Interpreting Non-Sentential Utterances in Dialogue

Experiments with non-sentential utterances and their processing with neural language models

Adrian Nadau Semb

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

Department of Informatics
The Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Spring 2022

Interpreting Non-Sentential Utterances in Dialogue

Experiments with non-sentential utterances and their processing with neural language models

Adrian Nadau Semb

© 2022 Adrian Nadau Semb

Interpreting Non-Sentential Utterances in Dialogue

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

Printed: Reprosentralen, University of Oslo

Abstract

In natural dialogue between humans, *non-sentential utterances* (NSUs), also known as fragments, frequently occur. These utterances do not have the form of a complete and grammatically correct sentence, but still convey the meaning of one, when inferred from the previous dialogue context. Due to the frequency of these utterances in human conversation, dialogue systems such as Question Answering (QA) systems, needs to be able to infer the meaning of NSUs in order to converse with humans in a natural and coherent manner.

This thesis presents several experiments that try to analyze neural language models' ability to infer NSUs in dialogue, which is an important step towards creating complete dialogue systems. Language models based on the transformer architecture (Vaswani et al. 2017), were used to classify NSUs based on the taxonomy and corpus presented by (R. R. Fernández 2006). Previous classification methods used automatically extracted features from annotated data, while the language models only have access to raw text. However, the language models was not able to achieve the same performance as previous methods, though the ones used are relatively small, containing "only" 110M-117M parameters, compared to other existing language models that can contain several Billion parameters.

This thesis then analyzes the NSU and the *contextual information* i.e. the previous utterances of the NSU, using feature attribution. It shows that, in general, the tokens of the NSU get attributed higher scores than the tokens in the contextual information.

Finally, the thesis presents a multiple-choice QA benchmark dataset, with conversations containing one or multiple NSUs. The questions posed in the context of a conversation have multiple associated choices, where only one is correct. This dataset was tested on multiple language models trained on UnifiedQA-v2 (Khashabi, Kordi, and Hajishirzi 2022) format. There were huge improvements with the increase in numbers of parameters, and the largest model containing 11B parameters was able to achieve an accuracy of only 6.3% lower than that of humans.

Acknowledgements

Firstly, I want to sincerely thank my supervisor Dr. Pierre Lison for helping and guiding me through this thesis.

I also want to thank my mother and sister for participating in a user study performed in this thesis. They both answered a total of 119 samples containing multiple-choice questions posed in the context to conversations.

Finally, I want to thank my partner for spell checking and correcting a lot of grammatical mistakes for several parts of this thesis, but most importantly supporting me every day.

Contents

1 Introduction **1**

1.1 Motivation . . . 2
1.2 Results . . . 3
1.3 Thesis outline . . . 4

2 Background and previous work **7**

2.1 Non-Sentential Utterances . . . 7
2.1.1 Taxonomy of NSUs . . . 8
2.1.2 The NSU classes . . . 8
2.1.3 Corpus Studies of Fragments . . . 11
2.1.4 NSU Corpus . . . 12
2.1.5 Conversational datasets . . . 13
2.1.6 Resolution of NSUs . . . 14
2.1.7 Classification of NSUs and Feature Engineering . . . 15
2.2 Language models and interpretability . . . 16
2.2.1 Transformer architecture . . . 16
2.2.2 Transformer Language Models . . . 18
2.2.3 Transfer Learning and Zero-Shot Task Generalization . . . 19
2.2.4 Feature Attribution . . . 20
2.2.5 Probing tasks . . . 22
2.3 Question Answering . . . 23
2.3.1 Conversational Machine Comprehension . . . 24
2.3.2 Open-domain Question Answering . . . 25
2.3.3 Prompting and prompt-based learning . . . 25
2.4 Summary . . . 27

3 Classification and Feature Attribution of NSUs **29**

3.1 Data . . . 30
3.1.1 Current dataset . . . 31
3.2 Previous Classification Methods . . . 32

3.2.1 Features . . . 32
3.2.2 Results from previous methods . . . 35
3.3 Experimental setup for classification . . . 35
3.3.1 Model architecture . . . 36
3.3.2 Input Text . . . 37
3.3.3 Training . . . 38
3.3.4 Metrics . . . 38
3.4 Results: Classification of NSU . . . 40
3.4.1 Baseline . . . 40
3.4.2 Experimentation results . . . 41
3.4.3 10-fold cross-validation . . . 42
3.4.4 Comparison with previous results . . . 44
3.5 Feature Attribution . . . 46
3.5.1 Experimental setup . . . 47
3.5.2 Feature attribution scores over test data . . . 49
3.5.3 Results and discussion . . . 50
3.5.4 Feature attribution for synthetic dialogues . . . 54
3.6 Summary . . . 58

4 Question Answering Benchmark **61**
4.1 NSU dialogue dataset . . . 61
4.1.1 Multiple-Choice Question Answering . . . 63
4.1.2 Dialogue example . . . 64
4.1.3 Text communication . . . 65
4.2 Experimental setup . . . 66
4.2.1 Random baseline . . . 67
4.2.2 User study . . . 67
4.3 Results and Discussion . . . 69
4.3.1 Comparing different model sizes . . . 70
4.4 Summary . . . 73

5 Conclusion **76**
5.1 Future work . . . 78

A Classification and Feature Attribution **81**

List of Figures

3.1 Distribution of the number of tokens in the contextual information and the NSU, when using BERT tokenizer. 51
3.2 Distributions of the attribution scores of the contextual information and NSU for both BERT and DialoGPT. 52
3.3 Histogram of the summation of contextual information, and NSU, for each sample in test data. 53
3.4 Histogram of the average of contextual information, and NSU, for each sample in test data. 54

4.1 Shows the how accuracies improve with the increase in model size 71

List of Tables

2.1 Overview of the NSU taxonomy 12
2.2 Occurrences and fraction of the classes in the corpus (class name in parenthesis). 13

3.1 Per class occurrences for different distances, between the antecedent and the NSU, in the corpus. 30
3.2 Features used for classification by R. Fernández, Ginzburg, and Lappin (2007) 33
3.3 Per class performance of the previous results of R. Fernández, Ginzburg, and Lappin (2007) and Dragone and Lison (2016) respectively, using 10-fold cross-validation. 36
3.4 Results of the baseline model (skip-gram). 41
3.5 Shows the weighted average scores on the test set for different classifiers for BERT. 42
3.6 Shows the weighted average scores on the test set for different classifiers for DialoGPT. 42
3.7 Per class average results when performing 10-fold cross-validation on BERT with a dense classifier. 44
3.8 Per class average results when performing 10-fold cross-validation on DialoGPT with a dense classifier. 45
3.9 The weighted average scores on 10-fold cross validation on NSU data. 45
3.10 Shows the weighted average scores on the test set for the classification task; all models use a dense classifier. 48
3.11 Similar dialogues with small changes to fall in another class. 55

4.1 Distribution of NSU instances for each class in the NSU dialogue dataset. 62
4.2 Shows the number of available choices and the associated number of samples. 64

4.3 Shows the number of parameters and accuracy for each model on the benchmark NSU QA dataset, along with the accuracy of the user study. 70

Chapter 1

Introduction

In natural dialogue between humans, utterances in form of incomplete sentences often take place. These utterances, called *non-sentential utterances* (NSUs), also known as *elliptical utterances* or *fragments*, do not have the form of complete sentences. However, NSUs can be inferred from the previous dialogue context, which makes them convey full sentential meaning, often in form of a question, a proposition, or a request. Hence, when NSUs are used in conversation, the meaning of such utterances are tightly coupled with its conversational context. Below we have listed some dialogue transcripts from the British National Corpus (BNC) (Burnard 2000), where the last utterance (in bold), is a NSU:

(1) A: Are you right or left handed?
B: **Right handed.** [BNC: G3Y 96 – 97]

(2) A: I wonder if that would be worth getting?
B: **Probably not.** [BNC: H61 81 – 82]

(3) A: It's Ruth's birthday.
B: **When?** [BNC: KBW 13116 – 13117]

Above (3), we see that "When?" is an utterance to the statement "It's Ruth's birthday.". By itself it does not convey much meaning, and it would be impossible to decipher without taking into account the previous statement. The previous utterance(s), called *contextual information*, needs to be used in order to understand that person B is asking for further information about the previous statement. The utterance "When?" could be paraphrased to the complete sentence "When is Ruth's birthday?", which is called to *resolve* the NSU. For neural language models the resolved sentence is usually easier to interpret than a NSU. Humans, however, use NSUs all the time without even thinking about them and we do have to resolve

NSUs in order to comprehend such utterances.

We can consider example (4), which is a multi-turn dialogue containing multiple NSUs throughout the conversation[1].

(4) a. A: When will our party be held?
b. B: Next Wednesday.
c. A: Have you sent out invitations by our party?
d. B: Yes, I have.
e. A: When?
f. B: Ten days ago.
g. A: Would Dr. Cole like to attend the party?
h. B: Yes, he will.
i. A: That's fine.

In example (4), we can see that there are in total 6 NSUs including 4b, 4d, 4e, 4f, 4h, and 4i. Most NSUs can be inferred using their preceding utterance, such as 4h, but other would need also consider utterances several turns away from the NSU. For understanding what the question "When?" (4e) is asking for, we would need not only to consider its previous turn, but also the turn 4c.

1.1 Motivation

Recently, we have seen significant improvements in natural language processing (NLP) and natural language understanding (NLU) tasks, with the introduction of pre-trained language models that creates contextualized word embeddings. Language models such as ELMo (Peters et al. 2018), BERT (Devlin et al. 2019), GPT-2 (Radford, Wu, et al. 2019), and others, have replaced feature-based models and creates word embeddings based on a words position and context in a text.

Especially the models based on the Transformer architecture (Vaswani et al. 2017), e.g. BERT and GPT, have become the state-of-the-art in many tasks. These models are pre-trained on a large corpora of raw text, like Wikipedia and BookCorpus (Y. Zhu et al. 2015), using unsupervised learning and has scored high on language understanding benchmark such as GLUE (Wang, Singh, et al. 2018) and SuperGLUE (Wang, Pruksachatkun, et al. 2020).

[1] example is from the DailyDialog corpus (Y. Li et al. 2017)

Our **main research question** is; to what degree are neural language models able to interpret NSUs in dialogues? As this is an open question we present several methods to try and answer this. One such method is the classification of NSU.

This gives rise to another question, which is; is the NSU or the contextual information most important for classifying NSUs in dialogue? We will be using different language models for different tasks including BERT, DialoGPT (Zhang et al. 2020), and T5 (Raffel et al. 2019) from Khashabi, Kordi, and Hajishirzi (2022).

Lastly, I think it is important to highlight the use of terms like *understanding*, or *comprehending*. As Bender and Koller (2020) points out, there is a tendency to use imprecise language when talking about language models. These terms are what they call "gross overclaims" when they are used to describe understanding and comprehension in a human-analogous way. Whether language models have the capability to "understand" meaning of natural language is still an ongoing debate (Bender and Koller 2020; Merrill et al. 2021; Michael 2020; Potts 2020), but it will not be discussed further. The term *interpret*, when used in the context of language models, will not be in terms of a language models' "understanding" of the meaning behind NSUs, but rather to which extent they are able to solve a task at hand given NSUs in dialogue.

1.2 Results

This thesis presents several methods for testing and evaluating the abilities of transformer based language models to infer NSUs in dialogue. The results from the 3 tasks are summarized below.

Classification

We use BERT and DialoGPT for to classify NSUs, using only the raw text from dialogues. To be able to classify NSUs in dialogue, some linguistic properties needs to be extracted from the both the NSU and the contextual information. The empirical results from the classification is lower than the results achieved by previous methods, however, the classification task presented is more difficult than previously done. Both language models are able perform reasonably well, hence, we can expect the contextualized word embeddings to contain some encoded information allowing the NSUs to be classified.

Feature Attribution

Extending the work previous task, we calculate feature attribution on

language models trained on the classifying NSUs. We show that the individual tokens of the NSUs generally gets higher attribution scores than the individual tokens of the contextual information. However, the contextual information as a whole gets slightly higher attribution scores than the NSU.

Question Answering

We present a multiple-choice Question Answering benchmark dataset, which contains conversations, question posed in context to these, and accompanied by multiple choices. This dataset will be available for further research and experimentation. We test 5 different T5 language models, trained on UnifiedQA-v2, on this dataset in a zero-shot setting, and empirically show a significant increase in performance as the number of parameters of the model increases.

1.3 Thesis outline

Chapter 2

This chapter contains background information of concept, resources, and methods needed in order to get a good understanding for this project. We will give an overview of NSUs and present their taxonomy, as well as previous methods that has been used for classifying and resolving these. We explain the transformer architecture and some of the language models that are using this, and detail some methods used for interpretability of such models. Finally, we will introduce several problem and methods in Question Answering.

Chapter 3

This chapter presents the task of classifying NSUs, and apply the transformer based language models, BERT and DialoGPT, on this task. It provides empirical results and compares them to a baseline using non-contextualized word embeddings, and to previous classification methods. Then we use feature attribution on our models trained on the classification task, to try to compare the NSU versus the rest of the dialogue, but will also view some attribution scores of synthetic dialogues.

Chapter 4

This chapter presents a multiple-choice QA benchmarking dataset. The dataset contains conversations with questions posed in context of these, with multiple

associated choices. We test T5 (Raffel et al. 2019) language models, trained on the UnifiedQA format (Khashabi, Kordi, and Hajishirzi 2022), on this benchmark dataset. There are 5 models in total with different number of parameters varying from 60M to 11B.

Chapter 5

This chapter contains the summarization of the outcomes in this thesis, while describing some directions in which this work can be extended.

Chapter 2

Background and previous work

In this chapter we will introduce the taxonomy of non-sentential utterances that we will be using forward, and different approaches that has been done for classifying and resolving these. We will continue by explaining some of the current language model architectures as well as exploring some of the methods for interpretability of these models. Finally, we will introduce the Question Answering problem with different tasks and datasets within this research area. Though it's assumed that the reader has some understanding in fields such as machine learning and natural language processing (NLP), we will explain the background assumptions required for this project. Since we touch upon several topics within NLP we will only explain some parts in detail and give an overview of others. The purpose is to establish a sufficient level of background knowledge for the coming chapters as well as understanding previous related work in the field.

2.1 Non-Sentential Utterances

As NSUs do not have the form of a full sentence its semantic meaning must be inferred from the surrounding context. NSUs do, however, convey the meaning of a complete sentence, usually in the form of a question or proposition. In dialogue the most important contextual factor for understanding the meaning of a NSU is the *antecedent* of the NSU, i.e. the contextual utterance used for the resolution of the fragment. Usually the antecedent is adjacent to the NSU, meaning that it is the immediate preceding utterance in the dialogue history. However, this is not always the case, especially for NSUs like *short answers*, the antecedent can appear several turns away from the NSU. This makes it difficult for a language model to identify which of the utterances in history that is the actual antecedent of a NSU.

2.1.1 Taxonomy of NSUs

Much work relating to NSUs and their taxonomies has been done by Fernandez and Ginzburg (2002) and Schlangen (2003), but earlier work of taxonomies does exist. Carberry (1990) categorizes NSUs based on the speaker's plan and intention. Another who also relies heavily on intentions for the classification of NSUs is Barton (1990) who distinguishes the NSUs based on the inference needed for their resolution. In Asher (2003), the author presents a dynamic semantic framework called Segmented Discourse Representation Theory (SDRT) where both discourse coherence and interpretation is explored in a logical approach. Schlangen shapes his taxonomy based on SDRT to develop the resolution of NSUs. R. R. Fernández (2006) has written a more detailed comparison of the classes where the author also analyzes these taxonomies more closely.

The taxonomy of NSUs we will be using forward is provided by Fernandez and Ginzburg (2002), and further refined by R. R. Fernández (2006), it contains a total of 15 classes. There are some fragments that fall outside these classes like greetings and closings e.g. "Hello" and "Bye". The 15 classes can be further divided into 5 families as viewed in table 2.1.

2.1.2 The NSU classes

Below are the 15 classes, containing a somewhat informal definition with examples[1]. A formal definition of each class with more details is written by R. R. Fernández (2006).

Plain Acknowledgement: utterances that signal that a previous utterance was understood or accepted, like "yeah", "mhm", "ok".

A: We should get off and interview Anna.
B: **Oh yes.** [BNC: KP4 4079 – 4080]

A: Cos you've already saved it.
B: **Right.** [BNC: G4K 206 – 207]

Repeated Acknowledgement: responses that repeat a part of the antecedent and are used as acknowledgement.

A: She was questioning.

[1]examples are (mostly) from the British National Corpus (BNC), and are classified by R. R. Fernández.

B: **Questioning yes.** [BNC: JYM 282 – 283]

Clarification Ellipsis (CE): utterances indicate some sort of understanding problem, and the desire for clarification of the antecedent.

A: Where was this?
B: Oh what we call Waterhall.
A: **Waterhall?**
B: Yeah, where you come down that
hill from the boy's grave. [BNC: HDH 46 – 49]

A: What about in days gone by?
B: **What?**
A: What about in days gone by? [BNC: HDH 350 – 352]

Direct sluice: these utterances do not ask for clarification, like the CE *wh*-phrases, they ask for further information that was explicitly or implicitly uttered in the antecedent.

