bergem (2018) found that filter sizes of 15–20 had the best performance on multi-class sentiment prediction on the NoReC dataset. Since the average sentence length in the dataset is 16 words, he suggests that the classifier has

[5] https://matplotlib.org/api/_as_gen/matplotlib.pyplot.boxplot.html

the best performance when the filters are able to capture information from the whole sentence. In our experiments, we also want to find out how the accuracy is affected when using even larger filter sizes, to try to capture longer dependencies. In addition to testing larger filters, we also want to test filters of size one, which have been found to capture unigram information (Jacovi, Sar Shalom, and Goldberg 2018). The results of testing different filter sizes are presented in Figure 5.1.

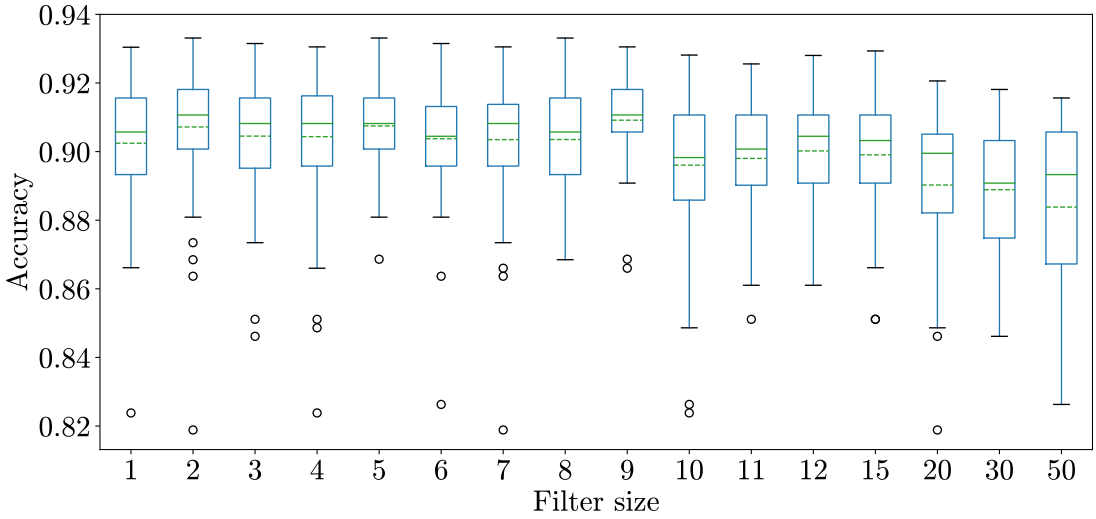


Figure 5.1: Filter size and CNN accuracy.

In our implementation of the network, the convolutional layer always has three different filter sizes. It is difficult to interpret the results of the filters individually, because each filter is always influenced by two other filters. For filter sizes between 20 and 50, the lower end of the range is much lower than for the smaller filter sizes and they also have a much larger range of accuracies between the upper and lower quartile. Among the top 10 configurations in Table 5.4, all of the configurations except for one used filter sizes between 1 and 10. Most of the configurations have one or two small filters (1–3) and one or two filters of a larger size (5–10).

Only one of these configurations used the filter sizes that had the best performance in the work by Bergem (2018), whereas all of the other configurations used smaller filter sizes. In our project we are performing binary classification, the network is predicting whether a review is positive or negative, whereas Bergem (2018) performs multi-class prediction, the network is predicting the rating of the review (rating 1–6). It can seem like the polarity is predicted from individual words or short dependencies whereas predicting review ratings require more contextual information and longer sequences of words. Another possible explanation is that we are using different subsets of the NoReC dataset. He uses the full training split for training the network and the full

development split for validating the results, while we are only using the reviews in NoReC$^{\pm}$. The subset of the dataset used in our project only contains the most positive and negative reviews, which are most likely easier to label than mildly positive or negative reviews.

Number of filters The second parameter that was tuned in this experiment was the number of filters used for each filter size. The CNN accuracies are presented in Figure 5.2. In these experiments, using a larger number of filters per filter size increases the performance of the network. We could have benefited from testing even more filters per filter size, but these large values greatly influence the training time of the network. Among the top 10 systems in Table 5.4, all of the systems use 100 or more filters, and 8 of the systems used 200 filters per filter size.

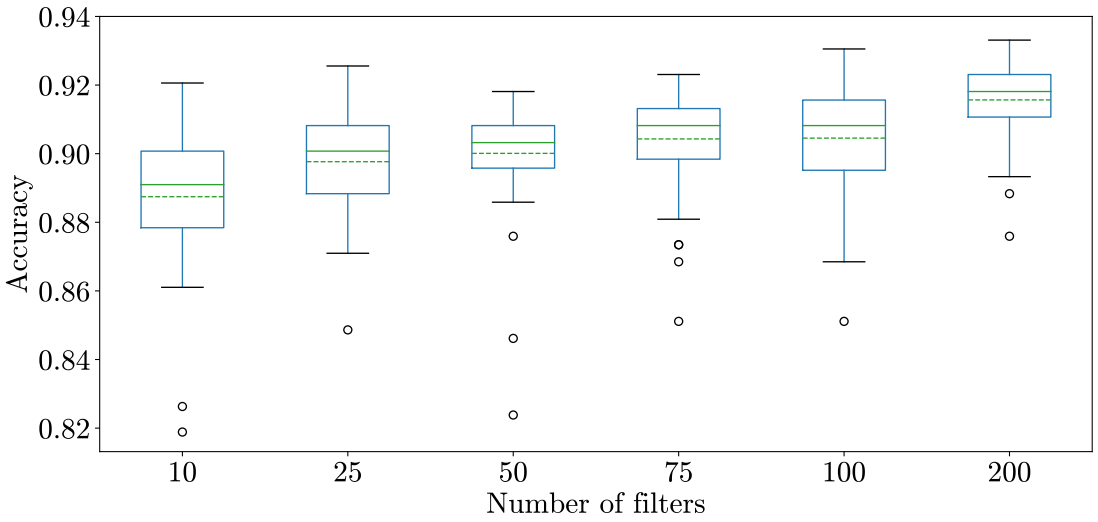


Figure 5.2: Number of filters per filter size and CNN accuracy.

Dropout The third hyperparameter that we tested was the dropout rate. The reviews in the NoReC$^{\pm}$ CNN train are long, the average review length is 409 words. Since the reviews are long, we can expect them to contain a lot of information that is not relevant to the sentiment of the review. We explored different dropout rates to find out if the network benefited from regularization. The results are presented in Figure 5.3. The CNN can achieve good accuracies from all the dropout rates that were tested. The dropout rates that have the best median performances as well as the highest ranges of accuracies are rates of 0.3 and 0.5.

Among the top ten configurations in Table 5.4, nine of the dropout rates are above 0.5, and half of the configurations use a dropout rate of 0.5. Since only one of the configurations uses a lower dropout rate, it is likely that the network benefits from

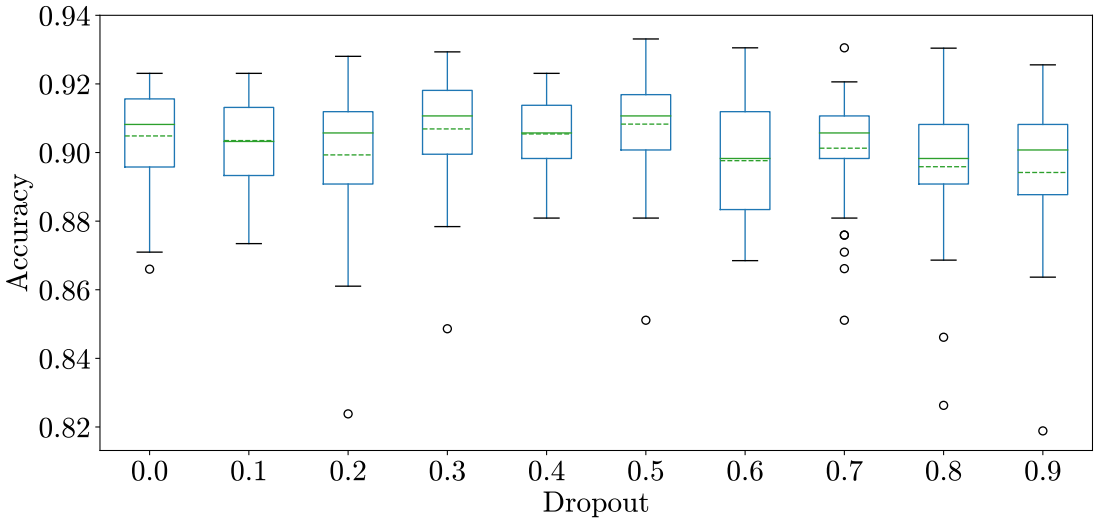


Figure 5.3: Dropout and CNN accuracy.

some kind of regularization. Most of the configurations use a larger number of filters along with medium to high dropout rates. It seems that using dropout rates above 0.5 in combination with a large number of filters is beneficial for the network, and produces the best accuracies.

The best configuration from the tuning experiments uses filters of size 2, 5, and 8, a dropout rate of 0.5, and 200 filters per filter size. For the other parameters that we have not tuned, we will use the configuration by Y. Kim (2014), which was described in Section 2.5.1 in Chapter 2. This configuration uses Softmax as the activation function in the output layer and ReLU in the other layers. The loss function is the categorical cross entropy loss, the pooling strategy is max pooling, and Adadelta is used as the optimizer. In the following section, we will explore methods for fine-tuning word embedding models in a CNN using the configuration of the parameters which we have described in this paragraph.

5.2 Fine-tuning word embedding models in a CNN

In the previous section, we explored different CNN configurations to find the optimal configuration for our task, which will be used to fine-tune the word embedding models. The goal of the fine-tuning is to get more accurate representations for word vectors of the sentiment-bearing words. Another goal is to make the representations of antonyms less similar, which is an important task when using the word embedding models for sentiment lexicon creation. If antonyms have high cosine similarity scores, there is a high probability that they will be added to the lexicon, and cause poor results

on evaluation task. We will explore two main approaches for fine-tuning the word embeddings in a CNN, both in a single-channel and in a multi-channel convolutional network, like in Y. Kim (2014). In a single-channel CNN, there is only one set of inputs which is passed through the network, whereas in multi-channel CNN's, two or more input representations are passed through the convolutional layer and summed before performing the pooling operation. This architecture allows us have two different representations of the input text, for example using the representations from two different word embedding models. In what follows, we start with exploring single-channel dynamic word embeddings.

5.2.1 Fine-tuning the Skipgram word embedding model

The top configuration of the convolutional neural network from the previous section will be used to fine-tune the word embedding models. First, we are going to fine-tune the fastText Skipgram model with dimensionality of 100, which had the best performance among the word embedding models trained on "Aviskorpuset" *(The Norwegian Newspaper Corpus)* in Section 4.4.1 in Chapter 4. The resulting fine-tuned word embedding models are then used to create sentiment lexicons. Since the word embedding models are fine-tuned on sentiment data, we are hoping to see better results from these lexicons than the lexicons created from the non-fine-tuned word embedding models. Both the NoReC$^{\pm}$ and the Pros and Cons^{+} CNN train have been used for this task. The evaluation results of the sentiment lexicons created from the fine-tuned Skipgram word embedding models are presented in Table 5.5.

Lexicon description	Lexicon size	Accuracy NoReC$^{\pm}$	Accuracy Pros and Cons	CNN accuracy
Seed words (Adj/adv)	100	63.81	71.45	-
Non-fine-tuned embeddings	1,639	**70.39**	**77.01**	-
Fine-tuned CNN (NoReC$^{\pm}$)	1,874	67.10	70.44	91.56
Fine-tuned CNN (Pros and Cons)	1,885	66.44	71.01	96.84

Table 5.5: Results of evaluating the sentiment lexicons created from the fine-tuned Skipgram word embedding model of dimensionality 100. The fine-tuned models were fine-tuned on the dataset in the parentheses.