A: Well you made him sound really, really boring.
B: **Why?** [JSN 194 – 195]

A: Who did you interview?
B: Benjamin.
A: **When?**
B: Last night. [BNC: KE0 138 – 141]

Check question: short queries that request explicit feedback from the addressee. The requested feedback is usually either if the addressee has understood a previous utterance or agrees with it.

A: So <pause> I'm allowed to record you.
Okay?
B: Yes. [BNC: KSR 5 – 7]

Short Answer: answers or responses to *wh*-questions.

A: How long would you camp up there?
B: **For about a week.** [BNC: H5G 39 – 40]

Plain Affirmative Answer: utterances, in the form of *yes*-words, that are used to give an affirmative answer to a polar question.

A: Have you got a blank on there that's not written on?
B: **Yes.** [BNC: J8D 94 – 95]

Repeated Affirmative Answer: these utterances give an affirmative answer to a polar question, but they contain a response that repeats or reformulates a part of the query.

A: Did you shout very loud?
B: **Very loud**, yes. [BNC: JJW 571 – 572]

Propositional Modifier: modal adverbs that act as a response to a contextual proposition from either an assertion or polar question, which it modifies.

A: I wonder if that would be worth getting?
B: **Probably not.** [BNC: H61 81 – 82]

A: We could hear it from outside.
B: Oh you could hear it?
A: **Occasionally yeah.** [BNC: J8D 13 – 15]

Plain Rejection: utterances that are used to give an answer to a polar questions using rejections like *no*.

A: Do you want some wine?
B: **No.**

Helpful Rejection: utterances that are negative answers to polar questions or assertions, and the rejection is accompanied with a contrasting alternative. They are used to correct some piece of the antecedent with an alternative.

A: How much do you think?
B: Three hundred pounds.
C: **More.**
B: A thousand pounds.
A: **More.** [BNC: G4X 44 – 48]

A: I thought he said fifty.
B: **Oh no, fifteen.** [BNC: J9A 376 – 377]

Factual Modifier: utterances that add information or modify some contextual available entity. They act as a response to a contextually presupposed fact, a factual adjectives, and can be thought of as an extension or continuation of the dialogue.

A: There's your keys.
B: **Oh great!** [BNC: KSR 137 – 138]

A: He goes here there and everywhere.
B: **Marvelous!** [BNC: KSR 175 – 176]

Bare Modifier Phrase: utterances that behave like adjuncts, modifying a contextual utterance, hence add/modify information of antecedent.

A: ... they got men and women in the same dormitory!
B: **With the same showers!** [BNC: KST 992 – 996]

Conjunct: extend previous utterances by conjunction.

A: Alistair erm he's, he's made himself coordinator.
B: **And section engineer.** [BNC: H48 141 – 142]

Filler: utterances that fill a previous unfinished utterance.

A: [...] would include satellites like erm
B: **Northallerton.** [BNC: H5D 78 – 79]

2.1.3 Corpus Studies of Fragments

Earlier corpus studies of fragments found that NSUs make up the following rates: R. R. Fernández (2006) found 9%, Fernandez and Ginzburg (2002) found 11.15% and Schlangen and Lascarides (2003) found that 10.2% of the studied corpus were fragments. More recently, Su et al. (2019) found that in 2,000 Chinese multi-turn conversations, 70% of them had some degree of co-reference and/or omission. This study was done on 2,000 Chinese multi-turn conversations. Chinese is a pro-drop language i.e. a language where certain classes of pronouns may be omitted when they can be inferred grammatically or pragmatically. Hence, pro-drop languages, like Chinese or Japanese, will contain more omissions than English. Though co-reference does not imply a NSU, omission predominantly does. This gives us a sense of the importance of NSUs in our natural language. R. R. Fernández (2006) also discusses where antecedents usually appear in the dialogue compared to the NSU and generally it is the immediately preceding utterance. Nevertheless, there are significant differences between the NSU classes and also in dialogue (between two parties) and multilogue i.e. dialogue between multiple parties. Larger distance between the antecedent and the NSU almost only appear in multilogue for Short Answer.

Family	Class
Acknowledgements	• Plain Acknowledgement • Repeated Acknowledgement
Questions	• Clarification Ellipsis • Direct Sluice • Check Question
Answers	• Short Answer • Plain Affirmative Answer • Repeated Affirmative Answer • Propositional Modifier • Plain Rejection • Helpful Rejection
Extensions	• Factual Modifier • Bare Modifier Phrase • Conjunct
Completions	• Filler

Table 2.1: Overview of the NSU taxonomy

2.1.4 NSU Corpus

Little annotated data for NSUs exists, but one corpus study was done by Fernandez and Ginzburg (2002) and later refined by R. R. Fernández (2006) and R. Fernández, Ginzburg, and Lappin (2007), where they found NSUs in conversations and labeled them. This study is composed of an annotated dataset of 1283 NSU instances from the BNC corpus. The conversations are both between two people (dialogue) and between more than two people (multilogue). The sub-corpus of BNC, which was examined, contains a total of 14,315 sentences. Annotation of the dataset was done by 3 annotators independently, and a reliability test was performed afterwards. It yielded a *kappa score* of 76% which the authors calls "reasonably good results"; we will refer to R. R. Fernández (2006) for details about this score. The reliability test was carried out by two non-expert annotators. A total of 50 instances were selected at random from the

Class	Occurrences	Fraction
Plain Acknowledgement (Ack)	599	0.467
Short Answer (ShortAns)	188	0.147
Plain Affirmative Answer (AffAns)	105	0.082
Clarification Ellipsis (CE)	92	0.072
Repeated Acknowledgement (RepAck)	86	0.067
Plain Rejection (Reject)	49	0.038
Factual Modifier (FactMod)	27	0.021
Repeated Affirmative Answer (RepAffAns)	26	0.020
Helpful Rejection (HelpReject)	24	0.019
Check Question (CheckQu)	22	0.017
Filler	18	0.014
Bare Modifier Phrase (BareModPh)	15	0.012
Propositional Modifier (PropMod)	11	0.009
Direct Sluice (Sluice)	11	0.009
Conjunct (ConjFrag)	10	0.008
Total	1283	

Table 2.2: Occurrences and fraction of the classes in the corpus (class name in parenthesis).

corpus, containing a minimum of 2 instances of each class. These non-expert annotators were then asked to label these instances. In table 2.2 we can see the distribution of the different NSU classes, done through a corpus study of a portion of BNC.

2.1.5 Conversational datasets

Despite the lack of annotated data for NSUs, some large datasets of multi-turn dialogues have been constructed. We have the British National Corpus (BNC) (Burnard 2000), from which the NSU Corpus is made up. BNC contains 100 million words from most written, about 90%, and spoken dialogues (10%) in English with a wide variety of different genres. The OpenSubtitles corpus by Tiedemann (2009) and Tiedemann (2012) and was further refined by Lison and Tiedemann (2016), and contains movie and TV subtitles with the latter corpus

containing a collection of 2.6 billion sentences from 60 different languages. The Twitter Dialogue Corpus (Ritter, Cherry, and B. Dolan 2010; Ritter, Cherry, and W. B. Dolan 2011; Danescu-Niculescu-Mizil, Gamon, and Dumais 2011) contains roughly 1.3 million conversations from Twitter, but the conversations are chat-like and rather noisy. Finally, DailyDialog (Y. Li et al. 2017) contains dialogues from everyday chit-chats, and attempts to reflect the daily communication between humans.

2.1.6 Resolution of NSUs

The task to rewrite a NSU into a complete sentence based on the surrounding context is called the *resolution* of a NSU (Schlangen 2005; R. R. Fernández 2006). This is according to R. R. Fernández (2006) "one the most important issues in the analysis of NSUs". Accordingly, most work in machine learning regarding NSUs and fragments are their resolution.

Consider the following examples:

(5) A: What do you want to call it?
B: **Just career guidance.** [BNC: G4X 154 – 155]

(6) A: Well you made him sound really, really boring.
B: **Why?** [BNC: JSN 194 – 195]

In example 5 the NSU can be resolved to:

Just careers guidance. → I just want to call it career guidance.

In example 6 the NSU can be resolved to:

Why? → Why did I make him sound really boring?

We can clearly see above that NSUs by themselves does not give a lot of information, but we can understand the context with only the fully resolved sentence.

Several methods have been presented to rewrite fragments to full sentences. Ginzburg and Sag (2000) present a grammatical framework, based on Head Driven Phrase Structure Grammar (HPSG), that models a feature structure from semantic, syntactic and contextual information. They try to combine the contribution of the NSU with the contextual information by grammatical construction to find the resolution of a NSU.

Schlangen (2005) tries to tackle this problem by aligning a NSU with its antecedent, in a multi-party dialogue. This is done using machine learning to identify fragments, and then find their antecedent. Another approach is to start by

classifying the NSU, as done by R. Fernández, Ginzburg, and Lappin (2007). They try to constrain a resolution based on the class of the NSU. In their work, they present a simple machine learning approach to classify the NSUs, and continue by finding their resolution. Dragone and Lison (2016) extend this work with detection by active learning techniques, to address the data scarcity of NSUs, as well as using additional features when classifying NSUs. Though there are many who tries to find the resolution of NSUs (Schlangen 2005; R. R. Fernández 2006; Kumar and Joshi 2016; Debnath, Sengupta, and Wabgaonkar 2018; Su et al. 2019), Jonathan (2012, Chapter 7) argues against a number of existing approaches to rewrite NSUs into in terms of fully-fledged sentences, where authors have tried to create a grammatical theory of NSUs.

With the introduction of (large pre-trained) language models with contextualized word embeddings we could argue that the resolution of NSUs might be less relevant. This is because these models create contextualized word embeddings that might be able to contextualize the NSUs based on their antecedent and the surrounding context. In addition to this, humans generally learn early on to use and understand NSUs without rewriting a NSU into a grammatically correct sentence before processing and understanding it. We understand it based on the previous dialogue. As we get more high-quality multi-turn dialogue datasets and the architecture of language models improves, we might see a similar pattern where language models do not need to rewrite a NSU in order to infer the meaning. In Chapter 4 we will learn how current language models perform, when inferring the meaning of a NSU based on a question posed in the context of a conversation.

2.1.7 Classification of NSUs and Feature Engineering

Classification of NSUs is a task that has seen little previous work. We will be focusing classification of NSU in conversations according to the NSU corpus explained in section 2.1.4. This is a classification task where the a model, that can be everything from a simple machine learning decision tree to a complex pre-trained language model, is supposed to classify the NSU, given the previous utterances in the conversation.

The conversation can either be in form of extracted features of the dialogue, as done by R. Fernández, Ginzburg, and Lappin (2007) and Dragone and Lison (2016), or in the form of raw text, as we will see in Chapter 3. One major assumption here is that only the preceding utterances, from the NSU, until the *antecedent*, are needed to classify these NSUs. However, for non-expert humans, it might be

helpful to get additional dialogue context to classify it i.e. either utterances before the antecedent, or after the NSU.

R. Fernández, Ginzburg, and Lappin (2007) presents classification of this taxonomy of NSUs using four machine learning algorithms. Dragone and Lison (2016) extends the previous work by R. Fernández, Ginzburg, and Lappin (2007) by adding additional features and performing *active learning* i.e. the process of letting machine learning algorithms label data by learning from a few samples of labeled data, hence increasing the amount of scarce data (Settles 2009).

For the classification task, these algorithms needs features to be preprocessed to perform better, as it was not a viable option at the time to use raw text. Hence, they are using *feature engineering* i.e. the process of using domain knowledge to select the most appropriate data (features), for a given model, for the current task (Zheng 2018, Chapter 1). We will go more in detail in Chapter 3, about the assumptions of what data is used and explain the used features.

2.2 Language models and interpretability

We will in the following section briefly view some language models used in later experiments, and continue by introducing some methods for interpretability of these such models. From earlier pre-trained word representations that create non-contextualized embeddings, such as CBOW and skip-gram (Mikolov et al. 2013), we have seen huge improvements to NLP tasks with the introduction of contextualized word embeddings, such as ELMo (Peters et al. 2018) and transformers Vaswani et al. (2017). With the increased complexity of large neural network models, there is an inflation of interest to understand the decisions and reasoning behind the produced outputs of these models (Baehrens et al. 2010; Kokhlikyan et al. 2020).

2.2.1 Transformer architecture

The transformer architecture from Vaswani et al. (2017) has sparked major improvements in NLP tasks over the last years (T. B. Brown et al. 2020). As such, we will give an overview of how the transformer architecture works[2]. The transformer consists of an **encoder** and a **decoder**, where the encoder maps a sequence of input, represented as $x_1, ..., x_n$, to a continuous representation $\mathbf{z} = (z_1, ..., z_n)$, and the decoder uses this representation to produce a sequence

[2]For an in depth explanation see the original paper (Vaswani et al. 2017), as well as guides such as The Annotated Transformer by HarvardNLP

of output $y_1, ..., y_n$. For each produced symbol, the model will consume the produced symbol when generating the next, and is hence auto-regressive.

Both the encoder and decoder, of this specific implementation, consists of a stack of $N = 6$ identical layers, but the layers in the encoder and decoder are different. The number of stacked layers can vary depending on the size of the model, and larger models can contain both additional layers and increased embedding size e.g. GPT-3 has 96 layers with embedding size of 12888 (T. B. Brown et al. 2020). There are three main components in these layers; a *multi-head attention* layer, a layer normalization, and a feed-forward neural network (FNN). Each multi-head attention layer, and FNN is follow by a layer normalization, which takes as input a residual connection. Hence, if a the previous layer l of the layer normalization $norm$ takes as input a and produces the output $l(a) = b$, the layer normalization produces $norm(a + b)$. The size of the produced outputs are always the same at $d_{model} = 512$. Keeping the same dimension throughout the connection facilitates the residual connections. When continuing explaining what the encoder and decoder contains we will omit mentioning the layer normalization, since the output of each layer is sent through this.

We will start the an **encoder layer**. It consists of a multi-head attention layer, and this is sent into a FNN layer. A **decoder layer** is a bit different. It contains first a masked multi-head attention layer, which is the same as a multi-head attention layer, but some of the inputs are masked. Then it contains a multi-head attention layer, but this layer takes as input the first layer, and the output of the encoder. Finally, we have a FNN layer.

Attention

At the core of this architecture is the self-attention mechanism. Vaswani et al. (2017) are calling their particular attention for *Scaled Dot-Product Attention*, and can be thought of a mapping of a query, key and value to an output. The following is the formula for calculating this:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

Where d_k is the total dimension of the queries and keys. This formula is what allows for self-attention when an input x is sent in as Q, K, and V.

Finally, we will explain multi-head attention, which is h number of scaled dot-product attentions in which the queries, keys, and values first goes through a

through (part of) a linear layer. The goal of multi-head attention is to allow the different "heads" to capture, or attend, to different representations of subspaces. So for we get $head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$, where W^Q, W^K, and W^V, are the linear layers for Q, K, and V respectively. And we have:

$$\text{MultiHead}(Q, K, V) = \text{Concatenate}(head_1, ..., head_h)W^O \tag{2.2}$$

Where W^O is the a linear output layer. In their model they use $h = 8$ number of heads.

2.2.2 Transformer Language Models

Two models that have become immensely popular and that are offsprings of the transformer architecture (explained in 2.2.1) are GPT (Radford, Narasimhan, et al. 2018) and BERT (Devlin et al. 2019). GPT uses only the *decoder* in the transformer model, with a few modifications. It is trained on a diverse set of corpus using generative pre-training. The main goal in the pre-training phase is to predict the current token u_i, given the previous tokens $u_{i-k}, ..., u_{i-1}$, in a unsupervised corpus of tokens $U = u_1, ..., u_n$. The model can afterwards be fine-tuned on downstream tasks using supervised tasks. However, such models are usually used on text generation tasks, but we will see in section 2.2.4 that a multitude of problems can be expressed as such.

Another model that has derived from the transformer architecture is BERT. Unlike GPT, that is based on the encoder from the transformers model, BERT is based on the *encoder*. It is also trained differently, as Devlin et al. (2019) argue that when an entire sentence is available one should not mask everything after the current position, but instead use the entire sentence and not lose any information when solving NLP tasks, hence becoming fully bidirectional. As BERT is fully bidirectional, it does not make sense to predict the next token, as done by GPT.

Hence, one of the tasks BERT is pre-trained on is rather to predict masked tokens in a sentences (specifically 15% of the input tokens are masked at random). It can then use the surrounding unmasked tokens to predict the masked tokens.

The second task of the pre-training phase, is next sentence prediction (NSP). BERT is given two sentences, A and B, during training and the objective is to tell wether B is a plausible sentence that follows A. The sentence B is 50% of the times the actual sentence that follows A, and 50% a random sentence in the corpus.

Lastly, we have models that follows more close to the original transformer model,

that uses both an *encoder* and a *decoder*, such as T5 (Raffel et al. 2019). For sequence to sequence tasks the format of encoder-decoder seems to work great (Sutskever, Vinyals, and Le 2014; Kalchbrenner, Grefenstette, and Blunsom 2014). T5 model is treating every problem as a text-to-text, i.e. taking text as input and new generated text is produced as output. Hence, for the different tasks, the input text will contain the problem to solve, for example a document to be summarized, and the output text will contain the answer, in this example the summarized text. For the model to understand the current task, a prefix is added to the original input text and then the model consumes this in entirety and is trained to produce an associated output text.