The non-fine-tuned model with static weights in the embedding layer, was used in the tuning experiments of the CNN on the NoReC$^{\pm}$ CNN train, and achieved an accuracy of 93.31, whereas the model that was fine-tuned on the same dataset only achieved a CNN accuracy of 91.56. The lexicon created from the non-fine-tuned word embedding model outperforms both of the lexicons that were created from the fine-tuned models. We have manually inspected the lexicon created from the model which

was fine-tuned on the Pros and Cons$^+$ CNN train, and we found words in the positive list such as "også" *(also)*, "er" *(is)*, and "ok" *(ok)*. There are 230 negative reviews in our NoReC$^\pm$ evaluation dataset, and these words from the positive lexicon occur 206 times in 126 different negative reviews. Among the negative words, we find words such as "mulig" *(possible)*, "forstå" *(understand)*, "sånn" *(like)*, "sånt" *(like that)*, and "så" *(so)*. These words occur 650 times in 180 of the 226 positive reviews in the NoReC$^\pm$ development split. After removing stopwords from these lexicons, the total number of hits in the evaluation datasets decreased, however, the accuracies remained roughly the same.

One of the main goals of fine-tuning the word embedding models is to make the representations of antonyms less similar. We have extracted the top ten neighbours of the word "god" *(good)* from the non-fine-tuned and the fine-tuned Skipgram model of dimensionality 100. The top ten neighbours are presented in Table 5.6. The English translation of these words is presented in Table A.2 in Appendix A

Word	Cosine similarity	Word	Cosine similarity
bra	83.50	bra	81.87
kjempegod	79.86	kjempegod	76.73
brukbar	77.84	knallgodet	75.72
knallgodet	77.30	utmerke	75.55
kjempegodet	77.26	kjempegodet	75.14
utmerket	77.12	utmerket	75.03
knallgod	76.70	fin	74.81
fin	76.24	brukbar	74.45
dårlig	76.18	knallgod	73.98
utmerke	75.61	kjempegode	72.94

Table 5.6: The top ten neighbours of the word "god" *(good)*, in the original word embedding model (left) and in the fine-tuned word embedding model (right).

The antonym "dårlig" *(bad)* is among the top ten neighbours in the non-fine-tuned model, however, it is only the 45th closest word in the fine-tuned word embedding model, with a cosine similarity of 0.66. This is an example of how fine-tuning the word embedding model made the vector representations of antonyms less similar. However, we had to search thoroughly to find a word that had a different representation for antonyms in the non-fine-tuned and the fine-tuned model. All of the words from Table 4.9 in Chapter 4 still had their antonyms among their top five neighbours. The goal of fine-tuning the models is to get more accurate representations of the words, but in our case, the representations had not changed very much. We will therefore explore

different approaches for fine-tuning the word embedding models in a CNN in the following sections.

5.2.2 Catastrophic forgetting

The performance by the lexicons created from the fine-tuned models in Table 5.5 was lower than the performance by the lexicons created from the non-fine-tuned word embedding models. Large updates to the weights in the embedding layer in the first epochs of training the network can have caused these results. Since all of the weights of the network, except for the weights in the embedding layer are initialized to random numbers, we can expect the loss to be quite large in the first epochs of training. If the weights in the embedding layer are updated too quickly, we can lose important information in the word vectors, and the embeddings will be destroyed for the rest of the training. This is referred to as "catastrophic forgetting" (McCloskey and Cohen, 1989; Ratcliff, 1990), which occurs when a network is trained on task A, then continues training on task B, and forgets the representations that were important for task A. Catastrophic forgetting may explain why the lexicons created from the non-fine-tuned word embedding models perform better than the lexicons created from the fine-tuned word embedding models. In what follows, we will explore approaches for dealing with catastrophic forgetting.

Gradual unfreezing of the weights in the embedding layer

The first approach for dealing with catastrophic forgetting is gradual unfreezing of the weights in the embedding layer. In this approach, the weights in the embedding layer are static in the first epochs of training, to avoid large updates to the weights. After a certain number of epochs, or when the loss or accuracy reach a set threshold, the weights in the embedding layer are trained along with the rest of the network. This approach is called "gradual unfreezing". The models created from gradually unfreezing the weights in this work were used to create sentiment lexicons. The evaluation results of these lexicons as well as the CNN accuracies, are presented in Figure 5.4.

All of the accuracies obtained by the networks trained using gradual unfreezing, were in the range between 90.90 and 93.05. There does not seem to be a clear relationship between the number of epochs used for gradual unfreezing and the CNN accuracy. The models trained using gradual unfreezing were used for sentiment lexicon creation. For both evaluation datasets, the lowest accuracies are achieved by the lexicons created from models that start training the weights in the embedding layer in the first few epochs. The model that was trained using zero epochs for gradual unfreezing had dynamic weights in the embedding layer from the first epoch. The highest accuracies are obtained from the lexicons from models that start training the weights in the embedding layer in the final epochs, after the network starts to converge.

For the development set of the Pros and Cons, none of the lexicons created from the fine-tuned models outperform the lexicon created from the non-fine-tuned model, which achieves an accuracy of 77.01. On NoReC$^{\pm}$ dev, the lexicons created from the fine-tuned models outperform the lexicon from the non-fine-tuned model when the weights in the embedding layer are static for 6 and 12 epochs, and then trained along with the network. We also tried to gradually unfreeze the weights in the embedding layer using the Pros and Cons^{+} CNN training dataset, but all of the lexicons created from these models had considerably lower accuracies than the lexicons created from the non-fine-tuned models.

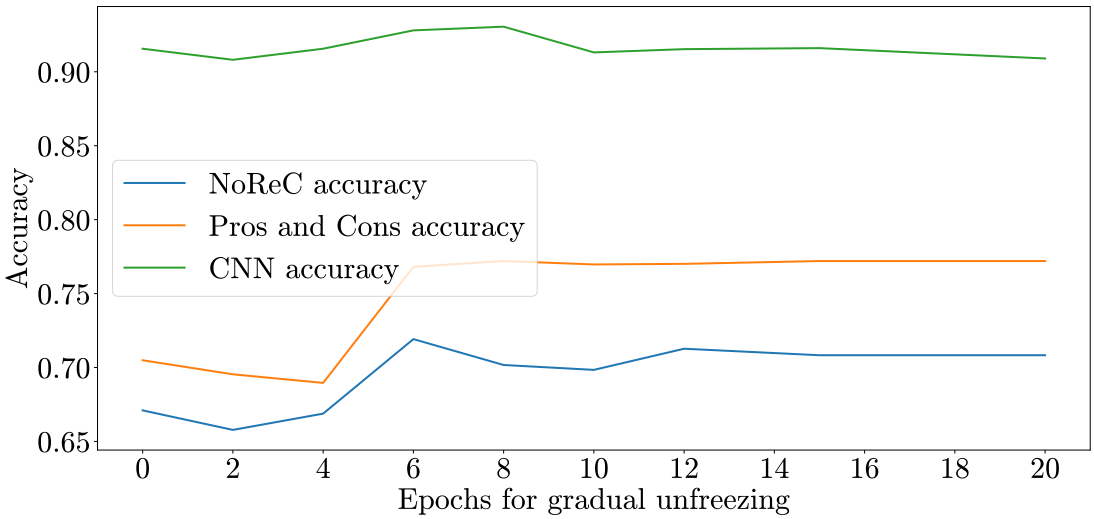


Figure 5.4: CNN accuracies and evaluation results for sentiment lexicons created from the fastText Skipgram word embedding model of dimensionality 100, fine-tuned using gradual unfreezing in a CNN. For each of the numbers on the x-axis, a CNN model was trained with static weights in the embedding layer for the number of epochs corresponding to the numbers of the x-axis, and then trained along with the rest of the network. Each model that was trained using gradual unfreezing for n epochs was used for sentiment lexicon creation. For each model, we report the CNN accuracy, as well as the evaluation results on the development sets of NoReC$^{\pm}$ and the Pros and Cons by the sentiment lexicon created from the model.

Testing the influence of optimizers

In the second approach for dealing with catastrophic forgetting, we will experiment with different optimizers and learning rates. Large updates to the weights in the first epochs can cause great shifts in the weights in the embedding layer, which can damage the word embeddings. Optimizers have a learning rate, which determines how much the weights in the network are updated in each epoch. Low learning rates cause very small updates to the weights, and the network converges slowly, which cause long

training times. Large learning rates cause large updates to the weights, and if the learning rate is too high, the network may not converge. The optimizer used by Y. Kim (2014) and Zhang and Wallace (2017) is Adadelta (Zeiler, 2012), which is an adaptive optimizer. Optimizers with adaptive learning rates have learning rates that change over time. For optimizers such as SGD, we can manually tune the learning rate to fit our task. We have fine-tuned word embedding models using both adaptive optimizers and SGD with different learning rates. The sentiment lexicons created from the resulting models are presented in Table 5.7.

Lexicon description	Lexicon size	Accuracy NoReC$^{\pm}$	Accuracy Pros and Cons	CNN accuracy
Seed words (adj/adv)	100	63.81	71.45	-
Non-fine-tuned embeddings	1639	**70.83**	**77.20**	93.31
Fine-tuned Adadelta	1,874	67.10	70.49	91.56
Fine-tuned Adam	1,880	50.00	67.17	91.06
Fine-tuned RMSprop	1,877	60.30	70.24	91.31
Fine-tuned SGD (0.01))	1,887	**70.83**	77.15	72.45
Fine-tuned SGD (0.03))	1,895	**70.83**	77.15	74.93
Fine-tuned SGD (0.05)	1,887	**70.83**	77.15	79.15
Fine-tuned SGD (0.08)	1,890	**70.83**	77.15	82.63
Fine-tuned SGD (0.1)	1,894	**70.83**	77.15	80.89
Fine-tuned SGD (0.3)	1,894	**70.83**	77.15	89.57

Table 5.7: Testing the influence of optimizers on the fine-tuning of the fastText Skipgram word embedding model of dimensionality 100. The accuracies on the datasets are the results obtained by the resulting lexicons from the models trained using different optimizers, whereas the CNN accuracies represent the accuracies achieved by the model in a CNN. The number in the parentheses for the SGD represents the learning rate.

There are large differences between the models fine-tuned using different optimizers, in terms of both CNN training times, CNN accuracies, and the accuracies obtained from the resulting sentiment lexicons. The training times for the networks with adaptive learning rates were much lower than when using SGD with a fixed learning rate. With adaptive learning rates, the network stops training after 10–20 epochs, whereas for SGD, the network was trained for 40–60 epochs. The CNN accuracies are much higher when using the adaptive optimizers such as Adadelta, Adam, and RMSprop, as opposed to using SGD, which achieves low accuracies for low learning rates and a decent accuracy for a learning rate of 0.3.

All of the lexicons created from the fine-tuned models using SGD as the optimizing routine, perform much better than the lexicons created from models that were trained using adaptive optimizers. Using an optimizer with a fixed learning rate seems to eliminate the effects of catastrophic forgetting, but the resulting lexicons have almost exactly the same performance as the lexicons created from the non-fine-tuned models. Since they all achieve roughly the same results, it seems that the fine-tuning of the word embeddings using the fixed learning rates did not have effect on the resulting lexicons. A manual inspection of the fine-tuned word embedding models also showed extremely small changes in the vector representations of the words, even in models that were trained using large learning rates.

In this section, we have explored various approaches for fine-tuning the fastText Skipgram word embedding model of dimensionality 100. We have not seen any major improvements in the accuracies obtained by the resulting sentiment lexicons. In the following section, we will fine-tune the domain-specific word embedding models created from the NoReC training corpus, to find out if word embedding models trained on a different kind of corpus benefit from fine-tuning.

5.2.3 Fine-tuning the NoReC word embeddings

In this section, we are going to fine-tune the CBOW and Skipgram word embedding models, trained using only nouns, adjectives, verbs, and adverbs from the NoReC training corpus. We will fine-tune the models using both the optimal configuration from the hyper-parameter tuning experiments in Section 5.1.3, and using gradual unfreezing of the weights in the embedding layer. We will use the configuration that achieved the best accuracies, which kept the weights in the embedding layer static for the first six epochs, and then trained along with the network. The results are presented in Table 5.8.

The overall accuracies by the lexicons created using the CBOW models are much higher than the accuracies by the lexicons from the Skipgram models. Some of the lexicons from the CBOW models outperform both the manually annotated set of seed words and the lexicons created from the non-fine-tuned model. The two lexicons that achieved the best accuracies were both created from CBOW models, which were fine-tuned on the NoReC$^{\pm}$ CNN train. The top accuracy on NoReC$^{\pm}$ dev was achieved by the lexicon from the model that was fine-tuned using gradual unfreezing, whereas the top accuracy on the Pros and Cons dev was achieved by a lexicon from a model that was fine-tuned in a basic CNN. We were expecting the lexicons created from word embedding models fine-tuned on the Pros and Cons^{+} CNN train to have the best performance on the Pros and Cons evaluation dataset, and the models trained on NoReC$^{\pm}$ CNN train to have the best performance on NoReC$^{\pm}$ dev. However, fine-tuning on a specific dataset does not seem to cause better results on the same dataset.

	Lexicon description	Lexicon size	Accuracy NoReC$^{\pm}$	Accuracy Pros and Cons	CNN accuracy
	Seed words (Adj/adv)	100	63.81	71.45	-
CBOW	Non-fine-tuned embeddings	2,146	64.03	63.98	-
	FT NoReC$^{\pm}$	2,142	58.33	**73.94**	81.88
	FT NoReC$^{\pm}$ (freeze)	2,159	**66.43**	70.29	82.87
	FT Pros and Cons^{+}	2,143	65.35	73.56	92.84
	FT Pros and Cons^{+} (freeze)	2,144	57.93	67.37	92.42
Skipgram	Non-fine-tuned embeddings	2,689	53.07	62.64	-
	FT NoReC$^{\pm}$	2,600	50.87	62.18	90.53
	FT NoReC$^{\pm}$ (freeze)	2,635	50.11	61.55	91.06
	FT Pros and Cons^{+}	2,572	54.82	68.71	94.73
	FT Pros and Cons^{+} (freeze)	2,625	49.42	60.77	95.15

Table 5.8: Evaluation results of the lexicons created from the NoReC embeddings. The fine-tuned (FT) models are trained on NoReC or the Pros and Cons dataset, and (freeze) represents a model trained using gradual unfreezing.

In the previous sections, we have fine-tuned word embedding models in a CNN using only one dynamic input channel. In the following section, we will explore a different approach for fine-tuning the word embedding models in a CNN by combining information from two different input channels, one static and one dynamic channel.

5.2.4 Multi-channel dynamic word embeddings

We have implemented a multi-channel convolutional neural network for fine-tuning the word embedding models, both using NoReC$^{\pm}$ and the Pros and Cons dataset. In the multi-channel network, two sets of input channels, one dynamic and one static, are passed through the convolutional layer, and then summed before performing the pooling operation. This is the same approach that was used in Y. Kim (2014) to make the representations of antonyms less similar. In our work, we will use the dynamic channel which is updated during training, to create a fine-tuned word embedding model, which will be used for sentiment lexicon creation. First, we used the same set-up as Y. Kim (2014), which used the same word embedding model as input to both the static and dynamic channel, which in our implementation is the fastText Skipgram model of dimensionality 100. We have also used the Word2Vec CBOW model of dimensionality 600, which had the best performance on the task of synonym detection by Stadsnes (2018). Since the dimensionality of the model is large, we used it in the static channel, along with the fastText Skipgram model of dimensionality 100 in the dynamic channel. The results are presented in Table 5.9.

Dataset for CNN fine-tuning	Lexicon description	Size	Accuracy NoReC$^{\pm}$	Accuracy Pros and Cons	CNN acc.
	Seed words (Adj/adv)	100	63.81	71.45	-
	Non-fine-tuned embeddings	1,639	**70.83**	77.20	93.31
NoReC	Same embeddings (MC)	1,881	69.88	70.67	89.33
	Different embeddings (MC)	1,900	69.28	76.96	90.02
Pros and Cons	Same embeddings (MC)	1,878	55.26	68.71	96.63
	Different embeddings (MC)	1,886	69.07	**77.54**	96.42

Table 5.9: Evaluation results of the lexicons created from fine-tuned word embedding models in a multi-channel CNN. The size represents the size of the resulting lexicons, whereas CNN acc. represents the accuracies in the CNN. The lexicons created from the same model (MC) were created using the Skipgram model as input to both the static and the dynamic channel, whereas the lexicons from different models (MC) used the CBOW model of dimensionality 600 in the static channel and the Skipgram model of dimensionality 100 in the dynamic channel.

The multi-channel CNN accuracies are marginally lower than the accuracies achieved in the single-channel networks. The lexicon created from the non-fine-tuned word embedding model outperforms all of the lexicons created from models that were fine-tuned in a multi-channel CNN on the NoreC$^{\pm}$ development set. The only lexicon that achieves a slightly better accuracy on the Pros and Cons development set than the lexicon from the non-fine-tuned model, is the lexicon from the fine-tuned model trained using the Skipgram model of dimensionality 100 in the dynamic channel and the CBOW model of dimensionality 600 in the static channel. The lexicon seems to benefit from a second and different embedding model, since it outperforms the lexicons that were created from models fine-tuned in a single-channel CNN, and when using the same model in both channels in a multichannel CNN.

In the previous sections, we have fine-tuned word embedding models in a CNN using different approaches to improve the quality of the embeddings. Few of the methods that we have used to fine-tune the word embedding models have produced lexicons that achieve better results than the lexicons created from the non-fine-tuned models. We have seen cases of large updates to the weights in the embedding layer in the first few epochs, which reduced the quality of the word embeddings, and the resulting lexicons had much worse performance than lexicons created from the non-fine-tuned models. In addition to fine-tuning the word embedding models in a CNN, we have also explored a very different approach for improving the quality of the word embedding models used for sentiment lexicon creation. In the following section, we

will describe retrofitting, which is another approach for improving the quality of the word embedding models.

5.3 Retrofitting

In the previous section, we explored different methods for fine-tuning word embedding models in a CNN. By fine-tuning the models, we were able to change the vector representations for certain words. Table 5.6 shows the vector representations of the antonyms "god" *(good)* and "dårlig" *(bad)*, which were made less similar after fine-tuning. However, the lexicons created from the fine-tuned word embedding models did not achieve better results on our evaluation task than the lexicons from the non-fine-tuned models. We have therefore implemented a different approach for fine-tuning the word embedding models, called retrofitting. Retrofitting is a technique developed by Faruqui et al. (2015) which involves combining the information from a word embedding model and a semantic lexicon.

Faruqui et al. (2015) first create an undirected graph from the semantic lexicon, where all the nodes represent the words in the lexicon, and the edges represent their relations. The word embedding model is represented as a matrix, each row is the vector representation for a different word. The goal of retrofitting is to keep the similarities of the word vectors in the word embedding model, but at the same time make sure that the words have similar representations as their neighbours in the semantic lexicon. This is achieved by minimizing the euclidean distance between the vectors that have relations in the semantic lexicon over n iterations. They have developed a fast method for retrofitting, the algorithm is able to retrofit a 300-dimensional word embedding model with a vocabulary of 100,000 words in five seconds. The algorithm can be used with different kinds of word embedding models such as CBOW or Skipgram, and with a variety of semantic lexicons. In the following section, we will use the method by Faruqui et al. (2015) to try to improve the quality of the word embedding models used for sentiment lexicon expansion.

5.3.1 Retrofitting word embedding models

We will use retrofitting to combine the information from the word embedding models used for lexicon expansion, with the Norwegian Synonymy Test Set, which contains lexical relations. We used the system developed by Faruqui et al. (2015), which is available from their repository.[6] It requires the files to be in the .txt file format, so we had to make a few adjustments to the code to accept the .JSON file format, which is the file format of the semantic lexicon the Norwegian Synonymy Test Set. We have retrofit a variety of the word embedding models described in Chapter 4.

[6] https://github.com/mfaruqui/retrofitting

All of the retrofit models will be evaluated in terms of the resulting lexicons, and on the intrinsic evaluation tasks which we will describe in Section 5.4. We have retrofit the Word2Vec CBOW and fastText Skipgram word embedding models from "Aviskorpuset" *(The Norwegian Newspaper Corpus)* of dimensionality 100 and 600. Table 5.10 presents the evaluation results obtained by the resulting lexicons. We have also retrofit the word embedding models trained on the NoReC training corpus, which were also used for sentiment lexicon creation. The resulting lexicons are presented in Table 5.11.

	Lexicon description	Lexicon size	Accuracy NoReC$^{\pm}$	Accuracy Pros and Cons
	Seed words (Adj/adv)	100	63.81	71.45
CBOW	Non-retrofit embeddings (100)	2,179	50.21	56.51
CBOW	Retrofit embeddings (100)	2,653	58.99	73.70
CBOW	Non-retrofit embeddings (600)	1,709	65.57	61.87
CBOW	Retrofit embeddings (600)	2,215	60.30	71.07
Skipgram	Non-retrofit embeddings (100)	1,639	**70.83**	**77.20**
Skipgram	Retrofit embeddings (100)	2,458	53.28	76.81
Skipgram	Non-retrofit embeddings (600)	1,940	61.40	69.92
Skipgram	Retrofit embeddings (600)	2,434	65.78	71.40

Table 5.10: Evaluation results of sentiment lexicons created from the retrofit and non-retrofit word embedding models trained on "Aviskorpuset" *(The Norwegian Newspaper Corpus)*. The number in the parentheses represents the dimensionality of the word embedding model.

	Lexicon description	Lexicon size	Accuracy NoReC$^{\pm}$	Accuracy Pros and Cons
	Seed words (Adj/adv)	100	63.81	71.45
CBOW	Non-retrofit embeddings	2,146	64.03	63.98
CBOW	Retrofit embeddings	1,996	**64.47**	**76.00**
Skipgram	Non-retrofit embeddings	2,689	53.07	62.64
Skipgram	Retrofit embeddings	2,424	50.12	60.07

Table 5.11: Evaluation results of lexicons created from the retrofit NoReC embeddings.

Some of the word embedding models benefited from retrofitting, whereas other retrofit models produced lexicons with worse performance on the evaluation datasets.

There does not seem to be a clear relationship between the dimensionality or training algorithm of the word embedding model, and whether it benefits from retrofitting. Among the models trained on "Aviskorpuset" *(The Newspaper Corpus)*, one of the retrofit models of dimensionality 600 produced a better sentiment lexicon, whereas the other produced a lexicon with worse performance on one of the evaluation datasets and better performance on the other dataset. One of the lexicons from the retrofit model of dimensionality 100 had a much better performance than the lexicon from the non-retrofit model. However, the lexicon from the other retrofit model of the same dimensionality performed worse. For the NoReC word embedding models, the lexicons created from the CBOW models benefited from retrofitting, whereas retrofitting the Skipgram models produced lexicons with lower performance on the evaluation task. After retrofitting the NoReC CBOW word embedding model, the resulting lexicon achieves a slightly lower accuracy than the best performing sentiment lexicon in Section 4.4.1 in Chapter 4, which obtains an accuracy of 77.20 on the Pros and Cons development set. In the following section, we will intrinsically evaluate these models to find out how retrofitting has affected the quality of the word embedding models.

5.4 Intrinsic evaluation of word embedding models

We have only evaluated the word embedding models by creating sentiment lexicons, which were evaluated using the standard evaluation task described in Section 4.1 in Chapter 4. In this section, we will use another approach for comparing the word embedding models, and measure their performance on a standard evaluation task that is not related to sentiment analysis. By using a standard evaluation task, we can measure the quality of the word embedding models based only on the information contained in the models, and assess the effects of fine-tuning or retrofitting more directly. We will use two tasks for evaluating the word embedding models, developed for the Norwegian language by Stadsnes et al. (2018). The two methods for evaluating word embedding models are synonym detection and analogy prediction.

5.4.1 Synonym detection

In the standard task, all of the synonyms from the Norwegian Synonymy Test Set and their top five and ten neighbours, are extracted from the word embedding model. We have chosen to only extract the top five neighbours, since it is the closest to the number of neighbours that we are extracting in each iteration for sentiment lexicon creation. The neighbours from the model are compared to the synonyms from the Norwegian Synonymy Test Set. The performance of the word embedding model on the synonym detection task is measured in terms of precision and recall. The precision is a measure of how many of the synonyms that were predicted were correct. Recall is a measure of how many correct predictions were made out of all of the possible predictions. This evaluation task is suitable for lemmatized word embedding models.