2.2.3 Transfer Learning and Zero-Shot Task Generalization

We will briefly explain some concepts when talking about learning tasks of language models. We start with *transfer learning*, which is the ability to transfer learning from some task to another (Zhuang et al. 2019). The main objective of transfer learning is to leverage the training a models on some domain so the knowledge can be used on different, but related tasks.

There are several benefits of this, but in NLP, one major benefit when pre-training large language models is that they require much less data and computing power, and can still achieve great performance (Conneau, Kiela, et al. 2017). One application of transfer learning is when used in data-scarce settings. Here, we can differentiate between three emerging tasks; *zero-shot learning*, where at test time there has been no task specific demonstration during training, *one-shot learning*, where there is only one task specific demonstration that the model trains on, and *few-shot learning*, where there are a few (where "a few" can vary a bit in number) demonstrations the model can train on before testing (T. B. Brown et al. 2020; Xian et al. 2017; Ye, Lin, and Ren 2021).

Zero-shot generalization is similar to *zero-shot learning*, but Sanh et al. (2021) explains it rather that something that can be achieved when a model is trained on a subset of tasks and are afterwards able to attain reasonable generalization on a diverse set of task without having seen a demonstration for these.

We have recently seen language models, such as GPT-3 (T. B. Brown et al. 2020), T5 (Raffel et al. 2019) and T0 (Sanh et al. 2021), that are able to achieve reasonable zero-shot generalization to new problems. These models have in common that they are all of type text-to-text. One benefit of treating every text processing problem as such, is the ability to use the same model, training procedure, objective, as well as decoding process when training on wide variety

of NLP tasks (Raffel et al. 2019). Then, when applying the model to a new task, the task can be created as a text-to-text problem, and the model does not have to be changed. We will come to this later in section 2.3.3.

2.2.4 Feature Attribution

To try and interpret large models several methods has been proposed and *feature attribution* (*feature importance* and *variable importance* are other names for this task) is one such method. This is the task of attributing the outputs predictions of a (neural network) model to the input features that generated it (Sundararajan, Taly, and Yan 2017). In image classification networks, such methods can find the pixels that are most responsible for the classification of a certain image (Simonyan, Vedaldi, and Zisserman 2013). On the other hand, for a sentiment analysis task in NLP, such as movie review, we can get a score for each input token that tells us wether they counted as positive or negative. Hence, if a language model classified a review as negative, we could then find which words that was most important for this classification. We will be using feature importance for a classification task in chapter 3.

There exist also two other attribution techniques called *layer attribution* and *neuron attribution*. With layer attribution we can evaluate the contribution of each neuron to the output for a specific layer in the model and neuron attribution analogous to this, but instead evaluates a particular neuron (Sebastian Bach et al. 2015).

A formal definition of feature attribution from Sundararajan, Taly, and Yan (2017) are written as:

> [...] suppose we have a function $F : \mathbb{R}^n \rightarrow [0,1]$ that represents a deep network, and an input $x = x_1, ..., x_n \in \mathbb{R}^n$. An attribution of the prediction at input x is relative to a baseline input x' is a vector $A_F(x, x') = (a_1, ..., a_n) \in \mathbb{R}^n$ where a_i is the contribution of x_i to the prediction $F(x)$.

We can see that for finding the feature attribution of an input x, we need to find the contribution of x relative to a the contribution of baseline input x'.

There exists several algorithms for feature attribution, such as *Saliency* (Simonyan, Vedaldi, and Zisserman 2013), *Integrated Gradients* (Sundararajan, Taly, and Yan 2017) and *DeepLift* (Shrikumar, Greenside, and Kundaje 2017), to mention a few. These are all gradient based approaches, where they calculate the backward gradient of the output with respect to the input features of a model.

Perturbation based approaches, on the other hand, are all based on a simple approach where one add small changes to an input neuron and watch the effect on output neurons. This can be done through permutation, occlusion or feature ablation (Wei, Zhenzhou, and Song 2015; Zeiler and Fergus 2013; Breiman 2001). Captum (Kokhlikyan et al. 2020) is a PyTorch (Paszke et al. 2019) library for interpretability of models that contains generic implementations of several attribution algorithms and methods, both gradient and perturbation-based, included the ones mentioned above. We will be using this library in later experiments.

Integrated Gradients

As Integrated Gradients is used in later experiments, we will briefly explain how this works. This attribution method wants to satisfy two axioms, sensitivity and implementation invariance. When every input x and baseline x' that only differs by one feature produces different predictions, then that feature should be given a non-zero attribution. An attribution method can achieve *sensitivity* when this condition is satisfied. For the latter we start by explaining *functionally equivalent* which is when two networks that might have very different implementations have the same output for all (distinct) inputs. When an attribution method does not give different attributions for networks that are functionally equivalent the attribution method is *implementation invariance*. Integrated Gradients satisfy these two axiomatic properties; sensitivity and implementation invariance.

Integrated Gradients, as most other feature attribution algorithms, requires a baseline input x' which is usually set to a zero embedding vector for text models, which we can think of as absence of feature. We want to calculate the gradient for each dimension (feature) i for both the input x and the baseline x', and is defined as follows:

$$\text{IntegratedGrad}_i(x) ::= (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha\,(x - x'))}{\partial x_i}\,\partial\alpha \tag{2.3}$$

For each feature i, $\frac{\partial F(x)}{\partial x_i}$ is the gradient of $F(x)$ along this dimension. $x_i - x_i'$ is the difference between the input and the baseline. The second part of the integral, accumulates all local gradients and hence attributes a score of how much is either added or subtracted to the models overall output. This part needs to be approximated, which can be done with Riemann sums (Sundararajan, Taly, and Yan 2017). The accumulation of gradients will not suffer from *saturation* i.e. when gradients of input features have small magnitude, but the network (as a whole) relies (heavily) on these features, then these gradients will not reflect the actual

feature importance (Sundararajan, Taly, and Yan 2016).

Deep neural networks that uses sigmoid, ReLU and pooling operation, will have a continuous function $F : \mathbb{R}^n \rightarrow \mathbb{R}$ everywhere, and the partial derivative of F will satisfy *Lebesgue integrability condition* (A. B. Brown 1936; Glorot and Bengio 2010). For such models the integrated gradients can be calculated at each feature i, hence for all features of the model.

Integrated Gradients is a good methods to develop an intuition for how the models works for specific examples, but it is hard to provide global feature importance across an entire dataset. Nor is it capable of explaining interactions between features.

2.2.5 Probing tasks

Another method for with which we can try to interpret the capabilities of language model is with *probing tasks*. Though not showed in any of the later experiments, we did some experimentation with a probing task for NSUs. Hence, we will shortly introduce what a probing task is, and introduce this probing task in section 5.1.

A probing task is usually a simple task, which in NLP can be something like parts-of-speech, to test the language model on something it has not been trained on Conneau, Kruszewski, et al. (2018). It is important that the task is something the language model has not seen during pre-training, as the goal is to assess the models capability on that task.

To predict some linguistic properties of the *probing task*, a simple supervised model, like a linear layer or multilayer perceptron (MLP) is needed. This supervised model, called *probe*, is trained on the probing task based on word (or sentence) embeddings from a pre-trained language model (like BERT) (Rogers, Kovaleva, and Rumshisky 2020; Hewitt and Liang 2019). If a probe is able to perform on the probing task, it is an indication that the pre-trained language model has encoded some linguistic properties in the embeddings that allow the simple model to extract these and predict correctly.

Probing tasks can also be used in different layers of a language model, to see where such linguistic features are encoded in the model. Hence, the goal of probing tasks is mainly to try and see if a model has understood some concepts of language, and also in which layers of the model these concepts might have been learned (Alain and Bengio 2018; Jawahar, Sagot, and Seddah 2019).

Some examples of probing tasks for reference are Conneau, Kruszewski, et al. (2018) who introduces 10 different probing tasks to test different linguistic features of language models, and *structural probes* for finding syntax in work representations (Hewitt and Manning 2019) in such models.

There are, however, different shortcomings when using probing tasks, especially in the way they are interpreted (Belinkov 2021; Ravichander, Belinkov, and Hovy 2021). One might conclude that when a probe achieves a high score on a probing task, the language model has "understood" the linguistic property of the probing task i.e. the embedding has some encoded linguistic structure, that can be extracted by the probe (Jawahar, Sagot, and Seddah 2019). However, both Belinkov (2021) and Hewitt and Liang (2019) are discussing the difficulty of interpreting the results of probes and try to find methods that give more confidence in such results.

2.3 Question Answering

The goal of Question Answering (QA) systems is to automatically answer questions posed by humans in natural language. Ever since the 1960s, the task of creating a system that is able to provide a precise answer to a user's question has been a goal (Green et al. 1961).

As a QA system needs to, not only retrieve information, but also to answer the question in a natural language, QA is considered a more complicated task than information retrieval (IR) (Abdi, Idris, and Ahmad 2016; Cao et al. 2010). In traditional information retrieval a system can return an entire document with relevant information from a request, but in QA the system would need to find the specific text, in the document, that is considered relevant (Kolomiyets and Moens 2011).

In general, the main goal of QA systems is to provide a flexible and easy way for humans to interact with computers, where they can provide direct answers from a query. However, a lot of work still remains (especially in multi-turn dialogue). There exist several problems within QA and we will outline some of them below while giving a brief explanation for the different tasks. This will help us both distinguish the tasks from each other, and understand their similarities.

A lot of work has been done in both Open-Domain Question Answering (F. Zhu

et al. 2021), and Conversational Machine Comprehension (CMC) (Gupta, Rawat, and Yu 2020), and recently we have seen prompting emerging as an input format for models (T. B. Brown et al. 2020; Stephen Bach et al. 2022).

2.3.1 Conversational Machine Comprehension

Conversational Machine Comprehension (CMC) is a challenge where a model is given an open-domain text in natural language, and engages in a multi-turn conversation to answer questions posed about this text (Gupta, Rawat, and Yu 2020). The questions posed in the conversation are often not paraphrased and will have multiple co-referenced questions i.e. a question can ask for further information about the previous question. This is very similar to a challenge called Machine Reading Comprehension (MRC), in which a model is also supposed to answer questions to an open-domain text, but unlike CMC the dialogue does not have multiple co-referenced questions.

CMC can be formally defined as: given a context of information, called passage P, and a current question Q_i asking about P, the model should predict the current answer A_i based on previously posed questions $\{Q_{i-1}, ..., Q_{i-k}\}$, and previous answers $\{A_{i-1}, ..., A_{i-k}\}$, if any (Gupta, Rawat, and Yu 2020). The answer A_i can be in the form of a text span, s_i, e_i, where s_i is beginning and e_i end of the current answer i in P, as done in the QuAC dataset (Choi et al. 2018). However, A_i can also be in the form of free text, as allowed by the CoQA dataset (Reddy, D. Chen, and Manning 2019).

A question can ask for further information about a previous question or answer, and be in the form of a NSU. This is exemplified by the two questions below, from Reddy, D. Chen, and Manning (2019)[3]:

Q12: Who went missing?
Q13: Where?

The large-scale datasets mentioned above, QuAC and CoQA, are both improving the advancement in CMC. A note to keep in mind is that CoQA and QuAC might not contain much diversity in NSU classes, as they only contain question/answer pairs.

[3]The questions are from Reddy, D. Chen, and Manning (2019) with conversation id 3h8dhmccw9bthwa0epswnh4as77kdu

2.3.2 Open-domain Question Answering

Open-Domain Question Answering (OpenQA) is a challenge where a system is supposed to answer questions posed by a human, in a natural language, based on access to unstructured documents, such as Wikipedia (F. Zhu et al. 2021). Unlike MRC and CMC, OpenQA does not have a specified context from which to answer the posed question. Hence, usually the system first has to search for a relevant document, which becomes an information seeking task (Harabagiu, Maiorano, and Pasca 2003).

This document can either be in the form of a local document, or through a document available on the Internet. The latter becomes a task similar to how humans try to find answers. We can consider solving CMC a step in the right direction for a fully functional OpenQA system. Large datasets, such as WIKIQA (Yang, Yih, and Meek 2015), SQuAD (Rajpurkar et al. 2016), and others (J. Li et al. 2020; Hermann et al. 2015), have helped the advancement of this process.

QA tends to be given in various formats, UnifiedQA (Khashabi, Min, et al. 2020; Khashabi, Kordi, and Hajishirzi 2022) tries to create a unified high quality QA format for multiple variant tasks. These models are based on text-to-text language models, and are pre-trained on 8 and 20 datasets, for v1 and v2 respectively. One such format is multiple-choice QA, where a question is posed to a text, and the language model is supposed to select an answer, from the available answers. The UnifiedQA-v2 model with multiple-choice format will be used in Chapter 4, on a benchmark NSU conversation dataset.

Open-domain Dialog Systems differ from OpenQA, since they have the main goal to chit-chat with humans in a natural and coherent manner. These systems are trained to produce human-like responses (Huang, X. Zhu, and Gao 2020; Roller et al. 2021). The responses produced by these agents are not necessarily factually correct, but rather try to emphasize on other conversational skills such as maintaining a consistent persona and providing domains to talk about (Roller et al. 2021).

2.3.3 Prompting and prompt-based learning

Prompt-based learning, or *prompting*, is a process of creating tasks where some input text in natural language is asking the language model for a response (Liu et al. 2021; Stephen Bach et al. 2022). This response is a text generated by the language model in natural language. A prompt for a summarization task could be

"summarize: The painter couldn't even afford a single brush [...]", where the generated output of the model is interpreted as the prediction for the current task. Prompting has seen a recent growth after the introduction of GPT-3 (T. B. Brown et al. 2020) and also enable models to achieve reasonable zero-shot generalization on unseen tasks as shown by Raffel et al. (2019) and Sanh et al. (2021).

Usually, in supervised learning, when training a model with parameters θ, the model is trained on $P(\boldsymbol{y}|\boldsymbol{x};\theta)$ i.e. to produce an output $\boldsymbol{y}$ from the the input $\boldsymbol{x}$ and the parameters. For newer pre-trained language models, $\boldsymbol{x}$ is usually some text in a natural language, and the output $\boldsymbol{y}$ could be in a variety forms e.g. text or label, depending on the task ahead.

For a text classification task, such as in Chapter 3, $\boldsymbol{y}$ indicates the correct class label, from a fixed set of labels. However, an occurring problem for this training method is that $P(\boldsymbol{y}|\boldsymbol{x};\theta)$ is restricted by the amount of labeled data available. Labeling data is time consuming and hence it is usually not found in large amounts, as with the NSU corpus (2.1.4). Secondly, if we would want to add a new label, we would have to re-train the entire model so it can produce a new fixed output.

Prompt-based learning helps alleviate these problems by allowing the model to train on $P(\boldsymbol{x};\theta)$. This means the model trains on the probability of $\boldsymbol{x}$ when predicting $\boldsymbol{y}$, which in turn lowers the demand for large domain specific datasets (Liu et al. 2021) as this tries to bridge the gap between the pre-training and the downstream task. But there are problems to this as well, namely that during pre-training, the information encoded in the prompt might not be enough for the language model, and there might be large discrepancies between the domain specific data used on a downstream task versus the data used during pre-training (Y. Chen et al. 2022). Lastly, depending on the task, Le Scao and Rush (2021) found through experiments that a prompt can encode up to 3500 labels.

Prompt Engineering

Designing a correct prompt has shown to be critical for a successful model deployment for downstream tasks, especially for zero-shot generalization, but also for the performance of the current task itself (Stephen Bach et al. 2022; Liu et al. 2021). *Prompt engineering* is the process of creating prompts that result in the best performance on downstream tasks. This requires annotated data for validating the performance of a prompt (Y. Chen et al. 2022). The *public pool of prompts* (Stephen Bach et al. 2022) tries to help this process by focusing on creating and sharing such prompts, which enables exploring the best ways to design and

use prompts.

2.4 Summary

In this chapter we started by explaining the notion of NSUs and introduced some of the existing taxonomies created to categorize them. We detailed the taxonomy from R. R. Fernández (2006) that we will be using throughout the coming chapters. We also viewed the fraction of occurrences of NSUs in the daily dialogues of humans to get a sense of the importance of this topic and why we should continue to study it in order to create systems that are able to converse in a natural and coherent manner with humans. We also presented some of the methods for classifying and resolving NSUs. Then, we explained the core architecture (transformer) of most of the current state-of-the-art language models and we presented some of the language models we will be using. Probing tasks and feature attribution, was introduced and are the most commonly used methods for interpretability for these models. Finally, we reviewed some of the tasks and problems in Question Answering. Here, we explained what prompting is, and how it helps to achieve reasonable zero-shot generalization. We will be using prompting in a zero-shot setting in Chapter 4.

Chapter 3

Classification and Feature Attribution of NSUs

In the previous chapter we detailed what a NSU is and saw an overview of the taxonomy of NSUs that we will be using throughout this chapter. In the present chapter we will see, in detail, previous classification methods of NSUs, and we will continue by presenting our method using pre-trained language models. A major goal in the following experiments is to check wether current language models are able to capture linguistic properties to be able to classify NSUs.