5.4.2 Analogy prediction

The second method for intrinsically evaluating the word embedding models involves prediction of analogies in a dataset. The analogy test set was also created by Stadsnes et al. (2018) and is based on the Google Analogies Test set developed by Mikolov et al. (2013). Since the word embedding models used in this project are lemmatized, we only use the semantic subset of the dataset. In the evaluation task, three of the words from each entry in the dataset are presented to the word embedding model, and the model should predict the fourth word. Example 5.1 presents an example from the evaluation dataset.

(5.1) "konge dronning prins prinsesse"
king queen prince princess

The words "konge, dronning, prins" *(king, queen, prince)* are presented to the model, which should predict "prinsesse" *(princess)* as the fourth word. The accuracy is based on how many correct predictions the model is able to make. For the analogy prediction task, we will make the same restrictions on the vocabulary as Stadsnes (2018) and Mikolov et al. (2013), by restricting the vocabulary to the top 30,000 and the top 1 million most frequent words.

5.4.3 Evaluation results

All of the word embedding models that have been fine-tuned in a CNN, or used for retrofitting, have been evaluated on the tasks of synonym detection and analogy prediction. In Table 5.12, we present the intrinsic evaluation results of the fastText Skipgram and Word2Vec CBOW models trained on "Aviskorpuset" *(The Newspaper Corpus)* of dimensionalities 100 and 600. We have also evaluated all of the models created using different optimizers in a CNN, which are presented in Table 5.13. We have not included the results from the rest of the models trained using SGD as the optimizer, as well as the models trained using gradual unfreezing, and the models trained in a multi-channel CNN, because they achieved the exact same results as the non-fine-tuned model. Table 5.14 presents the evaluation results of the models created from the NoReC training corpus.

All of the retrofit models have much better performance than the non-retrofit models on the synonym detection task, which was expected since the retrofit models are created by combining information from the word embedding models with the synonyms in the Norwegian Synonymy Test Set. These models also have significantly worse performance on the analogy prediction task, which also indicate that they have been overfit to the Norwegian Synonymy Test Set, and do not generalize well to another task.

	Word embedding model	Synonym detection		Analogy prediction accuracy	
		Precision	Recall	30K	1M
CBOW	Non-fine-tuned embeddings (100)	18.13	15.90	56.30	53.20
	Retrofit embeddings (100)	54.13	48.19	01.70	02.60
	Non-fine-tuned embeddings (600)	26.26	23.03	76.40	64.00
	retrofit embeddings (600)	**78.34**	**69.74**	01.70	02.30
Skipgram	Non-fine-tuned embeddings (100)	10.29	09.02	68.10	52.10
	CNN Fine-tuned (NoReC$^{\pm}$) (100)	10.26	09.00	68.10	52.20
	CNN Fine-tuned (Pros and Cons) (100)	10.30	09.03	68.10	52.10
	Retrofit embeddings (100)	70.77	63.00	09.90	04.70
	Non-fine-tuned embeddings (600)	07.34	06.44	**79.80**	35.90
	Retrofit embeddings (600)	74.02	65.90	13.70	02.90

Table 5.12: Intrinsic evaluation of the fine-tuned and retrofit word embedding models. The number in the parentheses represent the dimensionality of the model.

Word embedding model	Synonym detection		Analogy prediction accuracy	
	Precision	Recall	30K	1M
Non-fine-tuned embeddings	10.29	09.02	68.10	53.20
Adadelta	10.26	09.00	**68.10**	52.20
Adam	10.14	08.89	67.60	51.60
RMSprop	09.97	08.74	67.80	51.70
SGD (0.01)	10.28	09.01	**68.10**	52.20
SGD (0.05)	10.29	**09.02**	**68.10**	52.10
SGD (0.1)	10.29	**09.02**	**68.10**	52.20

Table 5.13: Intrinsic evaluation of word embedding models created using different optimizers in a CNN. Since the models created using SGD as the optimizing routine achieved similar results, only a few of the models are present in the table.

It does not seem like there is a correlation between the results on the intrinsic evaluation tasks, and the evaluation accuracies achieved by the resulting lexicons. The lexicon that was created from the model using the Adam optimizer, only achieved accuracies of 50.00 on the NoReC$^{\pm}$ dev and 67.17 on the development set of the Pros

and Cons, but achieved similar results on the intrinsic evaluation as the model trained using RMSprop. However, the resulting lexicon from the RMSprop model, achieved higher accuracies on the development split of NoReC$^{\pm}$ (60.30) and on the development split of the Pros and Cons (70.24).

	Word embedding model	Synonym detection		Analogy prediction accuracy	
		Precision	Recall	30K	1M
CBOW	Non-fine-tuned (all)	06.80	03.44	**19.60**	**07.40**
	Non-fine-tuned (relevant)	02.25	04.47	11.50	04.40
	Fine-tuned (NoReC$^{\pm}$)	03.19	01.60	11.50	04.40
	Fine-tuned (NoReC$^{\pm}$) freeze	03.50	01.76	13.50	05.50
	Fine-tuned (Pros Cons)	03.24	01.63	11.60	04.50
	Fine-tuned (Pros Cons) freeze	03.24	01.63	11.50	04.40
	Retrofit model	31.67	16.33	04.70	02.00
Skipgram	None-fine-tuned (all)	07.69	03.89	17.00	06.40
	None-fine-tuned	03.50	01.76	13.70	05.10
	Fine-tuned (NoReC$^{\pm}$)	03.46	01.74	13.60	05.10
	Fine-tuned (NoReC$^{\pm}$) freeze	03.19	01.60	11.50	04.40
	Fine-tuned (Pros Cons)	03.53	01.77	13.70	05.10
	Fine-tuned (Pros Cons) freeze	03.49	01.75	13.70	05.10
	Retrofit model	**38.52**	**19.86**	04.70	01.90

Table 5.14: Intrinsic evaluation of the NoReC word embedding models. The non-fine-tuned (all) represent the models that were created using all of the words in the NoReC training corpus, whereas the non-fine-tuned (relevant) were trained using only adjectives, adverbs, verbs, and nouns. All of the fine-tuned and retrofit models are based on the original NoReC (relevant) models. The datasets represent the datasets that were used for fine-tuning, and the models marked with freeze were created using gradual unfreezing.

The models trained on the NoReC training set have a small vocabulary, of roughly 65,000 words. These models have much lower recall and precision on the synonym detection task than the majority of the other models. All of the other models are trained on "Aviskorpuset" *(The Norwegian Newspaper Corpus)*, and contain almost 1,5 million words. The lexicons created using the NoReC CBOW model outperformed the lexicons created using the NoReC Skipgram model, both after fine-tuning the models in a CNN or after retrofitting. However, for both of these intrinsic evaluation tasks, the NoReC Skipgram model has better overall performance than the NoReC CBOW model.

In this chapter, we have explored methods for improving the quality of the word embedding models used for sentiment lexicon expansion. We described methods for retrofitting, and for fine-tuning the word embedding models in a CNN. There was a small improvement in the accuracy on the NoReC$^{\pm}$ evaluation dataset by a lexicon created from a word embedding model that was fine-tuned in a CNN using gradual unfreezing. The top performing lexicon on the NoReC$^{\pm}$ dev set was created from a model that used static weights in the embedding layer for the first six epochs of training to avoid large updates to the weights, and then dynamic weights for the rest of the training. Some of the lexicons created from the word embedding models trained using the NoReC training corpus improved significantly both after fine-tuning in a CNN and after retrofitting. On the task of intrinsic evaluation of word embedding models, the non-fine-tuned and the fine-tuned models achieved similar results, even though the accuracies by the resulting lexicons were different. It seems that fine-tuning the word embedding models can have an impact on the accuracies by the resulting lexicons, however, the small changes in the vector representations do not seem to influence the results on the task of intrinsic evaluation. In the following chapter, we will analyse some of the top performing lexicons that we have created using the various strategies for lexicon creation explored in this work.

Chapter 6

Final evaluation and analysis

In the previous chapters, we have examined two main approaches for sentiment lexicon creation. We have created sentiment lexicons by automatically extracting words from labelled text, and also created sentiment lexicons from a set of seed words. Different resources have been used to expand the size of the lexicon, for example word embedding models, and resources containing lexical relations, such as WordNet and the Norwegian Synonymy Test Set.

In this chapter, we will evaluate the top performing lexicons created using the various strategies for lexicon creation. To evaluate the sentiment lexicons on the held-out test sets of NoReC$^{\pm}$ and the Pros and Cons, we will use the standard evaluation method described in Section 4.1 in Chapter 4. In what follows, we will describe the lexicons that we have chosen for final evaluation in Section 6.1. We will also analyse their top hits on the development sets of NoReC$^{\pm}$ and the Pros and Cons, and examine similarities and differences between the lexicons. In Section 6.2 we will evaluate the lexicons on the held-out test sets and analyse the results.

6.1 Final analysis of the lexicons

We have chosen to perform the final evaluation on the four top performing lexicons created using the different strategies we have explored in this work. In this section, we are going to analyse the top performing lexicons and their most frequent hits in the development sets of NoReC$^{\pm}$ and the Pros and Cons. We will analyse two lexicons created using approaches described in Chapter 4, and two lexicons from Chapter 5. The first lexicon we will analyse is the combined Potts lexicon extracted from the NoReC$^{\pm}$ and the Pros and Cons training sets. The second lexicon is based on the manually annotated set of seed words containing adjectives and adverbs, and expanded using the fastText Skipgram word embedding model of dimensionality 100. The third lexicon is the top performing lexicon among the lexicons created from models that have been fine-tuned in a CNN. The fourth lexicon is a lexicon created from a retrofit word embedding

model, which had the best performance among the retrofit models on the development sets of NoReC$^{\pm}$ and Pros and Cons. In the following paragraphs, we will remind the reader of these lexicons by providing a short description of the top performing ones.

Potts lexicon The first lexicon that we will analyse in this chapter, is the lexicon that was extracted from the training sets of NoReC$^{\pm}$ and the Pros and Cons, which had the best performance on the development datasets used for evaluation. The lexicon was created by combining the lexicons extracted from the two datasets. These lexicons were created by extracting words from the datasets that had a Potts score of more than 0.9 for either positive or negative reviews, and a frequency in the dataset of at least five. The concatenated lexicon had much better performance on the development set of Pros and Cons than on NoReC$^{\pm}$, but after removing the stopwords, the accuracy significantly improved on the development set of NoReC$^{\pm}$. We will therefore use the concatenated lexicon from NoReC$^{\pm}$ and the Pros and Cons without stopwords, which we will call the Potts lexicon, for final analysis and evaluation.

Lexicon from word embeddings The second lexicon that we have chosen for final evaluation is the lexicon created from the set of seed words containing adjectives and adverbs, and expanded by extracting six neighbours in two iterations from the Skipgram model of dimensionality 100. This model was also used to find the optimal parameters for neighbours and iterations for lexicon creation in Chapter 4. It had the best performance among the lexicons that were created based on the manually annotated seed words and expanded using a word embedding model that had not been fine-tuned or retrofit. This lexicon will be called the lexicon from embeddings.

CNN fine-tuned lexicon The third lexicon was created from a fine-tuned word embedding model, which had the best performance on the development sets of NoReC$^{\pm}$ and the Pros and Cons, among the lexicons created from the fine-tuned word embedding models. This lexicon will be called the CNN fine-tuned lexicon. It was created from the same set of seed words and word embedding model as the lexicon described in the previous paragraph, however, this lexicon was created from a word embedding model which has been fine-tuned in a CNN on the NoReC$^{\pm}$ CNN training set. The lexicon that achieved the highest accuracies on the evaluation task was a lexicon generated from a word embedding model which had been fine-tuned using gradual unfreezing. The model was trained using static weights in the embedding layer for the first six epochs, and dynamic weights for the rest of the training. The lexicon that we have chosen for final evaluation had almost identical results on the intrinsic evaluation task as the lexicon described in the previous paragraph, however, the CNN fine-tuned lexicon achieved better accuracies on the NoReC$^{\pm}$ development set.