In the first section we will present the data that we will be using for our classification task. This is extracted raw text from the BNC corpus of the classified labels from We will compare pre-trained language models that create contextualized word embeddings to a baseline of non-contextualized word embeddings. Our results will also be compared results of previous methods.

We will be using BERT as our pre-trained language model to examine, but we will also see how it compares to a GPT-based model, DialoGPT (Zhang et al. 2020). Since these pretrained language models create contextualized word embeddings, we will see if they are able to capture some linguistic properties of the NSUs that allows these to be classified.

We will then analyze the importance of NSU and the contextual information for the classification task, by using feature attribution. Hopefully, we can find whether the NSU is more important for the language model than the contextual information, and we will view some specific examples of synthetic data that has been feature attributed.

Class	**Occurrences**	d_1	d_2	d_3	$d >= 4$
Plain Acknowledgement (Ack)	599	582	15	2	0
Short Answer (ShortAns)	188	105	21	16	46
Plain Affirmative Answer (AffAns)	105	100	5	0	0
Clarification Ellipsis (CE)	92	76	13	2	1
Repeated Acknowledgement (RepAck)	86	80	2	4	0
Plain Rejection (Reject)	49	48	1	0	0
Factual Modifier (FactMod)	27	23	2	1	1
Repeated Affirmative Answer (RepAffAns)	26	25	1	0	0
Helpful Rejection (HelpReject)	24	18	5	0	1
Check Question (CheckQu)	22	15	7	0	0
Filler	18	16	1	0	1
Bare Modifier Phrase (BareModPh)	15	10	4	0	1
Propositional Modifier (PropMod)	11	10	1	0	0
Direct Sluice (Sluice)	11	10	1	0	0
Conjunct (ConjFrag)	10	5	4	1	0
Total	1283	1123	82	26	51
Fraction	1	0.875	0.064	0.020	0.040

Table 3.1: Per class occurrences for different distances, between the antecedent and the NSU, in the corpus.

3.1 Data

For our data we used the annotated corpus from R. R. Fernández (2006), introduced in Section 2.1.4, containing a total of 1283 samples. Each sample is a reference to a dialogue in the British National Corpus, containing information about where to find a NSU in the BNC, as well as the antecedent, and is labeled by the class the NSU belongs to. Each instance is identified per line where the first number represents the id of the antecedent and the second number represents the id of the NSU. Usually the distance d between the NSU and the antecedent is 1, meaning the antecedent is the immediately preceding utterance in history. In the NSU corpus a total of 87.5% of the dialogues have a distance of 1. Table 3.1 gives us an overview of the different distances for each class in the corpus. We can see that this is a highly imbalanced dataset.

Some classes like, Short Answer, Check Question and Conjunct have a high fraction of NSUs that occurs with distance 2 or more from their antecedent. Short Answer is the only class with a high number of NSUs occurring with distance 4 or more from the antecedent.

British National Corpus

As mention in section 2.1.5, BNC is a large collection of dialogues, mostly written, from national newspapers, journals, academic books, fiction and so on, and the spoken part are from informal conversations that has been transcribed (orthographic transcription). This transcription is done to a XML file. The corpus has been encoded to include part-of-speech tagging, as well as headings and paragraphs containing a number of properties, such as the current genre and details about conversation settings. Every sentence has a unique identifier, which is a counter starting at 1, and these identifiers is what R. R. Fernández has used in her text file to identify the conversations. Each raw word in a sentence is marked with a word tag and contains three properties; a tag following C5 tagset (Leech, Garside, and Bryant 1994), a simplified wordclass derived from C5 in a POS tag, and a lemma derived token from the raw word.

3.1.1 Current dataset

For our dataset we extracted the raw text from the conversations of BNC using the annotated NSU corpus and put it in a json-file; which is an easier format to work with. The assumptions made when extracting the dialogues was that only the dialogue containing the antecedent to the NSU was needed when classifying the NSU. The utterances that were extracted from BNC was from and including the antecedent to and including the NSU. However, it might have been easier for humans to classify a NSU by knowing a bit more context, hence to add additional previous dialogue utterances might have been beneficial for language model as well. However, some antecedents are quite long so for language models with a max input size of 512, we needed to truncate it (this was done by the tokenizer).

During the extraction only the plain text was included, and unknown words (with unknown tag) was skipped. The dataset contains all utterances with an associated speaker where the speaker name was an alphabetical enumeration i.e. first speaker was named “A”, second “B”, and so on. Hence, each dialogue turn with utterances was kept. To easier extract the NSU from the rest of the dialogue, the conversation was divided into two parts, the *contextual information*, and the

NSU, the former contains the antecedent and all the utterances before the NSU, and the latter contain the NSU. For each conversation the NSU class was added, but with Short Answers, R. R. Fernández (2006) divided them into two groups, but these were merged into one.

3.2 Previous Classification Methods

For the classification task both R. Fernández, Ginzburg, and Lappin (2007) and Dragone and Lison (2016) only used the dialogues where the distance between NSU and antecedent were 1, which corresponds to column d_1 in table 3.1. This means the antecedent is the preceding utterance of the NSU. This simplified restriction facilitates the feature extraction procedure, but reduces the size of the data with about 12%, which now contains 1123 samples.

R. Fernández, Ginzburg, and Lappin (2007) have two classification experiments, one where they leave out two classes (Plain Acknowledgements and Fillers) and the second extending the first, where they use all classes. We are interested in latter, so when talking about the classification task below, we mean the classification of all classes. They use four different machine learning algorithms based on decision tree algorithm, or memory-based learning. We will only mention the SLIPPER, which is a rule based learner (Cohen and Singer 1999) and was the one producing best scores of the four. For these algorithms feature extraction is needed, which was done automatically. They use total of 9 features, which are all extractable with help from the PoS tag in BNC. The features are summarized in table 3.2 and described below:

3.2.1 Features

Below we will explain the features used by R. Fernández, Ginzburg, and Lappin (2007) and (some of) Dragone and Lison (2016) for the classification task. The first four features *nsu_cont*, *wh_nsu*, *aff_neg*, and *lex* are relating to the properties of NSU. The features *ant_mod*, *wh_ant*, and *finished* are related to properties of the antecedent of the NSU. The last two features, *repeat* and *parallel* indicates some similarity between the NSU and the antecedent.

nsu_cont

It is supposed to distinguish whether a NSU is a question or a proposition.

wh_nsu

Tells whether the NSU is a *wh*-phrase i.e. *what*, *which*, *when*, *who*, *where*.

Feature	Value
nsu_cont	p,q
wh_nsu	boolean
aff_neg	boolean or empty
lex	p_mod, f_mod, mod, conj, e
ant_mood	decl, n_decl
wh_ant	boolean
finished	fin, unf
repeat	$0-3$
parallel	$0-3$

Table 3.2: Features used for classification by R. Fernández, Ginzburg, and Lappin (2007)

Which will help classify CE and Sluices.

aff_neg

Signals if there is an a word in form of an acknowledgement, a yes-word, or a no-word.

lex

Indicates the appearance of lexical items, which is either in form of model adverb (p_mod), factual adjectives (f_mod), prepositions (mod) or conjunctions (conj). These features are expected to be important for the classification of multiple classes.

ant_mood

Is used to distinguish between declarative and non-declarative antecedent utterances. This features help signal if the antecedent contain a *wh*-phrase, which in turn helps to classify a NSU as a Short Answer.

wh_ant

This is the same as *wh_nsu*, but denotes instead the presence of a *wh*-phrase in the antecedent.

finished

Helps the identification of Fillers, by encoding wether the antecedent has a full stop in form of punctuation, question mark, or exclamation mark.

repeat

Counts the occurrences of same words between the antecedent and the NSU. Can indicate wether the answer is a Repeated Affirmative Answer or Repeated Acknowledgements.

parallel

Counts the PoS tags in common between the antecedent and the NSU, and is helpful for classifying Helpful Rejections.

The features described above a feature engineered by R. R. Fernández, hence are chosen by the author to help the machine learning algorithms perform.

Dragone and Lison (2016) uses the same features as R. Fernández, Ginzburg, and Lappin (2007), and start by trying to replicate the results of the latter. They continue their experiments by adding an additional 23 linguistic features. These features will not be explained in detail, suffice to say that they are divided in 5 categories. These categories of additional features are listed below:

- **PoS-level features**: 7 additional PoS features that are extractable from BNC.
- **Phrase-level features**: 7 features that contain different syntactic structures in the NSU and the antecedent.
- **Dependency features**: 2 features that contain patterns of dependencies in the antecedent, which can be words that are negations or wh-words.
- **Turn-taking features**: 1 features that contains patterns of turn-taking in the dialogue, e.g. if the antecedent and NSU was uttered by the same speaker.
- **Similarity features**: 6 features that contain different numeric measures of similarities between the antecedent and the NSU.

In addition to the extended features set, Dragone and Lison (2016) uses active learning to gain an additional 100 annotated instances. The active learning procedure employed was using a pool-based method, which means that samples where drawn randomly from a set (pool). These unlabeled samples where classified by the existing classifier, using the extracted features, for each class and they added a confidence measure associated, in form of a probability distribution. The supervisor was then prompted to annotate the samples where the classifier is least confident, and hence the classifier would gain more information by knowing these samples.

3.2.2 Results from previous methods

The results for the previous classification methods are shown in table 3.3 [1]. Despite the lack of samples in classes like Conjunct, Check Question, and Sluice, the results from using the selected features very good. However, both fails for the classification of Helpful Reject, which was the lowest scoring class, only achieving a f_1-score of 0.23 and 0.33 respectively. R. Fernández, Ginzburg, and Lappin is discussing the poor results of Helpful Reject, with a possibility that most of the features used to classify Plain Acknowledgement are very similar to the ones used for Helpful Reject, hence they are difficult to distinguish. Dragone and Lison (2016) were not able to replicate the previous results of R. Fernández, Ginzburg, and Lappin (2007) and the results of their replica was not as good results.

3.3 Experimental setup for classification

In this section we will briefly explain the classification experiment. The classification task is a few-shot learning problem (2.2.3) as we only have a few samples for each class. As opposed to R. R. Fernández (2006) and Dragone and Lison (2016) we will be using the entire data NSU corpus, with our dataset containing raw text, explained in section 3.1.1. For the followings tasks we used two different transformer models; BERT (bert-base-uncased) and DialoGPT (Zhang et al. 2020) (microsoft/DialoGPT-small). The models used are from the HuggingFace library (Wolf et al. 2019). The model we will be using of BERT contains 110 million parameters, while DialoGPT contains a total of 117 million.

We first started by dividing the data into a training and a test data. Due to the imbalance in the dataset the data was split in a stratified fashion, to keep an equal proportion of training and testing samples for each class. We used 80% of the data as training data, and the rest as test. This split was mainly to test different model architectures and hyperparameters to see if there was significant differences in performance. Due to the large amount of hyperparameters, we would not be able to test everything, and we wanted to do quick testing to get some insight of different hyperparameters and model architecture. The goal, however, is not to create the best possible model on this classification task, but rather to see if current language models are able to perform on par with previous techniques.

[1] The per class results of Dragone and Lison (2016) were only available with two decimals, so the entire table was adjusted to this.

	SLIPPER			SMO		
Class	**Precision**	**Recall**	**f1-score**	**Precision**	**Recall**	**f1-score**
Ack	0.97	0.96	0.96	0.97	0.98	0.97
AffAns	0.83	0.87	0.85	0.81	0.90	0.85
BareModPh	0.83	0.69	0.76	0.77	0.75	0.75
CE	0.96	0.94	0.95	0.88	0.92	0.89
CheckQu	0.87	1.00	0.93	1.00	1.00	1.00
ConjFrag	1.00	1.00	1.00	1.00	1.00	1.00
FactMod	1.00	1.00	1.00	1.00	1.00	1.00
Filler	0.70	0.56	0.62	0.82	0.83	0.78
HelpReject	0.30	0.19	0.23	0.31	0.43	0.33
PropMod	1.00	1.00	1.000	0.92	1.00	0.95
Reject	0.78	1.00	0.87	0.90	0.90	0.89
RepAck	0.84	0.92	0.88	0.77	0.77	0.77
RepAffAns	0.68	0.73	0.70	0.72	0.55	0.58
ShortAns	0.85	0.84	0.85	0.92	0.86	0.89
Sluice	0.94	1.00	0.97	0.80	0.84	0.81
Weighted avg.	0.92	0.93	0.92	0.91	0.91	0.91

Table 3.3: Per class performance of the previous results of R. Fernández, Ginzburg, and Lappin (2007) and Dragone and Lison (2016) respectively, using 10-fold cross-validation.

After some experimentation, we selected the best performing model and used 10-fold cross validation, on the entire dataset, to test the models and gain more reliable results. Finally, for the model training the criterion used was mostly cross-entropy loss, but experimentation with Dice Loss (X. Li et al. 2020) was also done, as this is according to the authors helps narrow the gap between the F1 score in training and evaluation. The optimizer used was AdamW (Loshchilov and Hutter 2019).

3.3.1 Model architecture

BERT and DialoGPT were used as pre-trained language models for the word embeddings. Since BERT and DialoGPT differs slightly in architecture a

comparison between these would be interesting. We might expect BERT to perform better as it is deep bidirectional, has a classification token, and is an encoder. However, DialoGPT is trained on conversational-like data, so this might help it classifying our dataset of conversations. Three different classifiers (heads) were experimented with for the classification task; a linear, a dense, and a RNN (with a linear layer). These heads, go on top of the language model to classify the embeddings. We tested the different heads for each model pre-trained model, as well as experimentation with pooling of entire input, pooling of contextual information, c, and NSU u separately (only for linear and dense classifiers). It was experimented with performing an attention (not multi-head), explain in section 2.2.1, between c and u, $\text{Attention}(c, u, u) = \text{softmax}(\frac{CU^T}{\sqrt{d_k}})U$. The intuition for pooling and attend these separately was by trying to capture relationship between c and u. In the end, the results were slightly worse (about 2-3 percentage f1-score) than only pooling everything.

DialoGPT

DialoGPT is using the GPT-2 (Radford, Wu, et al. 2019) as basis (see section 2.2.2), which in turn is based on the generic transformer language model we explained in section 2.2.1, but has been trained 147 million conversation-like exchanges which was extracted from Reddit. The reason for choosing this model is because of its training on conversational data. Our intuition is that this will help when for this classification task. It is trained on this data to produce conversational turn generation. DialoGPT is trained on the objective to optimize $p(T_K, ..., T_2|T_1)$, where the target T is as ground truth response, and $T_1, ..., T_K$ is a multi-turn dialogue of K turns. This is the product

3.3.2 Input Text

There was some experimentation with the input text of the models. For BERT there is a separator token ([SEP]), but it is only used in pre-training between two sentences. As we had the contextual information c, and the NSU u we put the separator between these, hence the input $X = c$[SEP]u. However, each turn where separated with a newline character. We experimented with using the current speaker before the utterance, e.g. "A:", for speaker A, and also using a hyphen character (-).

This was make it a more like a written dialogue, but neither had any significant impact on the results either way, so these were removed. What was used in the end was only a newline character, to separated each turn by speaker.

This means if a speaker uttered multiple utterances after each other, these were concatenated, but when a new speaker started speaking, a newline character was added in between. The NSU, however, was always put after a separator token.

For DialoGPT, turns are normally separated between the eos_token (end-of-string token). Here, we experimented also with adding the current speaker, and hyphen character, but again there was no significant benefit, hence the eos_token was kept as the separator token. Again, a separator token was put before the NSU.
One thing to keep in mind with both the input text for BERT and DialoGPT is; if a speaker uttered the last utterance in the contextual information, hence it is the preceding utterance of the NSU, and the same speaker uttered the NSU as well, the language models would not have any way to tell that these were uttered by the same person.

3.3.3 Training

During training there was a difference in the number of epochs for each of the two language models. The models that used BERT converged faster during training than DialoGPT, and we used 15 epochs for training, while 35 epochs were used for training the models that used DialoGPT. For each epoch we evaluated on the validation data using the best *macro average* (see section 3.3.4) scoring model. This was with the intuition of achieving a model that could generalize well for each class, hence classifying each class reasonably well, rather than classifying the most frequent classes really well.

We then kept the weights a model for each time the current best metric was beaten, and loaded the weights of the best epoch for the training validation. This means that the model will be slightly overfitted on the validation data, since we use the weights of the model were it performs especially good on this data.

3.3.4 Metrics

To give a recap of the metrics used, we will briefly explain them in this section. Classification of samples can fall into 4 categories:

- True positives (TP), for a class c_1, are samples that are classified in c_1 and belongs to c_1.
- True negatives (TN), for a class c_1, are samples that are classified in another class c_2 and belongs to c_2.

- False positives (FP), for a class c_1, are the samples that are classified to c_1, but belongs to another class c_2.
- False negative (FN), for a class c_1, are the samples that are classified to another class c_2, but belongs to c_1.

We are mostly interested true positives, false positives, and false negative for the coming metrics. We will denote N as all samples in the dataset, and N_c as the samples of a class c.