The retrofit sentiment lexicon The fourth lexicon that we have chosen for the final evaluation is a lexicon created from a retrofit word embedding model. Similar to the

two previous lexicons, this lexicon is also based on the manually annotated set of seed words. The lexicon that had the best overall performance on the development splits of the evaluation datasets among the retrofit lexicons, was the lexicon created from the retrofit CBOW model, trained on the NoReC training corpus. We will call this lexicon the retrofit lexicon.

6.1.1 Analysis of top ten hits

In this section, we will present the top ten hits from the sentiment lexicons on the NoReC$^{\pm}$ and the Pros and Cons development sets. The number in the parentheses in the tables represents the frequencies in the dataset. Table 6.1 presents the top ten hits for the Potts lexicon, whereas Table 6.2 presents the top ten hits for the lexicon from embeddings and the CNN fine-tuned lexicon. Table 6.3 presents the top ten hits for the retrofit lexicon. The English translations are presented in Table A.3, A.4, and A.5 in Appendix A.

The Pros and Cons dev set	
Positive	Negative
komfort (15)	kostbar (8)
romslig (16)	kort (9)
behagelig (16)	vanskelig (9)
flott (17)	dyr (9)
komfortabel (17)	tung (12)
kjøreegenskap (25)	begrenset (17)
fin (33)	svak (22)
batteritid (38)	mangle (22)
ytelse (40)	dårlig (25)
bra (43)	litt (65)

The NoReC$^{\pm}$ dev set	
Positive	Negative
grei (47)	egentlig (90)
fantastisk (47)	vanskelig (93)
flott (48)	begynne (97)
familie (50)	grunn (111)
overraske (50)	dårlig (116)
bygge (59)	heller (127)
humor (62)	finne (180)
fin (79)	igjen (205)
bra (86)	litt (281)
lese (146)	film (581)

Table 6.1: The top ten hits by the Potts lexicon.

The Potts lexicon extracted from NoReC$^{\pm}$ and the Pros and Cons dataset is very different from the other lexicons, since all of the other lexicons are based on the manually annotated seed words and the Skipgram word embedding model of dimensionality 100. This is also evident from the lists of the top ten hits on the evaluation datasets. Many of the manually annotated seed words are present among the top ten hits for all of the other lexicons, which is not surprising since these seed words were used to create these lexicons. However, even though the Potts lexicon is not based on the seed words, there are six positive and four negative seed words among the top ten hits. It shows that different approaches for sentiment lexicon creation, such as manual annotation or review extraction, can lead to similar sentiment lexicons.

The Pros and Cons dev set	
Positive	Negative
elegant (11)	kostbar (8)
solid (14)	vanskelig (9)
spesiell (16)	dyr (9)
praktisk (16)	tøff (9)
flott (17)	billig (10)
svak (22)	tung (12)
fin (33)	så (12)
bra (43)	svak (22)
enkel (61)	dårlig (25)
god (271)	ikke (130)

The NoReC$^{\pm}$ dev set	
Positive	Negative
sterk (83)	morsom (86)
spesiell (91)	egentlig (90)
vakker (94)	vanskelig (93)
imponere (99)	akkurat (97)
bra (102)	vite (107)
enkel (135)	dårlig (116)
nok (223)	kanskje (149)
slik (249)	gammel (156)
også (703)	så (978)
god (989)	ikke (1,644)

Table 6.2: The top ten hits from the lexicon created from the manually annotated seed words and a word embedding model. This table represents the top ten hits for the lexicons expanded using both the fine-tuned word embedding model and the non-fine-tuned model, since the lists of the top ten hits were very similar. The top ten hits for the Pros and Cons development set were the same for the two lexicons. For the NoReC$^{\pm}$ development set, the CNN fine-tuned lexicon does not contain the word "så" *(so)*, so the 10th word for the CNN fine-tuned lexicon is the word "mulig" *(possible)*, with 81 hits.

The Pros and Cons dev set	
Positive	Negative
solid (14)	vanskelig (9)
praktisk (16)	dyr (9)
romslig (16)	skulle (12)
behagelig (16)	tung (12)
flott (17)	ville (12)
komfortabel (27)	burde (13)
fin (33)	om (16)
bra (43)	like (16)
enkel (61)	svak (22)
god (271)	dårlig (25)

The NoReC$^{\pm}$ dev set	
Positive	Negative
fin (79)	tidlig (135)
sterk (83)	mens (136)
morsom (86)	gammel (156)
vakker (94)	samme (201)
vanlig (94)	like (235)
imponere (99)	måtte (380)
bra (102)	ville (441)
faktisk (106)	man (495)
enkel (135)	skulle (557)
god (989)	om (1,335)

Table 6.3: The top ten hits from the lexicon created from the retrofit NoReC CBOW model.

The top ten hits for the NoReC$^{\pm}$ and the Pros and Cons development sets are very different, which demonstrates the difficulty of creating a sentiment lexicon that has good performance on both datasets. The frequencies of the top ten hits in the NoReC$^{\pm}$ development set are generally much higher than the frequencies in the Pros and Cons development set, which is not surprising since the development set of NoReC$^{\pm}$ contains 236,354 tokens, and the development set of the Pros and Cons dataset only contains 9,221 tokens. There are also quite a few differences in terms of word classes in the different lists of the top ten hits. The majority of positive hits are adjectives and adverbs, whereas for the negative hits, there are many adjectives and adverbs, as well as nouns and verbs.

Among the top ten hits for the Potts lexicon in Table 6.1, there are many words that would not be considered sentiment-bearing by a human, however, they may be important predictors of sentiment in these datasets. For example, the word "batteritid" *(battery life)* appears 31 times in positive summaries and only 7 times in negative summaries, whereas the word "ytelse" *(performance)* appears 32 times in positive summaries and 8 times in negative summaries in the Pros and Cons development set. Almost all of the words among the top ten hits in the Pros and Cons development set can be used in product descriptions.

In the list of the top ten hits by the Potts lexicon on the NoReC$^{\pm}$ development set, there are words that are indicative of different categories, such as products, books, and movies. Figure 3.3 in Chapter 3 presents the category distribution of the development split of NoReC$^{\pm}$, which can explain some of the words present in the list of the top ten hits in Table 6.1. The word "film" *(film)* is in the negative list, and it is the word that has the highest number of hits in the NoReC$^{\pm}$ development set. There are almost three times as many negative than positive movie reviews in the development set of NoReC$^{\pm}$. The word "lese" *(read)* in the positive list is an indicator of the literature category, which has around three times more positive than negative reviews in the development set of NoReC$^{\pm}$. Another word that is very indicative of category is the brand name "Samsung" *(Samsung)*, which is not among the top ten hits, but it has the 12th highest frequency from the positive list on the NoReC$^{\pm}$ development set. There are around 50 product reviews in the development split of NoReC$^{\pm}$, and almost all of the reviews that belong to the product category in NoReC$^{\pm}$ dev are positive.

Table 6.2 presents the top ten hits for the lexicons created from both the fine-tuned version and the non fine-tuned version of the Skipgram word embedding model of dimensionality 100. The only difference between the top ten hits for the two lexicons, is that the CNN fine-tuned lexicon does not contain the word "så" *(so)*, so the 10th word for the NoReC$^{\pm}$ development set is the word "mulig" *(possible)*, with 81 hits. The lexicons have the same performance on the development set of Pros and Cons, however, the CNN fine-tuned lexicon performs better on the NoReC$^{\pm}$ development set. In the development set of NoReC$^{\pm}$, the lexicon from word embeddings has 4,480 positive

and 4,843 negative hits, whereas the CNN fine-tuned lexicon has 4,286 positive and 3,971 negative. The CNN fine-tuned lexicon which does not contain the word "så" *(so)*, achieves a better accuracy on the development set of NoReC$^{\pm}$, and it likely that some of the increase in the accuracy can be attributed to removing this non-sentiment-bearing word from the lexicon.

We removed the stopwords from the lexicons, to find out how they affected the performance. Before removing stopwords from the lexicon from embeddings, it achieved an accuracy of 70.83 on NoReC$^{\pm}$ and 77.20 on the Pros and Cons development sets. The accuracies after removing the stopwords decreased to 64.47 on NoReC$^{\pm}$ and 69.67 on the Pros and Cons development set. For the CNN fine-tuned lexicon, after removing stopwords, the accuracy on NoReC$^{\pm}$ decreased from 71.92 to 66.44, and the accuracy on the Pros and Cons development set decreased from 76.81 to 69.47. These lexicons perform worse after removing the stopwords. Even though these words would not be considered sentiment-bearing by a human annotator, they may be important for predicting sentiment in the evaluation datasets.

One of the goals of fine-tuning the word embedding models in a CNN is to make the vector representations of antonyms less similar. The word "morsom" *(funny)* among the negative words in Table 6.2, should not be present since it is a positive word. It has been added to the lexicon because it was one of the top six neighbours of the word "dritmorsom" *(very funny)* in the word embedding model, which was among the top six neighbours of the negative seed word "dum" *(dumb)*. The word "morsom" *(funny)* had the 11th highest frequency among the negative words in the Pros and Cons development set, and the 10th highest frequency among the negative words in the NoReC$^{\pm}$ development set. Since the word "morsom" *(funny)* is among the top ten negative hits for both the lexicon from the fine-tuned model, and the lexicon from the non-fine-tuned model, the fine-tuning of the word embeddings was not successful for this word.

Another problematic word in Table 6.2 is the word "ikke" *(not)*, which is a word that would not be labelled positive or negative by a human annotator. The word "ikke" *(not)* appears 871 times in positive reviews and 773 times in negative reviews in the NoReC$^{\pm}$ development set. The frequencies in the Pros and Cons dataset are very different, it only appears 11 times in positive summaries, whereas it appears 119 times in negative summaries. Below are two examples of sentences containing the word "ikke" *(not)*, from the NoReC$^{\pm}$ dataset.

(6.1) "Serien klarer heller ikke å få noe særlig ut av sine omgivelser i de to første episodene."
The series also fails to get much out of its surroundings in the first two episodes.

(6.2) "Den øvelsen redder ikke serien fra å levere dårlig håndverk på manus- og skuespillersiden."

The exercise does not save the series from delivering poor craftsmanship to the screenplay and the acting.

These examples show how the word "ikke" *(not)* is used in negative reviews in the NoReC$^{\pm}$ dataset. They are not used to negate positive sentiment-bearing words, but rather as a part of a longer negative description. Below are two examples of summaries from the Pros and Cons dataset containing the word "ikke" *(not)*.

(6.3) "Ikke alltid like god fargegjengivelser, ustabil programvare?"
Not always as good color rendering, unstable software?

(6.4) "Bilde- og musikkvaliteten ikke så god, for liten oppløsning på skjermen"
The picture and music quality is not great, the monitor resolution is too low.

The NoReC reviews are very long, and can contain detailed descriptions to convey the sentiment of the review. On the other hand, the Pros and Cons summaries are straight to the point, and need to convey the sentiment in as few words as possible. Often, it results in summaries containing positive sentiment-bearing words, such as "god" *(good/great)* in the examples above, along with a negation to shift the sentiment of the summary. Since there is such an unequal distribution of the word "ikke" *(not)* in the positive and negative summaries in the Pros and Cons dataset, we can assume that the word "ikke" *(not)* is important for prediction of negative summaries in this dataset.

The top ten hits by the retrofit lexicon are presented in Table 6.3. Almost all of the positive words are from the manually annotated positive seed words. The word "morsom" *(funny)* is no longer present among the top ten hits after retrofitting. The list of negative hits only contains five words that would be considered negative by a human annotator, and the rest of the words are non-sentiment-bearing. Among the negative words, there are many modal verbs, such as "skulle" *(should)*, "burde" *(should)*, "ville" *(would)*, and "måtte" *(must)*, that were not present in the lexicons created from the non-retrofit word embedding models. We also removed stopwords from this lexicon, and the accuracy on the development set of NoReC$^{\pm}$ increased from 64.47 to 66.00, whereas the accuracy on the development set of the Pros and Cons decreased from 76.00 to 75.23.

We have presented and analysed the top ten hits from each of the top performing lexicons on the development sets of NoReC$^{\pm}$ and the Pros and Cons. We have shown how antonyms and words that are non-sentiment-bearing can influence the evaluation results and pose challenges to the automatic creation of sentiment lexicons. We have also demonstrated how different approaches for sentiment lexicon creation can result in similar lexicon entries. In the following section, we will evaluate the top performing lexicons on the held-out test sets of NoReC$^{\pm}$ and the Pros and Cons.