Accuracy

Accuracy is the most basic metric in this section. For multiclass classification it is the ratio of correctly classified to all samples. Below we see the accuracy for a class c:

$$\text{Accuracy}_c = \frac{\text{TP}_c}{\text{N}_c}$$

Precision

Precision can be thought of the fraction of relevant instances. For multiclass we must calculate the precision per class and for a class c is it calculated as:

$$\text{Precision}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c}$$

Recall

Recall can be thought of the fraction of relevant instances that was acquired. For multiclass we must calculate the recall per class and for a class c is it calculated as:

$$\text{Recall}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c}$$

F-score

The F-score takes into account both the precision and the recall. F_1 score, which we will be using, is when precision and recall is given equal weighting is the harmonic mean of precision and recall (Powers 2019). For a class c it is calculated as follows:

$$F_{1,c} = \frac{2 \times \text{Precision}_c \times \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c} = \frac{\text{TP}_c}{\text{TP}_c + \frac{1}{2}(\text{FP}_c + \text{FN}_c)}$$

Average of metric

After calculating a per-class score for a metric, we want to get an average for all classes. A metric can be the accuracy, precision, recall, or f_1-score, and we can take an average in several ways, but we are interested in the *macro average* and the *weighted average*.

Macro Average

We can take an macro average (Opitz and Burst 2021) for a metric m is the unweighted mean of each class c, for a total of k classes, where m_c is the score of a metric for a class c:

$$\text{macro}_{avg} = \frac{\sum_{c=1}^{k} m_c}{k}$$

Weighted Average

The weighted average, unlike macro average, takes into account the imbalance of labels in each class when averaging a metric. It is calculated as follows:

$$\text{weighted}_{avg} = \sum_{c=1}^{k} m_{\times} N_c$$

3.4 Results: Classification of NSU

In the following section we will view the empirical results of the classification task using the two different pre-trained language models with different classifiers on top. We will compare them to a baseline that is using non-contextualized word embeddings, as well as to previous classification methods.

3.4.1 Baseline

As a baseline we wanted to use a model that creates non-contextualized word embeddings and compare it to a model that creates contextualized word embeddings. We decided to use skip-gram (Mikolov et al. 2013) from a word2vec ("word2vec-google-news-300"), from the Gensim library (ehek and Sojka 2010).

	Precision	Recall	f1-score
Ack	0.467	1.000	0.637
AffAns	0.000	0.000	0.000
BareModPh	0.000	0.000	0.000
CE	0.000	0.000	0.000
CheckQu	0.000	0.000	0.000
ConjFrag	0.000	0.000	0.000
FactMod	0.000	0.000	0.000
Filler	0.000	0.000	0.000
HelpReject	0.000	0.000	0.000
PropMod	0.000	0.000	0.000
Reject	0.000	0.000	0.000
RepAck	0.000	0.000	0.000
RepAffAns	0.000	0.000	0.000
ShortAns	0.000	0.000	0.000
Sluice	0.000	0.000	0.000
Weighted avg.	0.218	0.467	0.297

Table 3.4: Results of the baseline model (skip-gram).

This had a vector dimension of 300, and on top of this we max-pooled the embeddings and used a linear layer for classification.

The model converged after a couple of epochs to the majority class, and was unable to extract other details in the embedding. Hence, this baseline is the same as an majority baseline, and it means that it got a score of zero for every other class than Plain Acknowledgement. The results are shown in table 3.4.

3.4.2 Experimentation results

The results of experimentation with different heads on top of the pre-trained language models are listed in table 3.5 for BERT and table 3.6 for DialoGPT. The listed results are the weighted average of all the classes for precision, recall, and f1-score.
We are mostly interested in the f_1-scores. For both pre-trained models we can

Classifier (head)	Precision	Recall	f1-score
Linear	0.873	0.875,	0.870
Dense	0.921	0.914	0.914
RNN	0.881	0.875	0.873

Table 3.5: Shows the weighted average scores on the test set for different classifiers for BERT.

Classifier (head)	Precision	Recall	f1-score
Linear	0.823	0.840	0.825
Dense	0.854	0.840	0.836
RNN	0.770	0.821	0.7921

Table 3.6: Shows the weighted average scores on the test set for different classifiers for DialoGPT.

see that the best measure was for the dense classifier. However, this model is likely to be a little overfitted on the validation data, but this can also apply to the other classifiers. As the dense classifier got the best results, we continued using this for the 10-fold cross-validation measure in the next task. However, it does not automatically mean that the dense classifier will the highest score on 10-fold cross-validation task, so the other classifiers could be tested on this specific task, in future work.

3.4.3 10-fold cross-validation

After experimenting with different hyperparameters and architecture we decided to use the best scoring classifier, and perform 10-fold cross validation on this architecture. This is in order to lower the variance in the experiments, and was also done in previous experiments by R. Fernández, Ginzburg, and Lappin (2007) and Dragone and Lison (2016).

To maintain the same ratio of class instances for each fold we used stratified 10-fold validation. However, since the test data from the classes with the lowest frequency, only contains 10-11 samples, stratified 10-fold cross-validation will only give 1 sample in the test data for these classes, as there are no overlap of the data for each fold.

As these classes contain 1 or more multi-turn dialogue, the training samples

is for the model is not always containing these, and also might only confuse the model when included during training. When an instance of a multi-turn dialogue is only available in the test set, and the model only has seen instances of single-turn conversation of this class, it will be difficult to classify such instance.

The classes that have a very low frequency of samples (11 or less) include Propositional Modifier (PropMod), Direct Sluice, and Conjunct (ConjFrag). We can see that both BERT- and DialoGPT-based classifiers are not able to perform well on these classes, especially in PropMod and ConjFrag. This is probability due to low frequency of instances in these classes in general. For PropMod and ConjFrag it might also be that they only contain 1 instance of multi-turn dialogue.

Discussion

In table 3.7 we can see the results of using BERT with a dense classifier, while in table 3.8 we can see the results of using DialoGPT with a dense classifier. Due to this very imbalanced dataset, where the about 47% of the samples are in Plain Acknowledgements (see 2.2), it is difficult for the model to perform well on the classes with the lowest frequency of samples.

For the NSU class Short Answers, the language models did reasonably well, both achieving a f_1-score well over 80%. The thing to keep in mind with these samples is that 37% of these are dialogues where the antecedent with a distance of 2 or more turns away from the NSU and as much as 24% of the total Short Answers are of distance 4 or more. This means that the language model is able to correctly classify multiple of these long-distance NSU in Short Answer, which has not been done in previous experiments.
Compared to our baseline approach using non-contextualized word embeddings, we can see that both BERT, and DialoGPT, are able to capture some linguistic properties of the conversation, that enables them to classify other classes than only the majority class. The baseline was not able to do this, which made it converge towards the majority class.

When comparing BERT (3.7) to DialoGPT (3.8), we can see that the classifier using BERT got better f_1-scores than the classifiers using DialoGPT, in general. This is true for all classes, except CheckQu, PropMod. It is somewhat to be expected that BERT performs better than DialoGPT, as BERT is fully bi-directional and usually a good choice for classification tasks (Devlin et al. 2019).

	Precision	Recall	f1-score
Ack	0.940	0.958	0.949
AffAns	0.845	0.845	0.831
BareModPh	0.500	0.400	0.433
CE	0.862	0.944	0.900
CheckQu	0.883	0.917	0.873
ConjFrag	0.233	0.300	0.250
FactMod	0.825	0.750	0.776
Filler	0.408	0.400	0.377
HelpReject	0.425	0.417	0.384
PropMod	0.300	0.250	0.267
Reject	0.893	0.880	0.878
RepAck	0.712	0.676	0.683
RepAffAns	0.520	0.617	0.535
ShortAns	0.888	0.847	0.857
Sluice	0.750	0.750	0.733
Weighted avg.	0.857	0.861	0.851

Table 3.7: Per class average results when performing 10-fold cross-validation on BERT with a dense classifier.

3.4.4 Comparison with previous results

It is important to keep in mind that these results are not directly comparable to the previous results obtained by R. Fernández, Ginzburg, and Lappin (2007) and Dragone and Lison (2016). This is because, as mention previously, both are only using the conversations where the antecedent is the preceding utterance of the NSU. The classification done by the language models are therefore a more complicated task than previously done, as about 12% of the total data are multi-turn dialogue. Another difference is that the previous methods used extracted features from the text to classify, while our models used only the raw text. Nevertheless, the language models are more complex models, containing hundred millions of parameters, so we would expect them to perform somewhat comparable. Table 3.9 show the weighted average scores of the previous results, along with the baseline, and from BERT and DialoGPT.

	Precision	Recall	f1-score
Ack	0.925	0.948	0.936
AffAns	0.785	0.855	0.808
BareModPh	0.400	0.300	0.333
CE	0.838	0.796	0.809
CheckQu	0.833	0.717	0.740
ConjFrag	0.000	0.000	0.000
FactMod	0.967	0.800	0.860
Filler	0.333	0.300	0.307
HelpReject	0.340	0.383	0.343
PropMod	0.417	0.450	0.395
Reject	0.817	0.910	0.851
RepAck	0.650	0.531	0.538
RepAffAns	0.275	0.250	0.253
ShortAns	0.811	0.857	0.829
Sluice	0.350	0.450	0.383
Weighted avg.	0.814	0.823	0.810

Table 3.8: Per class average results when performing 10-fold cross-validation on DialoGPT with a dense classifier.

Model	**Precision**	**Recall**	**f1-score**	**# samples**
SLIPPER	0.916	0.927	0.920	1123 (only d_1)
SMO (AL + extended features)	0.913	0.907	0.905	1223 (only d_1)
Baseline (skip-gram)	0.218	0.467	0.297	1283
BERT	0.857	0.861	0.851	1283
DialoGPT	0.814	0.823	0.810	1283

Table 3.9: The weighted average scores on 10-fold cross validation on NSU data.

It is also important to note that the goal was never to achieve state-of-art performance on the classification task, but rather to see to which extent current pre-trained language models were able to perform on this task. That said, our

models using pre-trained language models were not able to achieve as high score as the previous methods that are using simple machine learning algorithms. Though there is some class were both the language models scored higher than the previous methods. This was for the class Helpful Reject, where BERT and DialoGPT got a score of 38.4% and 34.3% respectively, compared to the best previous result from the SMO classifier at 33%. For CE, BERT got a score of 90%, which was better than the SMO classifier at 89%, but worse than SLIPPER at 95%.

As mentioned above, the poor results in some of the classes are probably due to the low frequency of samples in these classes. Although pre-trained language models usually can cope with scarce and imbalanced, Ack contains about 46.7% the of the samples in the dataset making it very imbalanced. And we would probably achieve higher performance with larger models i.e. containing even more parameters. Since, it is generally true, as we can see in the experiment in chapter 4, that the more parameters a language model has the better it performs on a challenge.

Fine-tuning larger models were not possible due to computational limitations, but we might expect to see significant improvements using models with around 1 billion parameters or more (for reference see table 4.3).

3.5 Feature Attribution

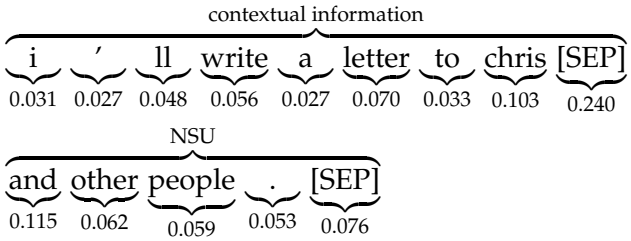
We will in this section describe the feature attribution challenge, and the experiments that were done. As we explained in section 2.2.4, the goal of feature attribution is to attribute a score for each input token. This score will be a percentage score (when normalized), that tells how much the token contributed to the output. Different feature attribution algorithms will give differences in the attribution for each token, as they calculate these attributions differently. The following task will be using Integrated Gradients (Sundararajan, Taly, and Yan 2017), which has been explained in section 2.2.4, as it fulfill sensitivity and implementation invariance (Guidotti et al. 2018; Sundararajan, Taly, and Yan 2017).

To exemplify process of feature attributions, we can consider the following dialogue:

A: I'll write a letter to Chris
B: And other people.

Using the Integrated gradients algorithm, this conversation will get the

following feature attribution scores by BERT (trained on NSU classification task):



Since BERT divides some words into multiple tokens, each of these tokens are attributed a score.

3.5.1 Experimental setup

For the feature attribution, we used models trained on the NSU classification task (see 3.3). These models are the same as explained in 3.3.1, using BERT and DialoGPT as pre-trained language model with a classifier on top. When training we used training and test datasets described in section 3.1.1. As mentioned earlier the training data contained 80% of the samples, and the test data contained the rest (20%). We decided to use a dense classifier for both pre-trained language models, as this was the classifier scoring highest, for both language models, when experimenting with architectures in the classification task.

There was, however, a difference in the model using BERT. For BERT instead of including the classification token ([CLS]), this was masked out and excluded completely (which was done through changing a hyperparameter), hence had no effect whatsoever on the other tokens, both during training and feature attribution process. This is due to the difficulty of interpreting the meaning of an attribution for this token, since it is difficult to know wether the antecedent or NSU were most important, or how they relate to each other, when this token is used. When masking out this token, it will get a value of 0, when calculating the feature attributions. Hence, BERT with a dense classifier was trained again, but with the CLS token masked out, on the training data, and the results are shown in table 3.10, along with the previous results of DialoGPT and BERT with CLS token.
We can see a slight drop in performance of the BERT model, when not using the CLS token. However, it still gets better f_1-score than DialoGPT. These results are not 10-fold cross-validated, so they are only meant as an indication of the performance.

Model	Precision	Recall	f1-score
BERT (without CLS)	0.887	0.895	0.887
BERT (with CLS)	0.921	0.914	0.914
DialoGPT	0.850	0.840	0.835

Table 3.10: Shows the weighted average scores on the test set for the classification task; all models use a dense classifier.

Technical details for calculating feature attributions

Using a model trained on the classification task, we calculated the attributions for each sample in the test set for both models. When calculating the feature attributions, they need to be calculated with respect to a given label. As such we could either use the label the model predicted, or the true label of the sample. Since we wanted to see the attributions for different classes, and the scores with respect to the dialogues of these, we calculated the attribution with respect to true label. So even if the model actually wrongly predicted a class, we calculate the feature attribution with respect to the true class label. After feature attributions are calculated, L2-norm is used to retrieve a score for each token, and then the values are normalized by a division of the sum of the attribution scores. The final scores are then percentage scores of how much the tokens attributed to the output.

We used Captum library (Kokhlikyan et al. 2020) for calculating the feature attributions. The feature attribution algorithm used was always Integrated Gradients. For a language model, such as BERT or DialoGPT, we need to input the word embeddings of the input text, and not the tokenized ids, to the model. This is regardless of the selected feature attribution algorithm. Hence, a wrapper for the classification model was created to handle the input embeddings, and the wrapper along with the current word embeddings could be passed to the feature attribution algorithm.

The tasks

We have created two different task for evaluating feature attributions of the models. The first will be to take the feature attributions over the entire dataset, and try to interpret some aggregated values. The second will be to look at the feature attribution for specific examples, and see how they change then we manipulate either the contextual information, or the NSU, to let the dialogue fall into another class.

3.5.2 Feature attribution scores over test data

There is some difficulty when calculating the feature attributions for an entire dataset, even though, as in our case, the test set used is not very large. This relates the interpretation the attributions scores across multiple conversations, which needs to be somewhat categorized. Hence, we want to find differences in the attributions scores between the contextual information, and the NSU, when classifying a dialogue.

We will start by some definitions; for a sample in the dataset, with tokens $X_i = x_{i,1}, ..., x_{i,l}$ and length l, it will contain the contextual information $C_i = x_{i,1}, ..., x_{i,k}$, with length k, and the tokens of the NSU $U_i = x_{i,k+1}, ..., x_{i,l}$ with length $j = l - k$. Here, both the contextual information and the NSU is part of the sample; $C_i \in X_i$ and $U_i \in X_i$.

For simplicity we will call the feature attribution function $\text{Attr}(X)$ The feature attributions for X_i will be $\text{Attr}(X_i) = A_i = a_{i,1}, ..., a_{i,l}$, and the feature attributions for contextual information are $\text{Attr}(C_i) = A_{i,c} = a_{i,1}, ..., a_{i,k}$ and the feature attributions for the NSU are $\text{Attr}(U_i) = A_{i,u} = a_{i,k+1}, ..., a_{i,l}$.

Since, the length k of the contextual information, is usually much longer than the length j of the NSU, we would need some way to measure these with respect to the length of the tokens. The lengths k and j also vary greatly for the different samples i in the dataset. If we would only sum the feature attributions of $A_{i,c}$, we would be more likely to get a much higher value than the total addition of $A_{i,u}$, given that each value in A_i similar.

To get an indication of how much the language model is dependent on either the contextual information or the NSU, when classifying, we will take the average of $A_{i,c}$ and $A_{i,u}$. However, the average alone isn't a good way for measuring the importance of the contextual information versus the NSU. This is because is the contextual information has more token in it, it will automatically get a lower average, since we are using percentage values.

To try and balance this out, we perform weighting for both the contextual utterance and the NSU. The weight of the contextual information is the number of containing tokens divided by the total amount of tokens, hence $\frac{k}{l}$. The weight of the NSU would be the same, but using the number of tokens in the NSU, hence $\frac{j}{l} = \frac{l-k}{l}$. We will be using this to try and get some indication of the importance of the contextual information versus the NSU.

We would expect the NSU in the conversation to be the most important

utterance to classify the NSU, as this should give the most information about which class it belongs to.

3.5.3 Results and discussion

We will now see some distributions of the contextual information and the NSU, for both models used. To get a sense of how many tokens there usually are in the contextual information compared to the NSU, we can take a look at figure 3.1. It contains the distribution of number of tokens in the contextual information and the NSU, when using the BERT tokenizer; using the tokenizer of DialoGPT, we would get similar results.

In figure 3.2a and 3.2b we see the distribution of attribution scores, for both BERT and DialoGPT respectively. These are per token attribution scores of the samples in both the contextual information and the NSU, hence shows us how high scores the individual tokens in the contextual information gets compared to the individual tokens in the NSU.

Finally, we can see the results of aggregation of the summation, the average in figure 3.3 and 3.4 respectively. Below we will explain how the aggregated scores are calculated.

Distributions of contextual information and NSU

The distributions of the attribution scores of the contextual information and the NSU were obtained by extracting all attributions of the dataset, individually for each and then plotted. The count in contextual information are higher than for NSU, which is simply because there are more tokens in the contextual information, than in the NSU. The plot for BERT is shown in figure 3.2a, while for DialoGPT it is shown in figure 3.2b.

We can see that for both BERT, and DialoGPT the attribution scores of the NSU are generally higher than for the contextual information.

Summation

We can find the summation of the attribution scores for the contextual information $\Sigma_C = \Sigma_{c,0}, ..., \Sigma_{c,N}$, and the NSU $\Sigma_U = \Sigma_{u,0}, ..., \Sigma_{u,N}$ individually, for each sample, where N is the total length of the dataset. We can calculate $\Sigma_{c,i}$ and $\Sigma_{u,i}$,

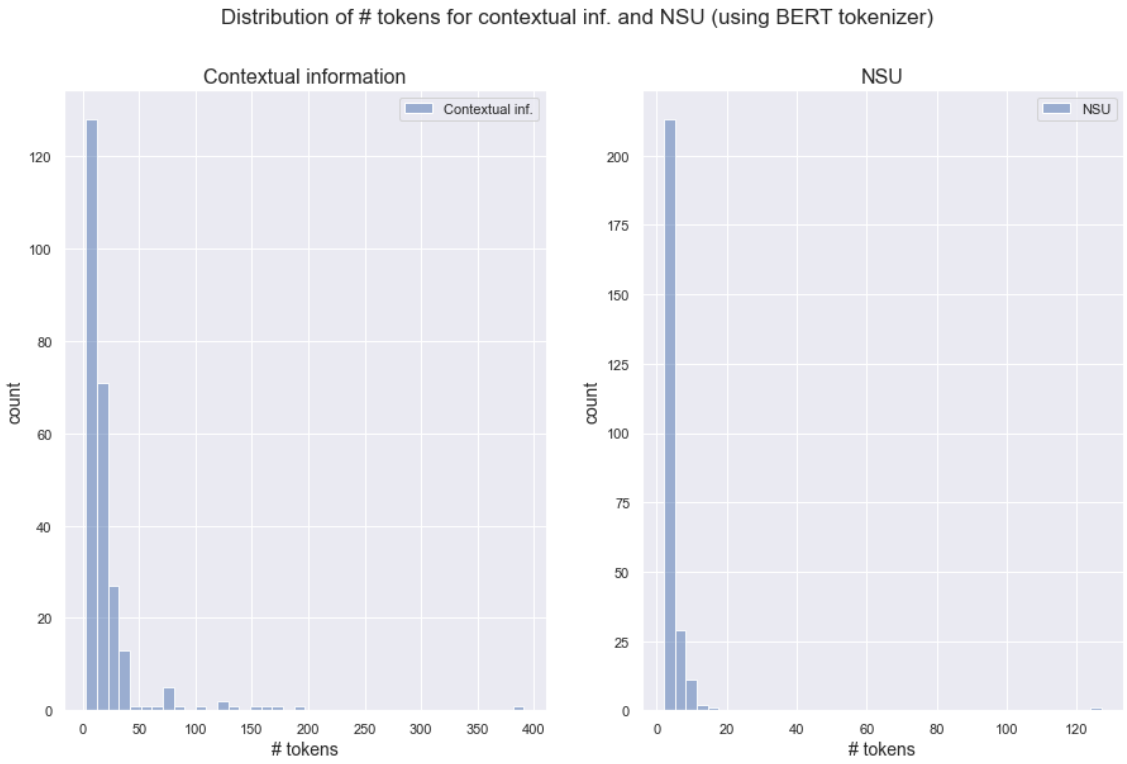


Figure 3.1: Distribution of the number of tokens in the contextual information and the NSU, when using BERT tokenizer.

for a sample of attributions $A_i = a_{i,0}, ..., a_{i,l}$ the following:

$$\Sigma_{c,i} = \sum_{\beta=0}^{k} a_{i,\beta}$$

$$\Sigma_{u,i} = \sum_{\beta=k+1}^{l} a_{i,\beta}$$

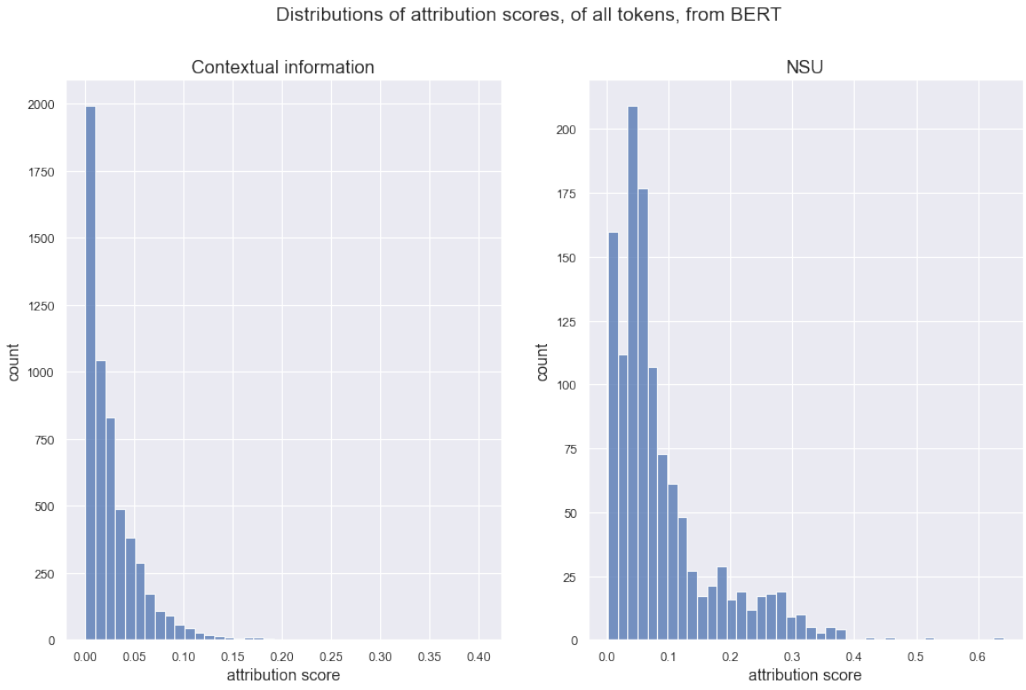
We then plot the values of S_C against S_U for a specific model as seen in figure 3.3.

Weighted average

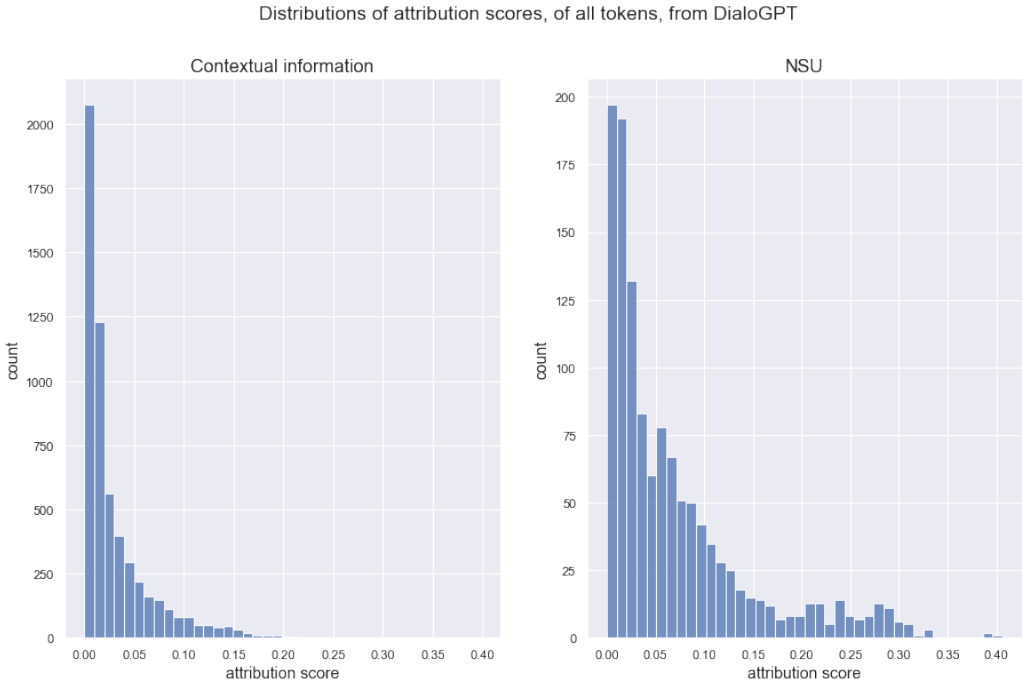
Finding the weighted average of the attribution scores for the contextual information $\mu_C = \mu_{c,0}, ..., \mu_{c,N}$ and the NSU $\mu_U = \mu_{u,0}, ..., \mu_{u,N}$, can be found by calculating $\mu_{c,i}$ and $\mu_{u,i}$ for each attribution score A_i:

$$\mu_{c,i} = \frac{k}{l} \times \frac{1}{k} \sum_{\beta=0}^{k} a_{i,\beta}$$

$$\mu_{u,i} = \frac{l-k}{l} \times \frac{1}{l-k} \sum_{\beta=k+1}^{l} a_{i,\beta}$$



(a)



(b)

Figure 3.2: Distributions of the attribution scores of the contextual information and NSU for both BERT and DialoGPT.

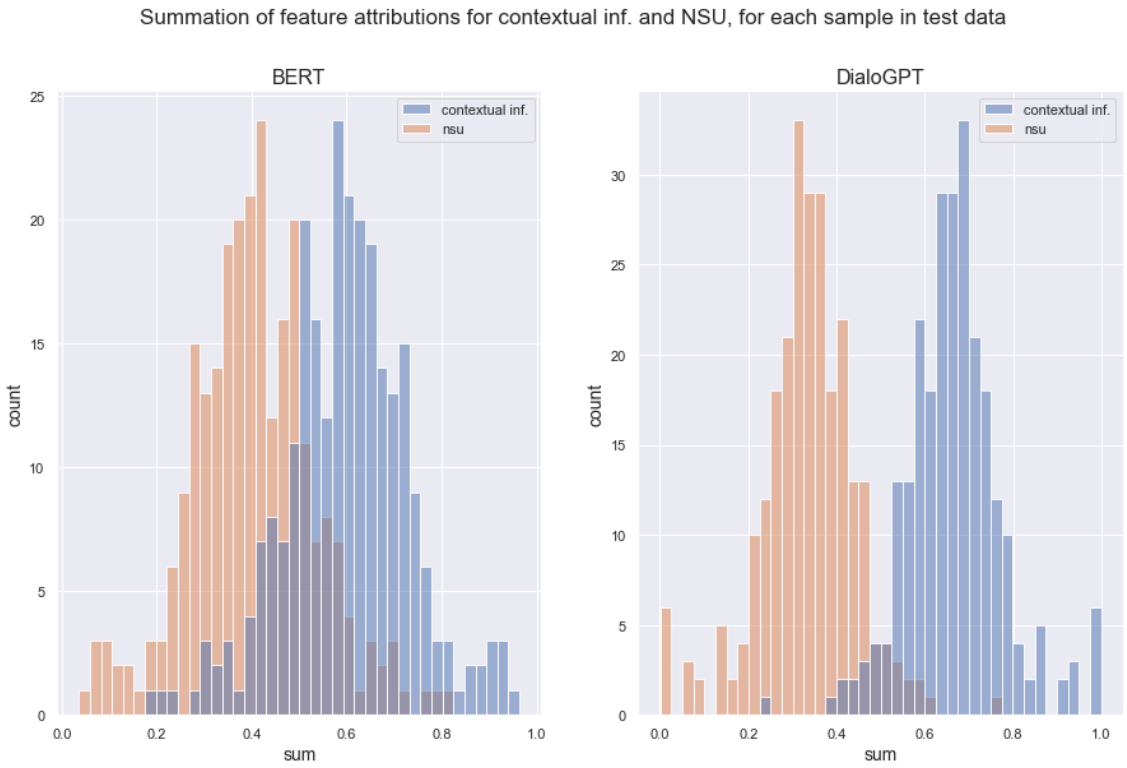


Figure 3.3: Histogram of the summation of contextual information, and NSU, for each sample in test data.

The plot of μ_C and μ_U is shown in figure 3.4.

Discussion

Since the contextual information often contains more tokens than the NSU, as seen in figure 3.1, the summation for each sample is generally higher than the summation of NSU (3.3). We know from the distributions of the attribution scores, that the individual tokens in the NSU often has higher values than the contextual information (figure 3.2a and 3.2b), but for the weighted average we can see that the contextual information is slightly higher than the NSU, as shown in figure 3.4.

Though, in general, the attribution scores for the contextual information are lower than for the NSU, the tokens of the NSU does not have a high enough attribution score to make up for the weighted average difference. This might simply be that the language models needs to process a lot more tokens in the contextual information and when it processes each of the tokens, the aggregated attribution score gets high.

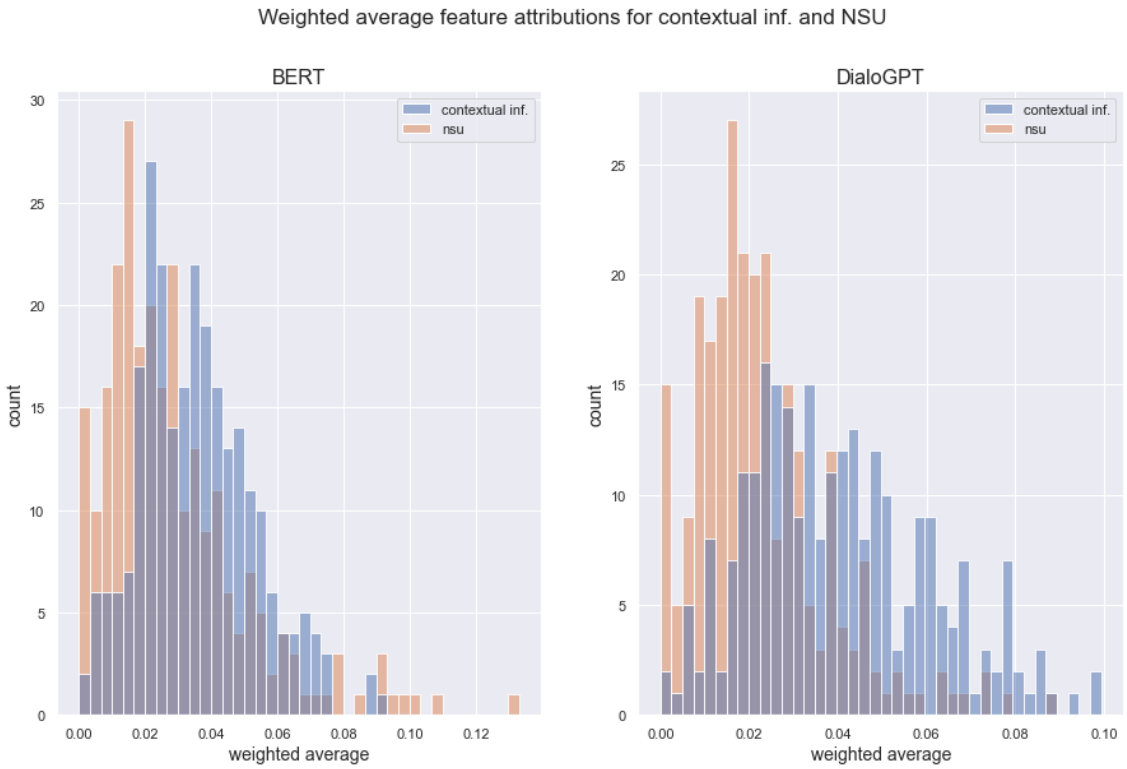


Figure 3.4: Histogram of the average of contextual information, and NSU, for each sample in test data.

It would have been interesting to perform a more granular aggregation of the feature attribution e.g. for multi-dialogues we could find the average feature attribution per utterance. This might let us see for antecedents that are of distance 2 or more of the NSU, if the antecedent is more important than the other utterances in between the antecedent and the NSU, for classifying a NSU.