6.2 Final evaluation on the held-out test sets

The top performing lexicons have been evaluated on the held-out test sets of NoReC$^{\pm}$ and the Pros and Cons. The results are presented in Table 6.4, along with the evaluation results on the development sets.

The first lexicon in Table 6.4 is the concatenation of the Potts lexicons extracted from the training sets of NoReC and the Pros and Cons. The lexicon from word embeddings is created from the manually annotated set of seed words containing adjectives and adverbs, and expanded using the Skipgram model of dimensionality 100. The CNN fine-tuned lexicon was created from the Skipgram model of dimensionality 100 which has been fine-tuned in a CNN, by unfreezing the weights in the embedding layer after six epochs of training the network. The retrofit lexicon was created from the retrofit model trained on the NoReC training corpus using the CBOW algorithm.

Lexicon description	Size	Accuracy NoReC$^{\pm}$ dev	Accuracy NoReC$^{\pm}$ test	Accuracy Pros and Cons dev	Accuracy Pros and Cons test
Potts lexicon	1357	73.90	72.18	79.84	83.68
Lexicon from embeddings	1888	70.83	69.89	77.20	78.83
CNN fine-tuned lexicon	1888	71.92	72.64	76.81	78.64
Retrofit lexicon	1996	64.47	64.82	76.00	77.47

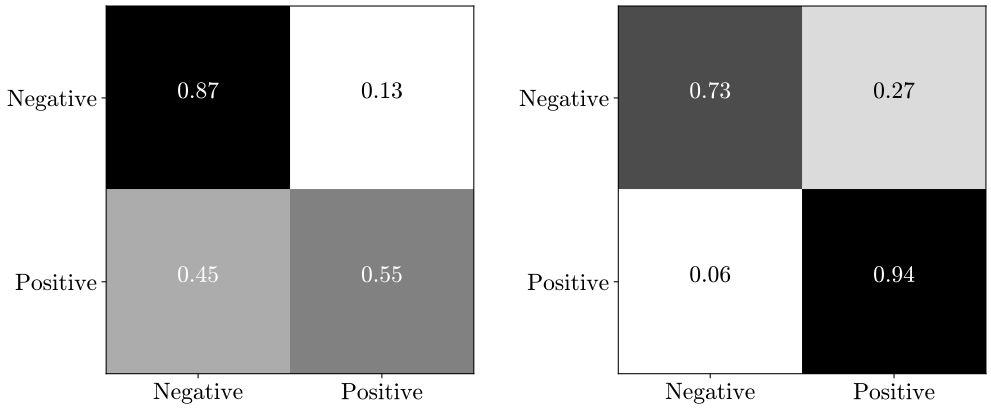
Table 6.4: Evaluation results by the top performing lexicons, on the development and test sets of NoReC$^{\pm}$ and the Pros and Cons. The size represents the size of the lexicons.

6.2.1 Potts lexicon

The Potts lexicon achieved the highest accuracy on the Pros and Cons test set, and only performs marginally worse than the top performing lexicon on the NoReC$^{\pm}$ test set. The accuracy on the Pros and Cons test set is significantly higher than any of the other accuracies obtained by the other sentiment lexicons. Figure 6.1 presents the confusion matrices on the test sets by the Potts lexicon. For the Pros and Cons evaluation dataset, the lexicon achieves good accuracies for predicting both positive and negative summaries, whereas for the NoReC$^{\pm}$ test set, the accuracy for positive predictions is quite low.

The Potts lexicon was created from the training splits of NoReC and the Pros and Cons, and evaluated on the development and test splits of the same datasets. We are using different splits of the datasets for the various tasks, but even if the splits contain different texts, they tend to be quite similar. Therefore, we cannot know if the

performance of the lexicon on the evaluation task can be attributed to overfitting, or if the lexicon will have good performance on an unseen dataset. Ideally, this lexicon should be evaluated on a third dataset, but to the best of our knowledge, NoReC and the Pros and Cons datasets are the only two datasets available for sentiment analysis in Norwegian.



(a) The NoReC$^{\pm}$ test set.

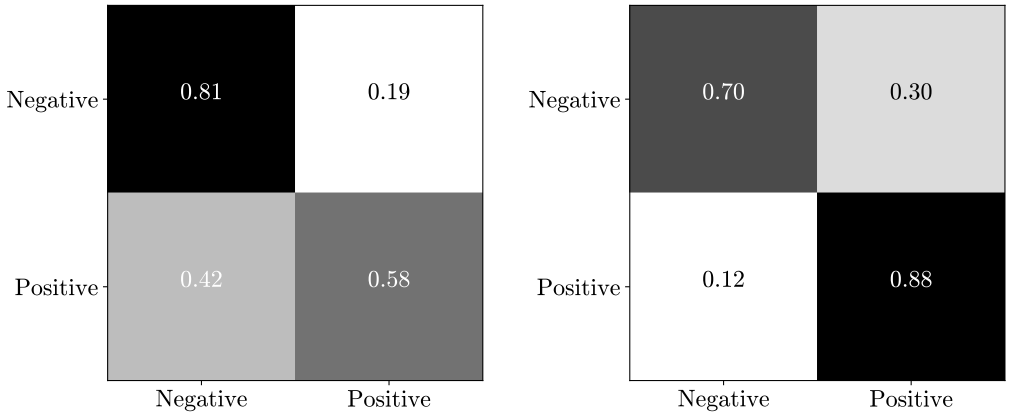
(b) The Pros and Cons test set.

Figure 6.1: Confusion matrices of class accuracies by the Potts lexicon.

In the following sections, we will analyse the lexicons that were created from the manually annotated set of seed words, and expanded using different versions of the Skipgram word embedding model of dimensionality 100. The lexicons created from the CNN fine-tuned word embedding model and the retrofit CBOW NoReC word embedding model, are both influenced by the NoReC dataset, which is the same dataset that is used for evaluation. However, the lexicon created from the manually annotated seed words and expanded using the word embedding model that has not been fine-tuned or retrofit, is not influenced by any of the evaluation datasets during the creation of the lexicon, and we can be more confident in the results.

6.2.2 Lexicon from embeddings

The sentiment lexicon created from the set of manually annotated seed words and expanded using the non-fine-tuned version of the Skipgram word embedding model of dimensionality 100, achieves slightly lower results on the NoReC$^{\pm}$ test set than on the NoReC$^{\pm}$ development set, and vice versa for the Pros ans Cons dataset. The confusion matrices of the accuracies obtained by the lexicon on the two test sets are presented in Figure 6.2. Similar to the lexicon that was evaluated in the previous paragraph, this lexicon is much more likely to make a negative prediction for reviews in the test set of NoReC$^{\pm}$, but for the Pros and Cons test set, it achieves good results for both positive and negative predictions.



(a) The NoReC$^{\pm}$ test set.

(b) The Pros and Cons test set.

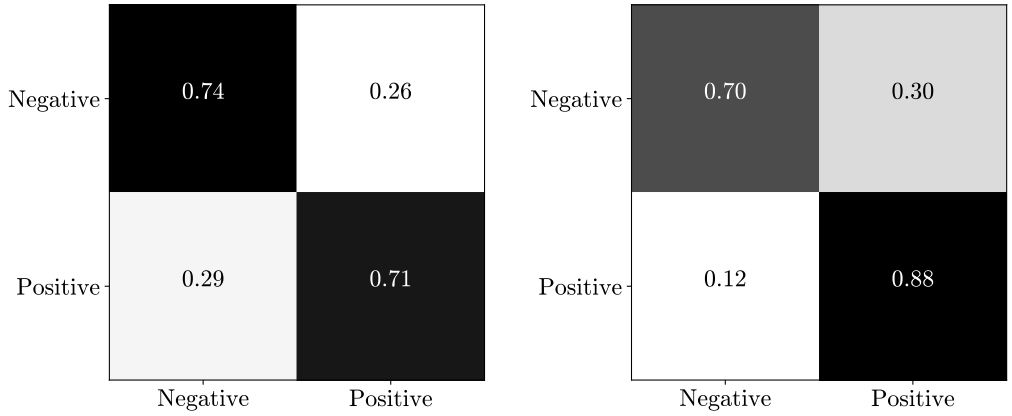
Figure 6.2: Confusion matrices of class accuracies on the test sets by the lexicon from word embeddings.

6.2.3 CNN fine-tuned lexicon

The third lexicon that we have evaluated is the lexicon created from the set of manually annotated seed words, and expanded using the Skipgram word embedding model of dimensionality 100 which has been fine-tuned in a CNN using gradual unfreezing. This lexicon is very similar to the lexicon described in the previous paragraph, however, it has a better performance on the NoReC$^{\pm}$ test set. Figure 6.3 presents the confusion matrices on the evaluation datasets. The lexicon created from the fine-tuned word embedding model has improved significantly in making correct positive predictions for the NoReC$^{\pm}$ test set. It predicts negative reviews with an accuracy of 74.00, whereas it predicts positive reviews with an accuracy of 71.00. The lexicon created from the non-fine-tuned word embedding model achieved a better accuracy on labelling negative reviews (81.00), but the accuracy was quite low for positive reviews (58.00). If the word "så" *(so)* is as frequent in the test set as it was in the development set of NoReC$^{\pm}$, this may explain why the CNN fine-tuned lexicon which does not contain this non-sentiment-bearing word, achieves a better accuracy than the lexicon from word embeddings which contains the word "så" *(so)*.

The model that was used for lexicon creation was fine-tuned in a CNN using the NoReC$^{\pm}$ CNN training dataset. Since the lexicon achieves better results on the NoReC$^{\pm}$ test set than the lexicon created from the non-fine-tuned model, we can assume that the word embedding model became more specific to the NoReC$^{\pm}$ dataset after fine-tuning. Since the results on the Pros and Cons dataset did not improve after fine-tuning, the model may have been overfit to fit the data in the NoReC$^{\pm}$ dataset and may not be able to generalize to another similar dataset. In the following section, we will analyse the

retrofit lexicon, which was created from a retrofit word embedding model trained on the NoReC training corpus.



(a) The NoReC$^{\pm}$ test set.

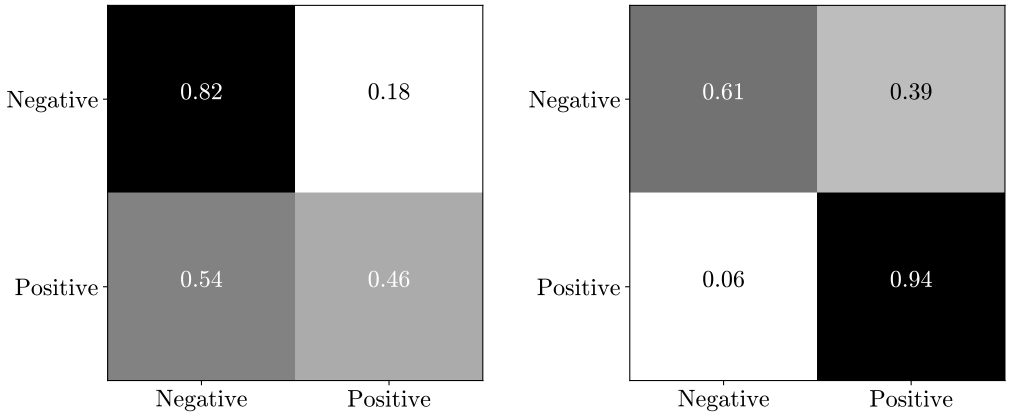
(b) The Pros and Cons dataset.

Figure 6.3: Confusion matrices of class accuracies of the CNN fine-tuned lexicon on the final test set.