3.5.4 Feature attribution for synthetic dialogues

In this section we will see some synthetic conversation, and we will see how the feature attribution changes when there are small changes to the dialogues, which makes the dialogue fall into another NSU class. We will be using the dialogues in table 3.11 containing one dialogue per NSU class [2].

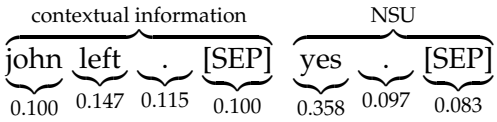
For the classification model using BERT, the dialogues from (3.11) gives the following attribution scores using Integrated Gradients (the scores for DialoGPT can be viewed in Appendix A):

[2]These are modified dialogues from Jonathan (2012, Chapter 7).

NSU Class	Example
Ack	A: John left. B: Yes.
AffAns	A: Did John leave? B: Yes.
BareModPh	A: John left? B: Yesterday.
CE	A: Did John leave? B: John?
CheckQu	A: John left. A: Okay?
ConjFrag	A: John left. B: And Mark.
FactMod	A: John left. B: Excellent!
Filler	A: Did John... B: leave?
HelpReject	A: Did John leave? B: No, Mark.
PropMod	A: Did John leave? B: Probably.
Reject	A: Did John leave? B: No.
RepAck	A: John left. B: John left, hmm.
RepAffAns	A: Did John leave? B: John, yes.
ShortAns	A: Who left? B: John.
Sluice	A: John left. B: Why?

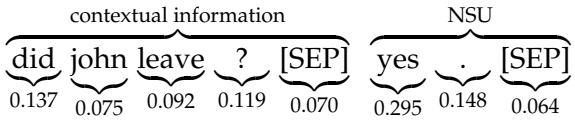
Table 3.11: Similar dialogues with small changes to fall in another class.

Ack



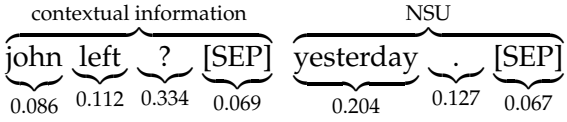
(3.1)

AffAns



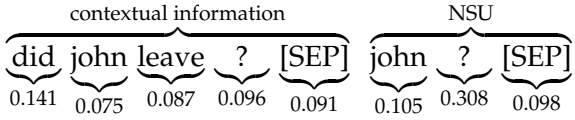
(3.2)

BareModPh



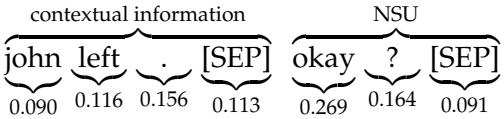
(3.3)

CE



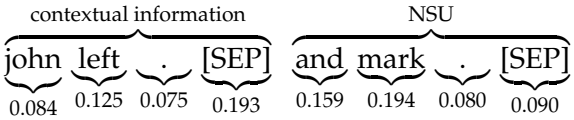
(3.4)

CheckQu



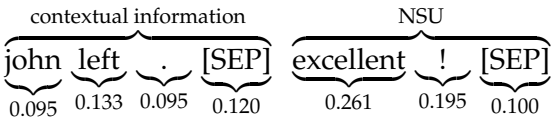
(3.5)

ConjFrag



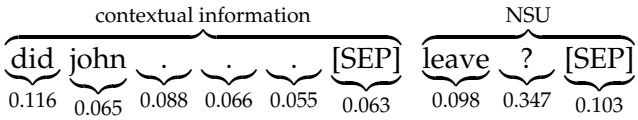
(3.6)

FactMod



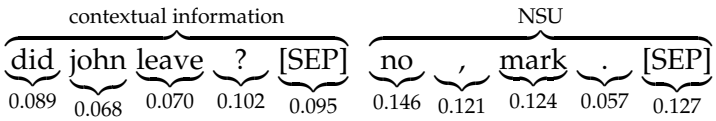
(3.7)

Filler



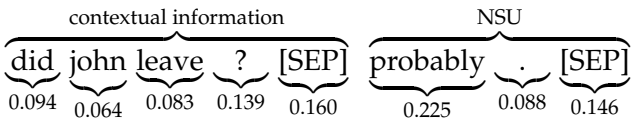
(3.8)

HelpReject



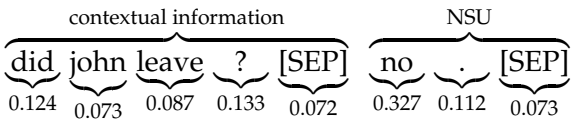
(3.9)

PropMod



(3.10)

Reject

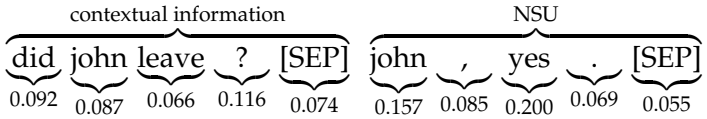


(3.11)

RepAck

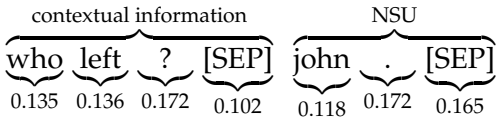
(3.12)

RepAffAns



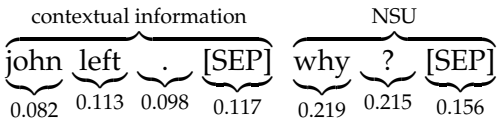
(3.13)

ShortAns



(3.14)

Sluice



(3.15)

Generally, the feature attributions are somewhat evenly distributed for all dialogue, and there are not large differences between the scores of the tokens, which might be somewhat unexpected. Since we are classifying the NSU, it seems plausible that the NSU itself should be significantly more important than the utterance(s) in the contextual information. We can see that, indeed, the tokens in the NSU are more important for the classification task, than the tokens in the contextual information, but not always by a large margin.

Above, we can see for the dialogue for AffAns (3.2) and Reject (3.11), the polar answer (yes/no), are given a relatively high value, of around 30%, and are the most important tokens for these dialogues. The lowest scoring token in AffAns was the middle separator token, at only 7%, while the lowest scoring token for the Reject dialogue was both "john", in the contextual information and the last separator token, in the NSU, at 7.3%.

In these examples, the highest scoring token was 22.5% and 25.4% more important than the least important token, for the AffAns and Sluice respectively. ShortAns, on the other hand, has only a difference of 7% between the highest and the lowest scoring token. For BareModPh is the only dialogue where a token in the antecedent is more important than the NSU, for these examples. In general, we can see that question marks, in both the antecedent and NSU, are usually given a higher score than the rest of the tokens in the antecedent, which is true for almost every dialogue.

Discussion

It is difficult to draw any conclusions for how important, for the model, the contextual information is versus the importance of the NSU, when classifying a NSU. However, we have seen that the tokens of the NSU usually get higher scores than the tokens in the contextual information, and the the highest scoring token usually lies in the tokens of the NSU.

So we could say that the individual tokens in the NSU seems to be important than the individual tokens in the contextual information. But as shown in the weighted average of attribution scores (3.4) the contextual information seems to be more slightly important as a whole than the NSU, in general. Though this could simply be because the contextual information generally contains a lot more tokens than the NSU (3.1), hence it contains more information for the language model to process.

For future experiments, it would be interesting to the the feature attributions for a multi-turn conversation, for a dataset such as that in Chapter 4. Then we could calculate the importance of each utterance in the dialogue, when the models is answering a question posed to that dialogue. When questions posed to that dialogue are asking about something that needs to be inferred from a NSU, we might be able to capture which aspects of the conversation, specifically which utterances, that are the most important for the language model to answer that question.

3.6 Summary

In this chapter we presented two different experiments, the first the task of classifying NSUs using pre-trained language models with the transformer architecture. Secondly we calculated feature attribution scores of such models trained on the classification task to try and interpret the attribution scores of the contextual information and the NSU.

We started the chapter by introducing the NSU corpus of R. R. Fernández (2006), based on conversations from BNC, and continued by explaining previous methods done in the classification of NSUs for the taxonomy introduced in section 2.1.2. The first work was from R. Fernández, Ginzburg, and Lappin (2007), where they tested 4 different machine learning algorithms, using a set of 9 features that could be automatically extracted from BNC. Dragone and Lison (2016) extended this work by adding additional features to expand the features set, giving a total of 32 features, and using active learning to increase the amount

of samples.

We then presented our experimental setup, using all 1283 samples in the NSU corpus, which contains dialogues where the NSU is of distance more than 1 utterance away from the antecedent. This differs from both previous methods, as these only used dialogues where the antecedent was the preceding utterance of the NSU, and all input data is raw text in natural language. Using BERT and DialoGPT as our pre-trained language models, we experimented with different hyperparameters and architecture, but settled on a dense classifier after the language model. Neither BERT or DialoGPT were able to achieve the same score of previous methods; only achieving weighted f_1-scores of 85.1% and 81.0% respectively, using stratified 10-fold validation on the entire dataset (where R. Fernández, Ginzburg, and Lappin (2007) got a weighted f_1-score of 92.0%). This is likely due to the scarcity of the available data, and the large class imbalance. Using model of BERT or DialoGPT with more parameters is also likely to increase (drastically) the performance, as we can see an inflation in performance when the number of parameters increases in Chapter 4.

Finally, we used models (BERT based and DialoGPT based) trained on the classification task, on training data, and calculated feature attributions on the a test set. We saw that the individual tokens of the NSU gets higher attribution scores than the tokens for the contextual information, however, the latter gets a slightly higher weighted average score. We also saw synthetic examples of how the attribution score changes, when the dialogue changes slightly. For some classes, individual tokens of the NSU got high attribution scores, where it seems that the language model have understood some properties of these classes. However, further experimentation is needed to draw any conclusions.

Chapter 4

Question Answering Benchmark

In previous chapter we tested pre-trained language models ability to classify NSUs in conversations based on text in natural language.

We will in the following chapter test the language models "understanding" of NSUs with a custom benchmark dataset. This will be another task to test the pre-trained language models ability to do inference on NSUs. Each sample in the dataset is a conversation containing one or multiple NSUs, where the intention is that these NSUs needs to be inferred in order to solve the challenge (a question). Hence, it is a benchmark for testing the ability of language models to infer NSUs. This dataset is used throughout chapter and is explained in section 4.1.

We decided to use UnifiedQA-v2 (Khashabi, Kordi, and Hajishirzi 2022) as this language model provides an easy to use format for prompting [1], and is trained on several QA datasets. This model supports a multiple-choice question-answering format so we could test a language model on the dataset. Such task will be zero-shot task, as explained in 2.2.3, since the language model is tested the on data that it has not seen during training. The only thing that is kept the same is the same format of the prompt.

4.1 NSU dialogue dataset

We will now introduce the dataset that was constructed for the task. The dataset contains only multi-turn dialogue and multiple-choice question-answering are asked to each dialogue. The dataset is not large enough to be used as training data, and hence should only be used for benchmarking. It was created mostly based on dialogue data from BNC using the refined annotated dataset by R. R.

[1] The code for the models are available at their github UnifiedQA-v2 and can be easily loaded with Transformers python package.

NSU Class	Instances
AffAns	32
Reject	13
HelpReject	11
CE	16
BareModPh	11
FactMod	4
ShortAns	29
PropMod	12
Sluice	15
Ack	30
RepAffAns	14
Filler	1
ConjFrag	12
RepAck	5
CheckQu	3

Table 4.1: Distribution of NSU instances for each class in the NSU dialogue dataset.

Fernández (2006) and DailyDialog dataset (Y. Li et al. 2017). However, it does also contain constructed dialogues, and finally, two dialogues was used from the Ubuntu Dialogue Corpus (Lowe et al. 2015). Our dataset only contain two person dialogues, but is constructed in such a way that it can contain multi-party conversations. This is because each utterance is associated with a name of the person speaking, hence we could have had dialogues between multiple people.

To cover a wide variety of NSUs, most classes were included in at least 10 different dialogues. Table 4.1 shows the distribution of NSU instances for each class in the dataset. It is important to keep in mind that a conversation can contain multiple NSUs.

Some classes, however, were dropped because they do not add a lot of information to the dialogue, such as *Ack*, *RepAck*, *CheckQu* and *Filler*. By "dropped" we don't necessarily mean that the dataset does not contain them, as it contains for example a lot of acknowledgement, but it means that questions

posed about the dialogue are not about these type of NSUs as that these classes are not the main focus.

For Ack, this is because of the of the difficulty of asking meaningful question to such dialogues e.g. if a person A has uttered a statement, and another person B acknowledge that statement by uttering a yes-word, the question posed to the dialogue would be about that statement, hence not very interesting for the NSU itself. For RepAck the same argument applies as Ack.

For CheckQu, dialogues are usually in the form of an 'Okay?' utterance, and followed by a 'yes' utterance, which would be difficult to ask a meaningful question about as well.

Lastly, for Filler, it can be discussed if it should have been included, as it was decided to include ConjFrag (which is a continuation of the antecedent by adding information). However, they were not included as they only extend the previous utterance, hence it might be difficult asking meaningful question about them. They can usually be thought of an utterance that was divided in two, but uttered by two different people. Though for some fillers where the antecedent is several turns away from the NSU, it might be interesting to add such dialogues.

Though, the dataset include conversations that have a distance between the antecedent and the NSU of greater than 1, this was not consistently kept in mind. So for future improvement, conversations that have NSUs with a distance of 2 or more to the antecedent, could be labeled. Lastly, in most dialogues we would have the occurrence of at least one, but often multiple NSUs classes (as mentioned); these are labeled for the conversation as a whole, but not for the specific utterances.

4.1.1 Multiple-Choice Question Answering

For each conversation one or multiple questions are asked about it, with associated possible answers as this is a multiple-choice QA problem. The number of available choices ranges between 3 and 5, but there are most samples containing 4 choices, as viewed in table 4.2. There is always a default choice of "Don't know", but this will only be correct in the cases where the question posed is not answerable given the dialogue. The questions posed to the dialogue are (hopefully always) asking about some of the NSUs. This means we would have to infer the meaning of the NSUs given the previous utterances in the dialogue, and the posed question will ask about these.

The available answers have to be non-ambiguous, in the sense that there should

# choices	# sample
3	15
4	95
5	9
Total	119

Table 4.2: Shows the number of available choices and the associated number of samples.

always only be one "most correct" answer, even though some of the other choices might seem correct, to some extent. If there are two (or more) answers that to a human feels equally likely to be correct, either the conversation, the question posed or the available answers are bad and should be removed or changed.

Also, the conversation needs to contain enough different information, so it makes it a bit more difficult to answer the question. For example if the dialogue only contains one noun and the question posed about the dialogue is asking about the noun, it would be too easy for a language model to exclude the possible choices that are mentioning other nouns. This is because they have not occurred in the conversation at all.

Hence, that dialogue should try to mention other nouns to make it more difficult and meaningful to answer questions posed to such dialogue. However, the point of these dialogues is not to ask difficult question or have a very complex dialogue, but rather to see if a language model is able to infer the meaning of the NSUs, which to humans should be non-problematic. This is at least the methodology that was tried to follow when creating this dataset.

4.1.2 Dialogue example

One important thing to be aware of is that this dataset, although dialogues from the real world have been used, is artificial. This is because a lot of the dialogues were reused with handcrafted changes to fall within another class of NSUs. This is done intentionally to test the language models on different types of NSU classes as the utterances changes slightly, to see if they understand how the meaning of the utterance when the dialogue changes. If we take a look at the following two almost identical dialogues:

(7) a. Melissa: Anne said she worked in a factory at one time.
    b. Ashley: Popcorn factory?

c. Melissa: Yes.
d. Ashley: And she drove to work?
e. Melissa: Yes, she owned a car.

(8) a. Melissa: Anne said she worked in a factory at one time.
b. Ashley: Popcorn factory?
c. Melissa: No, car.
d. Ashley: I see.

Above we can see two dialogues that starts the same, and then diverges on the third utterance (c). We observe how the meaning changes depending on the answer of *Melissa*. *Ashley* wants some clarification about the first utterance (a); specifically, if the factory actually was a *popcorn factory*. Hence, the first utterance of *Ashley* (b) is a CE, as she wants clarification of the antecedent. In dialogue 7 we can see that *Melissa* answers "Yes. " (7c), and we understand that *Anne* worked at a popcorn factory. In dialogue 8, however, *Melissa* answers with "No, car. " (8c). This is a negative answer question (8b) and is accompanied with a contrasting alternative, hence it is classified as a Helpful Reject. We can from this utterance infer that *Anne* was working at a *car factory* and not a popcorn factory.

These reason for including the utterances 7d and 7e in dialogue 7, is because we want the word 'car' to be included, as an additional subject to included in the conversation. This is to make it more difficult for the language model (though not)

A possible multiple-choice question to these dialogues can be *"What type of factory did Anne work in?"* with the associated possible answers:

(a) Don't know
(b) Popcorn factory
(c) Car factory
(d) Anne worked at a factory

We can also see that the dialogues contain an associated speaker, in form of a real name. This was chosen instead of using alphabetical names (A, B, C etc.), to try and help the language models with a conversation from a real world scenario. Both was tested, and there no difference in the accuracy either way.