6.2.4 The retrofit lexicon

The final lexicon that has been evaluated on the held-out test sets, is the lexicon created from the retrofit CBOW word embedding model trained on the NoReC training corpus. This lexicon achieves quite poor results on the NoReC$^{\pm}$ test set, which is surprising since it was created from a word embedding model trained on this dataset. The confusion matrices in Figure 6.4 present the accuracies across the positive and negative labels on the test sets of NoReC$^{\pm}$ and the Pros and Cons. On the Pros and Cons test set, the lexicon achieves very good results for predicting positive reviews, whereas it has much more error when predicting negative reviews. On the NoReC$^{\pm}$ test set, the lexicon achieves good results for predicting negative reviews, and poor results for predicting positive reviews. If the top ten hits on the test sets are similar to the top ten hits on the development sets presented in Table 6.3, we can see how some of these words influenced the evaluation results. For the NoReC$^{\pm}$ development set, the top ten negative words achieve over twice as many hits as the top ten positive words, and most of these are non-sentiment-bearing words. The word "god" *(good)*, which is the most frequent word from the lexicon in the Pros and Cons development set, achieves almost twice as many hits than all of the top ten negative words in this evaluation dataset.

In this chapter, we have analysed the top performing sentiment lexicons created using the approaches described in the previous chapters. The lexicons were evaluated on the held-out test sets of NoReC$^{\pm}$ and the Pros and Cons, to measure how well the



(a) The NoReC$^{\pm}$ dataset.

(b) The Pros and Cons dataset.

Figure 6.4: Confusion matrices of class accuracies by the retrofit lexicon on the final test set.

lexicons perform on unseen data. Most of the lexicons achieved slightly higher results on the test sets than the development sets. The top performing lexicon on the NoReC$^{\pm}$ test set was the CNN fine-tuned lexicon, which achieved an accuracy of 72.64. The top performing lexicon on the Pros and Cons test set was the Potts lexicon, which achieved an accuracy of 83.68. The Potts lexicon achieved the best overall accuracies among the sentiment lexicons evaluated in this chapter. The only lexicon that we have analysed in this chapter that was not influenced by any of the evaluation datasets during the creation of the lexicon, was the lexicon created from the manually annotated set of seed words and expanded using the non-fine-tuned Skipgram model of dimensionality 100. For all of the other lexicons, we risk that they have been overfit to the evaluation datasets, and that they will not perform well on sentiment classification on unseen data.

Chapter 7

Conclusion and future work

In this work, we have performed large-scale experiments on sentiment lexicon creation for the Norwegian language. We have explored various automatic approaches for the creation and expansion of lexicons, which resulted in many different lexicons with diverse attributes. We have used a simple approach for the evaluation of sentiment lexicons, which consisted of calculating the sum of positive and negative words from the lexicon, that are present in the review. The review is considered positive if there are more positive than negative words, and vice versa (any possible ties are resolved by random choice).

The two main datasets that we have used for the creation and evaluation of sentiment lexicons, are NoReC and the Pros and Cons dataset. The Pros and Cons dataset, which is currently being prepared as part of the SANT project, consists of roughly 7,300 review-summaries. The summaries are short sentiment-bearing sentences, or keyword-based fragments, and have an average length of 14 words. All of the summaries in the Pros and Cons dataset are taken from product reviews from NoReC. NoReC consists of over 35,000 reviews from nine different categories, such as screen, music, literature, products, and games. The reviews in NoReC are quite long, with an average length of 426 words. For the majority of the applications of NoReC in this work, we only used a small subset, containing reviews with a rating of 1, 2, and 6, which we labelled NoReC$^{\pm}$.

In this work, we have used various approaches for sentiment lexicon creation based on distributional methods. We have explored the use of:

1. Two main approaches for sentiment lexicon creation:
   (a) Extracting lexicons from labelled text.
   (b) Lexicon creation based on a set of seed words.
2. Several different sources of seed words:

(a) Manually annotated seed words.

(b) Seed words extracted from WordNet.

3. Several resources for lexicon expansion:

   (a) A variety of word embedding models:

      i. Using different frameworks, algorithms, dimensionalities, and different corpora.

   (b) The Norwegian Synonymy Test Set, which contains lexical relations.

4. Several methods for improving the word embedding models used for lexicon expansion:

   (a) Fine-tuning in a CNN using various approaches:

      i. Single- and multi-channel, gradual unfreezing, and different optimizers.

   (b) Retrofitting.

Our first approach for sentiment lexicon creation involved extracting positive and negative words from NoReC$^{\pm}$ and the Pros and Cons dataset. These lexicons were labelled the Potts lexicons, since they were created using a method based on the work by Potts (2011). The lexicon with the best overall performance on the held out test sets of NoReC$^{\pm}$ and the Pros and Cons, was the lexicon that was created by combining the Potts lexicons extracted from the two datasets and removing the stopwords. It achieved an accuracy of 83.68 on the test set of the Pros and Cons, and an accuracy of 72.18 on the test set of NoReC$^{\pm}$. However, since we have used the training split of the datasets for lexicon creation, and the development and test splits of the same datasets for evaluation, we cannot be certain if the performance can be attributed to overfitting, or if the lexicon will achieve similar results on an unseen dataset. A manual inspection of the lexicon revealed words that were domain-specific to reviews, such as the words "actionkomedie" *(action comedy)* and "film" *(film)*, but it also revealed many words that would be considered sentiment-bearing by a human annotator. The majority of the words among the top ten hits in the Pros and Cons development set in Table 6.1 in Chapter 6, are positive and negative adjectives and adverbs, which are words that belong in a sentiment lexicon.

The second main approach for lexicon creation, which was the approach that formed the basis for all of the lexicon expansion experiments, was based on a set of seed words. Since the manually annotated seed words had much better performance on the evaluation task than the WordNet seed words, it was used as the main set of seed words for lexicon expansion. Distributional models and lexical resources were used for lexicon expansion, to increase the size and the coverage of the lexicon. The main approach for lexicon expansion involved extracting words that were similar to the seed words from word embedding models. We performed extensive experiments using a fastText

Skipgram and a Word2Vec CBOW model to find the optimal configuration for the number of iterations and neighbours to extract from the models. The configuration that achieved the best results, consisted of extracting six neighbours in two iterations from the model, and was achieved by the fastText Skipgram model, which outperformed the Word2Vec CBOW model on this task. We also trained our own in-domain Word2Vec models on NoReC, experimenting with different training algorithms (CBOW and Skipgram) and dimensionalities.

We faced some of the challenges associated with automatic lexicon expansion, which involved adding non-sentiment-bearing words, or mislabelling antonyms in the lexicon. To further improve the quality of the word embeddings, and to try to overcome some of these challenges, we experimented with two main approaches for changing the vector representations in the models. We explored different approaches involving convolutional neural networks to fine-tune the word embeddings. Gradual unfreezing of the weights in the embedding layer produced models which resulted in the best sentiment lexicons, among the CNN fine-tuned models. Training the weights in the embedding layer from the first epoch resulted in too large updates to the weights, which decreased the performance of the resulting sentiment lexicons on the evaluation task. The lexicon that was created using gradual unfreezing for the first six epochs had the best performance on the NoReC$^{\pm}$ test set, with an accuracy of 72.64.

As an alternative to fine-tuning, we also experimented with the task of retrofitting, in which we used the system by Faruqui et al. (2015). We retrofit the models by combining the information in the word embedding models with the information contained in the Norwegian Synonymy Test Set. As can be expected, the performance on the intrinsic evaluation task of synonym detection was significantly better for the retrofit models than the non-retrofit models. However, retrofitting also seems to significantly decrease the performance on the other intrinsic evaluation task of analogy prediction, which indicates that the retrofit models do not generalize well to other tasks. Retrofitting had a great impact on the CBOW NoReC word embedding models, the resulting sentiment lexicons achieved much better results than the lexicons from the non-retrofit model. For both evaluation datasets, the accuracies on the development sets increased considerably after retrofitting. The accuracy on the development set of NoReC$^{\pm}$ increased from 50.51 to 58.99, and the accuracy on the Pros and Cons development set increased from 56.51 to 73.70.

In this work, we have only used a single evaluation method to evaluate the sentiment lexicons on NoReC$^{\pm}$ and the Pros and Cons dataset. The main advantage of this approach, is that it mostly manages to isolate the effects of the lexicon, and the classification relies on the words in the lexicon instead of being influenced by other features. However, a manual investigation of the Potts lexicon in Section 4.2.2 in Chapter 4, demonstrated how other features, such as the category of the review, influenced the evaluation. One of the disadvantages of this approach is that it does

not take compositional effects like negation into account when predicting the polarity of the review. We observed how negation influenced the evaluation results, especially for the Pros and Cons dataset, since many of the summaries contained a negation along with a positive sentiment-bearing word. Creating evaluation methods that account for negation could result in better accuracies, however, since we have chosen to focus on exploring a variety of automatic approaches for lexicon creation, it is outside the scope of this thesis, but a suggestion for future work.

We also observed many differences in the two evaluation datasets, for example from the lists of the top ten hits by the top performing lexicons, in Section 6.1.1 in Chapter 6, which shed light on the difficulties of creating sentiment lexicons with good performance on both of our datasets. The performance of the lexicons also tends to depend on the genre, text type, or domain of the review. Since there is not a standard task for evaluating sentiment lexicons, or a standardized tool to measure their overall quality, we can only compare the lexicons in terms of their performance on our evaluation task. Unfortunately, NoReC and the Pros and Cons dataset are the only available resources for sentiment analysis in Norwegian. We wanted to use existing datasets for evaluating the sentiment lexicons, instead of for example collecting reviews from the web, in order for our results to be easily reproduced by others.

The accuracies obtained by the sentiment lexicons using our simple evaluation method, were significantly lower than the accuracies achieved using a convolutional neural network for sentiment classification. On the Pros and Cons development set, the highest accuracy obtained by a sentiment lexicon was 87.52, which was achieved by the Potts lexicon that was extracted from the Pros and Cons training set. The top CNN configuration achieved an accuracy of 97.50, and was trained on the Pros and Cons CNN train and tested on the development set of the Pros and Cons. On the NoReC$^{\pm}$ development set, the highest lexicon accuracy was 77.85, which was obtained by the Potts lexicon extracted from the NoReC$^{\pm}$ training set. The highest CNN accuracy was 93.42, which was achieved using NoReC$^{\pm}$ train for training the network, and NoReC$^{\pm}$ dev for testing. The highest CNN accuracies for both datasets were achieved while testing the newly created datasplits in Chapter 3. The CNN's produced significantly better accuracies on our evaluation datasets, than the sentiment lexicons created in this work. However, sentiment lexicons have many advantages, such as the transparency of the lexicons, and how they can easily be altered or adapted to new domains. We can also perform a manual inspection of the lexicons, and understand why they make certain predictions, whereas the predictions by the CNN and other neural models, are much more difficult to interpret.

To the best of our knowledge, this is the first project involving binary CNN classification of extremely positive and negative reviews in NoReC. We have tuned the hyperparameters of the network using a random search, and established a baseline hyperparameter configuration, which produced the best results for our task. In this

work, we provide strong results for sentiment analysis using a convolutional neural network. Even though the focus of this work has been on the creation of sentiment lexicons, we have gained some insight into CNN's, by fine-tuning the word embedding models used for lexicon creation. A suggestion for further research is on the use of sentiment lexicons in convolutional neural networks. Sentiment lexicons can contain valuable information about the sentiment of words, and can be used as a feature in neural classifiers. Shin, Lee, and Choi (2016) describe several different approaches for using lexicon embeddings in a CNN.

The goal of this work was to provide sentiment lexicons for the Norwegian language, and to shed some light on the strengths and weaknesses of different automatic approaches for sentiment lexicon creation. Even though we have explored various approaches, there are still both manual and automatic approaches for lexicon creation that we have not examined in this work. In the previous attempts at creating Norwegian sentiment lexicons (Hammer, Bai, et al., 2014; Hammer et al., 2015; Hammer, Yazidi, et al., 2014), the machine translated version of the sentiment lexicon AFINN (Nielsen, 2011) had the best overall performance. However, since they have not published their translation of the lexicon, we do not know how it compares to the lexicons created in this work. English Sentiment lexicons can be translated to Norwegian, either manually or using a translation software. Other approaches for creating sentiment lexicons which we have not explored, include manual annotation of lexicons, by one individual, or as a crowd-sourcing project by several people. Positive and negative words can be extracted from a corpus or labelled text, using approaches other than calculating Potts scores (Potts, 2011) or PMI (Turney, 2002), for example using methods involving correlation. Different sets of seed words could result in fundamentally different sentiment lexicons, even if using exactly the same approaches for lexicon expansion as we have used in this work. All of these methods for lexicon creation and expansion are suggestions for further research on the topic of sentiment lexicons for Norwegian.

Appendices

Appendix A

English translations

Positive seed words	Negative seed words
fine, nice, pleasant, pretty, secure, good, gorgeous, genious, funny, impressive, honest, wonderful, beautiful, fantastic, positive, excellent, outstanding, strong, clever, true, cozy, cool, fun, easy, loving, amiable, warm, pleasing, intelligent, brilliant, successful, polite, clean, lucky, sweet, friendly, generous, perfect, super, glorious, great, bright, stunning, elegant, practical, modern, solid	bad, ugly, gruesome, mean, hideous, macabre, negative, dirty, angry, sick, violent, mad, wrong, foolish, bulky, slow, fake, weird, strict, crazy, difficult, lame, afraid, woefully, painful, silly, disgusting, dumb, jealous, weak, awful, painful, dark, boring, peculiar, tired, dangerous, sleepy, merkelig, heavy, lazy, old, chaotic, hard, terrible, tired, dangerous, vulgar, expensive, cold

Table A.1: The English translation of the positive and negative seed words.

Word	Cosine similarity	Word	Cosine similarity
good	83.50	good	81.87
very good	79.86	very good	76.73
passable	77.84	great	75.72
great	77.30	excel	75.55
very good	77.26	very good	75.14
excellent	77.12	excellent	75.03
great	76.70	fine	74.81
fine	76.24	passable	74.45
bad	76.18	great	73.98
excel	75.61	very good	72.94

Table A.2: The English translation of the top ten neighbours of the word "god" *(good)* in a word embedding model before and after fine-tuning in a CNN.

The Pros and Cons dev set		The NoReC$^{\pm}$ dev set	
Positive	Negative	Positive	Negative
comfort (15)	costly (8)	decent (47)	really (90)
spacious (16)	short (9)	fantastic (47)	difficult (93)
agreeable (16)	difficult (9)	flott-great (48)	begin (97)
great (17)	expensive (9)	family (50)	reason (111)
comfortable (17)	heavy (12)	surprise (50)	bad (116)
driving property (25)	limited (17)	build (59)	rather (127)
fine (33)	weak (22)	humor (62)	find (180)
battery life (38)	lacking (22)	fin-fine (79)	again (205)
performance (40)	bad (25)	nice (86)	little (281)
nice (43)	little (65)	read (146)	film (581)

Table A.3: The English translation of the top ten hits by the Potts lexicon.

The Pros and Cons dev set	
Positive	Negative
elegant (11)	costly (8)
solid (14)	difficult (9)
special (16)	expensive (9)
practical (16)	tough (9)
great (17)	cheap (10)
weak (22)	heavy (12)
fin-fine (33)	so (12)
nice (43)	weak (22)
simple (61)	bad (25)
good (271)	not (130)

The NoReC$^{\pm}$ dev set	
Positive	Negative
strong (83)	funny (86)
special (91)	really (90)
beautiful (94)	difficult (93)
impress (99)	exactly (97)
nice (102)	know (107)
simple (135)	bad (116)
enough (223)	maybe (149)
such (249)	old (156)
also (703)	so (978)
good (989)	not (1644)

Table A.4: The English translation of the top ten hits from the lexicon from embeddings.

The Pros and Cons dev set	
Positive	Negative
solid (14)	difficult (9)
practical (16)	expensive (9)
spacious (16)	should (12)
agreeable (16)	heavy (12)
great(17)	would (12)
comfortable (27)	should (13)
fin-fine (33)	whether (16)
nice(43)	to like (16)
simple (61)	weak (22)
good (271)	bad (25)

The NoReC$^{\pm}$ dev set	
Positive	Negative
fine (79)	early (135)
strong (83)	while (136)
funny (86)	old (156)
beautiful (94)	same (201)
usual (94)	like (235)
impress (99)	had to (380)
nice (102)	would (441)
actually (106)	one (495)
simple (135)	should (557)
good (989)	whether (1335)

Table A.5: The English translation of the top hits from the retrofit lexicon.

Bibliography

Baccianella, S., Esuli, A., & Sebastiani, F. (2010). SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. In *Proceedings of Language Resources and Evaluation*. Valetta, Malta.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, *3*.

Bergem, E. (2018). *Document-level sentiment analysis for Norwegian* (Master's thesis, Universitetet i Oslo, Oslo, Norway).

Bergstra, J. & Bengio, Y. (2012). Random Search for Hyper-parameter Optimization. *Journal of Machine Learning Research*, *13*.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, *5*.

Braasch, A. & Pedersen, B. (2010). Encoding Attitude and Connotation in Wordnets. In *Proceedings of the XIV Euralex International Congress*. Leeuwarden, Netherlands.

Bradley, M. & Lang, P. (1999). *Affective Norms for English Words (ANEW): Instruction manual and affective ratings* (tech. rep. No. C-1). The Center for Research in Psychophysiology, University of Florida. Gainesville, Florida.

Ceron, A., Curini, L., Iacus, S., & Porro, G. (2014). Every tweet counts? How sentiment analysis of social media can improve our knowledge of citizens' political preferences with an application to Italy and France. *New Media & Society*, *16*(2).

Charles, W. G. & Miller, G. A. (1989). Contexts of antonymous adjectives. *Applied Psycholinguistics*, *10*(3).

Chollet, F. et al. (2015). Keras. https://keras.io. GitHub.

Church, K. & Hanks, P. (1989). Word Association Norms, Mutual Information, and Lexicography. In *Proceedings of the 27th Annual Conference of the Association for Computational Linguistics*. Vancouver, Canada.

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*. Helsinki, Finland.

Esuli, A. & Sebastiani, F. (2006). SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining. In *Proceedings of the 5th Conference on Language Resources and Evaluation*. Genova, Switzerland.

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*. Gothenburg, Sweden.

Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., & Smith, N. A. (2015). Retrofitting Word Vectors to Semantic Lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado, USA.

Fellbaum, C. (1998). WordNet: An electronic lexical database.

Firth, J. R. (1957). A Synopsis of Linguistic Theory 1930-55. *Studies in Linguistic Analysis*, *1952-59*.

Gatti, L., Guerini, M., & Turchi, M. (2016). SentiWords: Deriving a High Precision and High Coverage Lexicon for Sentiment Analysis. *The IEEE Transactions on Affective Computing*, *7*.

Glorot, Xavier and Bengio, Yoshua. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. Sardinia, Italy.

Hamilton, W., Clark, K., Leskovec, J., & Jurafsky, D. (2016). Inducing Domain-Specific Sentiment Lexicons from Unlabeled Corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, USA.

Hammer, H., Bai, A., Yazidi, A., & Engelstad, P. E. (2014). Building sentiment Lexicons applying graph theory on information from three Norwegian thesauruses. In *Proceedings of Norsk Informatikkonferanse*. Fredrikstad, Norway.

Hammer, H., Yazidi, A., Bai, A., & Engelstad, P. E. (2014). Constructing sentiment lexicons in Norwegian from a large text corpus. In *Proceedings of the 17th IEEE International Conference on Computational science and Engineering*. Chengdu, China.

Hammer, H., Yazidi, A., Bai, A., & Engelstad, P. E. (2015). Building Domain Specific Sentiment Lexicons Combining Information from Many Sentiment Lexicons and a Domain Specific Corpus. In *Proceedings of the 5th IFIP TC 5 International Conference on Computer Science and Its Applications*. Saida, Algeria.

Harris, Z. (1954). Distributional Structure. *WORD*, *10*.

Hatzivassiloglou, V. & McKeown, K. R. (1995). A quantitative evaluation of linguistic tests for the automatic prediction of semantic markedness. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*. Cambridge, Massachusetts, USA.

Hatzivassiloglou, V. & McKeown, K. R. (1997). Predicting the Semantic Orientation of Adjectives. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*. Madrid, Spain.

Jacovi, A., Sar Shalom, O., & Goldberg, Y. (2018). Understanding Convolutional Neural Networks for Text Classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium.

Jurafsky, D. & Martin, J. H. (2018). *Speech and Language Processing* (3rd ed.). draft.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Baltimore, Maryland, USA.

Kim, S.-M. & Hovy, E. (2004). Determining the Sentiment of Opinions. In *Proceedings of the 20th International Conference on Computational Linguistics*. Geneva, Switzerland.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Quatar.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11).

McCloskey, M. & Cohen, N. (1989). Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In G. H. Bower (Ed.), (Vol. 24). Psychology of Learning and Motivation- Advances in Research and Theory. San Diego, California, USA: Academic Press.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *Computing Research Repository*, *abs/1301.3781*.

Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the Association for Computing Machinery*, *38*(11).

Nielsen, F. (2011). A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. In *Proceedings of the European Semantic Web Conference 2011 workshop on "Making Sense ofMmicroposts": Big things come in small packages, CEUR workshop proceedings*. Heraklion, Greece.

Pang, B. & Lee, L. (2008). Opinion Mining and Sentiment Analysis. *Foundation and Trends in Information Retrieval*, *2*.

Pedersen, B., Nimb, S., Asmussen, J., Sørensen, N., Trap-Jensen, L., & Lorentzen, H. (2009). DanNet: The challenge of compiling a WordNet for Danish by reusing a monolingual dictionary. *Language Resources and Evaluation*, *43*.

Potts, C. (2011). On the negativity of negation. In *Proceedings of the 20th conference on Semantics and Linguistic Theory*. Vancouver, Canada.

Ratcliff, R. (1990). Connectionist Models of Recognition Memory: Constraints Imposed by Learning and Forgetting Functions. *Psychological Review*, *97*.

Rubenstein, H. & Goodenough, J. B. (1965). Contextual Correlates of Synonymy. *Communications of the Association for Computing Machinery*, *8*.

Sand, H. (2016). *Extending the Norwegian WordNet using Word Embeddings* (Master's thesis, Universitetet i Oslo, Oslo, Norway).

Sand, H., Velldal, E., & Øvrelid, L. (2017). Wordnet extension via word embeddings: Experiments on the Norwegian Wordnet. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*. Gothenburg, Sweden.

Shin, B., Lee, T., & Choi, J. (2016). Lexicon Integrated CNN Models with Attention for Sentiment Analysis. *Computing Research Repository*, *abs/1610.06272*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*.

Stadsnes, C. (2018). *Evaluating Semantic Vectors for Norwegian* (Master's thesis, Universitetet i Oslo, Oslo, Norway).

Stadsnes, C., Øverlid, L., & Velldal, E. (2018). Evaluating Semantic Vectors for Norwegian. In *Proceedings of Norsk Informatikkkonferanse*. Svalbard, Norway.

Stone, P. J. & Hunt, E. B. A. (1963). A computer Approach to Content Analysis: Studies Using the General Inquirer System. Detroit, Michigan, USA.

Stone, P., Dunphy, D., Smith, M., & Ogilvie, D. (1968). The General Inquirer: A Computer Approach to Content Analysis. *American Journal of Sociology*, *73*(5).

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-based Methods for Sentiment Analysis. *Computational Linguistics*, *37*(2).

Turney, P. (2002). Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA.

Turney, P. & Littman, M. (2003). Measuring Praise and Criticism: Inference of Semantic Orientation from Association. *Computing Research Repository*, *cs.CL/0309034*.

Velldal, E., Øverlid, L., Bergem, E., Stadsnes, C., Touileb, S., & Jorgensen, F. (2017). NoReC: The Norwegian Review Corpus. In *Proceedings of the Language Resources and Evaluation Conference*. Miyazaki, Japan.

Wang, L.-Y. & Xia, R. (2017). Sentiment Lexicon Construction with Representation Learning Based on Hierarchical Sentiment Supervision. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark.

Warriner, A. B., Kuperman, V., & Brysbaert, M. (2013). Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior Research Methods*, *45*.

Wilson, T., Wiebe, J., & Hoffmann, P. (2005). Recognizing Contextual Polarity in Phrase-level Sentiment Analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Vancouver, Canada.

Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. *Computing Research Repository*, *abs/1212.5701*.

Zhang, Y. & Wallace, B. C. (2017). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 8th International Conference on Natural Language Processing*. Taipei, Taiwan.