4.1.3 Text communication

In our dataset there are only two conversations that are in form of text messaging. Due to the time consuming task of finding, preparing and classifying these, most

of the conversations are from the NSU corpus (2.1.4) where dialogues with NSUs has already been found and classified. However, to further include a wider variety of dialogues, and especially in the form of text messaging dialogues, it would have interesting to add more data from corpus like Ubuntu Dialogue Corpus or from other chatting services like Discord. As we would occasionally have other types of NSUs in text messages that is will not occur in oral chit-chat. For some NSU classes there are also NSUs that occur in oral dialogue will not occur in text message dialogue. As an example we can have a look at a NSU class that for some NSU will only occur in oral dialogue. If we consider the NSU class CE we can see that text message we would not ask for the previous message to be repeated the exact same way, as a person might do when mishearing (or not completely hear) an utterance. This is, of course, since a person can read a text message multiple times. A CE in text form can instead be a word that was misspelled and another person would ask for clarification about that, if not understood. It can also be that an utterance was written too short and simply need further clarification. Sometimes it can also be in the same form as in oral dialogue e.g. when one person asks for further clarification.

4.2 Experimental setup

In order to answer to prompt a model with a dialogue and ask a question to that dialogue, we will be using the UnifiedQA-v2 model (Khashabi, Kordi, and Hajishirzi 2022). The UnifiedQA-v2 model is using the T5 architecture, but is trained on datasets in 4 QA formats. It is important to emphasize that this is a model that has only been pre-trained and has not been fine-tuned on any samples of the dataset. Hence, for these models this is a *zero-shot* challenge. To make it easier, we will often time refer to a model by its size, hence *small* for the "UnifiedQA-v2-t5-small" model and *base*, *large*, *3B*, and *11B* for the larger models respectively.

When a data sample is given to the model, it needs to be in a specified format. First comes the question, then the possible choices, and finally the text, which in our case is a prefix followed by the conversation. All text have to be lower cased. The question, choices and text are separated by "\n"; this is not the newline character, but a backslash followed by the letter n. This is exemplified by the prompt below:

who left? \n (a) don't know (b) john (c) mark (d) rod \n
the following is a dialogue between rod and mark:
rod: did john leave?
mark: yes.

We can see that the the model in this case gets the information of which person spoke the utterance, unlike in Chapter 3. The model then generates an output text in natural language, which we match against the possible choices. If the generated text is equal to the correct answer, the model is considered to have selected the right choice.

4.2.1 Random baseline

We will compare the performance to a simple randomized baseline, to see the accuracy of choosing answers at random. The random baseline can be calculated using the number of available choices $K_c = \{k_{c,1}, ..., k_{c,i}\} = \{3, 4, 5\}$, where $i = 3$ is the amount of different available choices. Each choice has an associated amount of samples $K_s = \{k_{s,1}, ..., k_{s,i}\} = \{15, 95, 9\}$, as seen in table 4.2, where $N = 119$ is the total amount of samples. The probability of answering correct for an amount n_c of available choices is $P(\text{True}) = \frac{1}{n_c}$. We can take the weighted mean to get the baseline:

$$\begin{aligned} \text{Baseline} &= \sum_{i=1}^{k} \frac{1}{k_{c,i}} \times \frac{k_{s,i}}{N} \\ &\approx 0.257 \end{aligned}$$

We can see that the more available choices we have for a question, the harder it is for a model to answer correctly by chance.

4.2.2 User study

In order to test the understandability of the conversations, with question posed along with the available answers in the dataset, two persons were told to go through the entirety of it and answer all questions. The goal was to test wether the average humans were able to understand the content of the dialogue, and then answer the question posed about the dialogue, given some choices. These people are not affiliated to informatics or to NLP so they could represent the average human. Though none of them were native English speakers, both knew

English at a relatively high level. The ambition of this conversation benchmark dataset, is to create it easy enough for the average human to be able to answer all questions correctly, without needing to be told any specific background information.

The users were only told to answer the question posed to a dialogue, and select the answer they thought were the *most* correct from the available choices. A prompt to a human user looked like the following:

```
The following is a dialogue between Sarah and Lisa:
Sarah: Alistair he's, he's made himself ehm he has made himself coordinator.
Lisa: And section engineer.
Sarah: And section engineer, yes.

What has Alistair made himself as?
(a) Both coordinator and section engineer
(b) Coordinator
(c) Don't know
(d) Section engineer

Input [a/b/c/d]:
```

To remove possible consistent bias in positioning of the correct answer, the order of the choices were randomized for each question. The order of the dialogues were also randomized to remove possible similar dialogues appearing after one another. This was done so there should be as little as possible bias in the order of either dialogues and choices. Of course, a seed was used for reproducibility when randomizing the order. This made it possible to retrieve the correct place of the predicted choice for the current dialogue. Also, the user were not given any indication of whether an answer was correct after choosing one.

In DailyDialog, and for some in BNC, there are oral dialogues. Some of these might be unlikely to happen in written text messages and can occasionally be confusing. This might be since we are not hearing how a person speak the utterance i.e. the tone of a speaker. As the person might be unsure about something, or very confident and so on, which is not reflected in the transcribed text of the oral dialogue. Some of these might be Helpful Reject, but where the person is not saying a negation beforehand. If we consider the dialogue below:

A: When we increase the resistance, we make more resistance.
B: **Less current.**

In an oral conversation B might be denying this while shaking the head

sideways, which in turn helps A understand that the statement uttered was wrong. For a conversation in text form, this might be clearer when adding "No" before; becoming "No, less current".

User Study Accuracy

Both users were able to answer 111 of 119 correct, which gives an accuracy of **0.937**. However, none of the dialogues which the individual users answered wrong were in common. This mean, if D is the set of all the dialogues in the dataset, $A_{\text{false}} \subset D$ are the dialogues where user A answered the question wrong, and $B_{\text{false}} \subset D$ are the dialogues where user B answered wrong, then $A_{\text{false}} \cap B_{\text{false}} = \varnothing$. So the errors done by these humans might be related to human fatigue or drop in the ability to concentrate over time. A month after first going through the dataset, user A where presented the dialogues A_{false}, and were asked to do the same task. This time the user was able to answer the question posed to the dialogues correctly.

4.3 Results and Discussion

We will now view the results of the UnifiedQA model with different number of parameters. Table 4.3 shows the accuracies of the models along with their number of parameters[2] and this is plotted in figure 4.1.

In general we can see that the models containing more parameters are able to perform reasonably well when taking into account that this is in a zero-shot setting. However, the small and base models are struggling with accuracies of only 49.6% and 58.0% respectively. These models might not be able to capture the meaning of NSUs, especially if they contain more distance between them, but this needs further experimentation. On the other hand, the accuracy between the best performing model, 11B, and the human accuracy is only a difference of 6.3%. This means that these larger models are somewhat able infer the meaning of NSUs, since the conversations are all containing a lot of NSUs and the question posed to these dialogues are (usually) questioning about something that must be inferred from a NSU.

We can see the accuracy improves significantly as the number of parameters increases. There is an improvement of 8.4% in accuracy from the small model to the base model. For comparison of parameters of models, in Chapter 3, we used

[2] All models are using the checkpoint 1363200.

Model name	# parameters	# correct	Accuracy
baseline (random)			0.257
unifiedqa-v2-t5-small	6.062×10^7	59	0.496
unifiedqa-v2-t5-base	2.232×10^8	69	0.580
unifiedqa-v2-t5-large	7.384×10^8	93	0.782
unifiedqa-v2-t5-3b	2.854×10^9	100	0.840
unifiedqa-v2-t5-11b	1.131×10^{10}	104	0.874
human		111	0.937

Table 4.3: Shows the number of parameters and accuracy for each model on the benchmark NSU QA dataset, along with the accuracy of the user study.

BERT (base) and DialoGPT (small), only containing containing about 110 million and 117 million parameters respectively, versus the base model of T5 that contains almost twice as much, at about 220 million parameters. However, the most significant jump in performance is from the base model to the large model; which is an improvement of 20.2% in accuracy. The large model, however, contains about 740 million parameters.

A notable thing is that a couple of times the models did not answer any of the available choices. This means that a model generated an answer which was not among any of the available choices. When this happened, the answer the model gave was automatically considered false.

However, for one example, in the 11b-model, the generated prediction was synonym to the correct text. The generated answer was "six and seven oclock", where the actual answer was "six and seven o'clock". The only difference was an apostrophe, and hence the generated answer was automatically categorized false (these models can generate apostrophes, as seen when they generate "don't know"). If we adjust the performance for this instance the UnifiedQA-v2 model with 11 Billion parameters for an accuracy of $\frac{105}{119} \approx 0.882$.

4.3.1 Comparing different model sizes

As we can see from table 4.3 the largest model indeed got the most answers correct, with a total of 104/119. This is a drastic increase in performance from the model containing the least amount of parameters, only answering 59/119 correct. We will now see some examples of where the models answered both

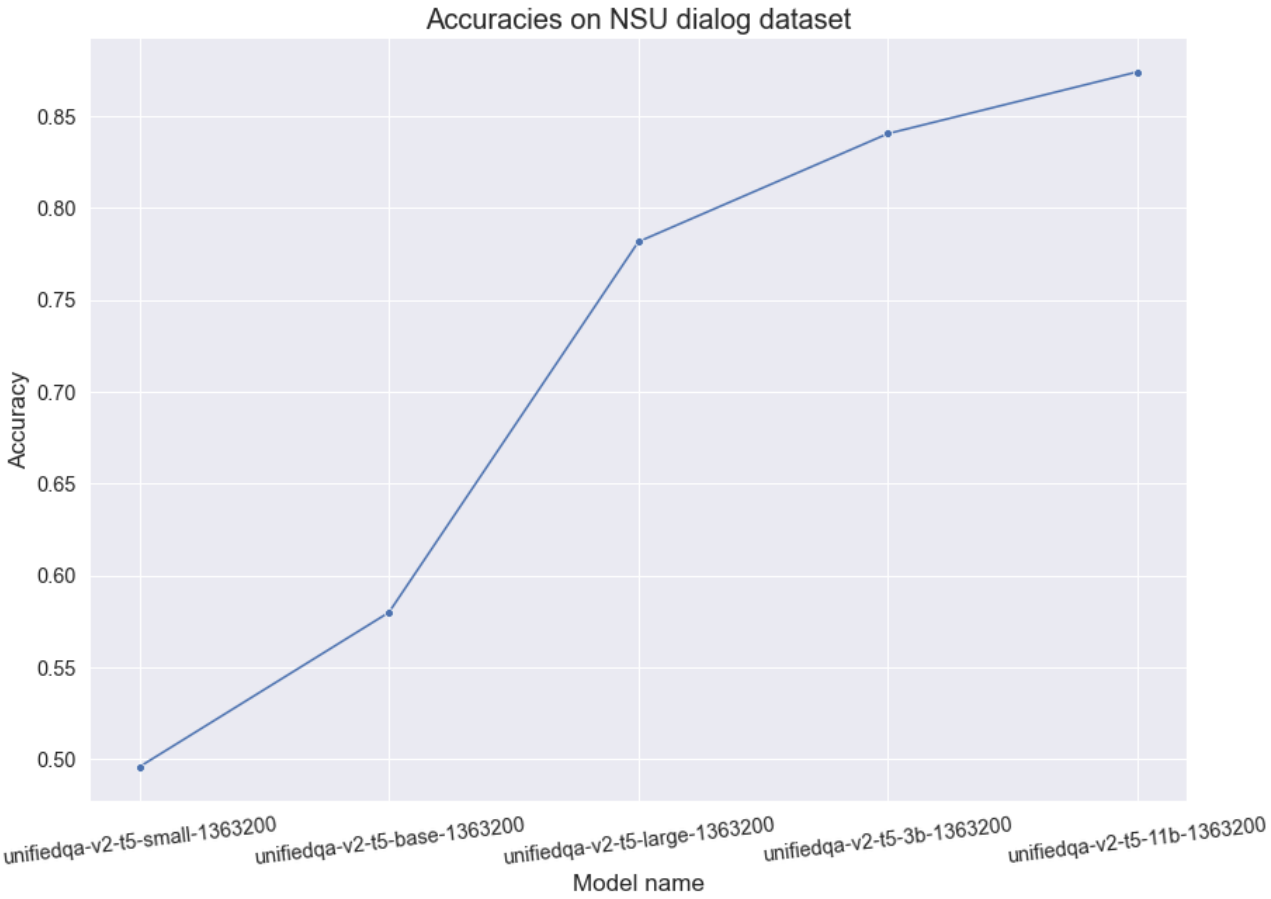


Figure 4.1: Shows the how accuracies improve with the increase in model size

wrongly and correctly.

If we consider a sample from the dataset in example 9 below:

(9) The following is a dialogue between David and Susan:

David: Would you like some more wine?
Susan: Wine?
David: Yes.
Susan: No.
David: Are you sure?
Susan: Yes.

Does Susan want some more wine?
(a) Don't know (b) Yes (c) No

We can see infer from the conversation (9) that the answer to the question posed is "No". *Susan* want some clarification in the beginning of the first question *David*

posed, but then is answering the question by rejecting. The antecedent to the utterance "No", which is *Davids* first question, is of distance 3 from the NSU. And between these utterances, there is and utterance of "Yes", which might confuse the language models. In addition to this, the second question of *David* is answered with "Yes", which means that the language model cannot use the latest utterance. However, only the small model answered wrongly with "Yes", and all other models were able to answer correctly. This means they must have understood that *Susans* utterance "No", was referencing the first question.

There was in total 4 samples that the 11B model answered correctly and where all of the other models answered wrong. Below in example 10 we show a sample conversation. We can see that *Roberts* answer is a Helpful Reject, however, it does not contain any negation in front, making it more difficult to infer. This type of conversation is also harder to understand in transcribed format, compared to an oral conversation where a person might shake the head while answering. Example 11 show another instance, where the utterance of *Teresa* is of a Helpful Reject. Here, the 11B model is again able to infer that the NSU is a correction of the previous statement.

(10) The following is a dialogue between Rod and Robert:

Rod: Did John leave?
Robert: Mark.

Who left?
(a) Don't know (b) John (c) Mark (d) Rod

(11) The following is a dialogue between Gerald and Teresa:

Gerald: And it's called enhanced careers guidance.
Teresa: Individual careers guidance, actually.
Gerald: I see.

What is it called?
(a) Don't know (b) Enhanced careers guidance (c) Individual careers guidance (d) I see

There were a couple of instances where none of the models were able to answer correctly; 6 in total. A sample is viewed in example 12.

(12) The following is a dialogue between Rod and Mark:

Rod: Did John leave?

Mark: John?

Who left?
(a) Don't know (b) John (c) Mark (d) Rod

In example 12, all the models answered "John", even though the conclusion cannot be drawn the the conversation. Hence, the answer should be "Don't know". Being overly confident of an answer, was usually more apparent with the models containing less parameters. If we count the instances of where an answer was "Don't know", but a model answered another choice the following values are counted; 6 for small, 7 for base, 3 for large, 4 for 3B, and 1 for 11B.

However, we number of times "Don't know" is answered, when it is not the correct answer, seems to increase with the number of parameters, with the following counts: 0 for small, 0 for base, 1 for large, 1 for 3b, and 5 for 11b. This means that 5/15 of the answers that the 11B model answer incorrectly was the choice "Don't know". So it seems that in general the models get less confident with the increase of parameters.

However, in some cases the models with less parameters were able to answer correct while the 11B-model answer wrong. The instances the the models with less parameters answered correct and the 11B answered wrong was the following; 1 for small, 2 for base, 5 for large, and 8 for 3B.

4.4 Summary

In this chapter we have presented a benchmark dataset to test the language models inference on NSU. We explained the methodology used when creating the dataset and showed examples of samples containing conversations, questions posed to the conversation with available associated choices. The dataset was tested on humans that are not affiliated to informatics or NLP, and both users achieved an accuracy of 93.7%.

This challenge is a zero-shot challenge, where we used T5 models trained on the UnifiedQA format to test the performance on conversational data they had not seen before. We showed that the models with the lowest amount of parameters, small and base, achieved poor results on the dataset, with accuracies of only 49.6% and 58.0% respectively. This means that these models do not seem to be able to infer the meaning of NSUs in, what we can assume are more complex conversations. However, the largest models achieved impressive results, where the model containing 11B parameters, got an accuracy of 87.4% which is only

6.3% lower than the accuracy of the user study. This means that the larger models are able to understand the meaning of NSUs to a reasonable extent.

While this benchmarking dataset might challenge the models to some certain extent, we believe it needs a more variety of conversations, especially in text messaging format, as well as improved labeling that which utterances that are NSUs, and their associated antecedent. We believe it would also be interesting to combine this dataset with feature attribution methods as shown in Chapter 3 for further analyzing which utterances are the most important in a conversation for answering the question at hand. Finally, this benchmarking dataset will be available for further research and experimentation, and can be used to test a language models ability to infer NSUs in conversations.

Chapter 5

Conclusion

In this thesis we have performed experiments with non-sentential utterances to try and analyze a language model's ability to infer such utterances in conversations. We started by presenting what NSUs are; utterances in the form of incomplete sentences, which convey the full meaning of a full grammatically correct sentence, when inferred from the surrounding context.

Classification of NSUs using transformer language models

We presented in Chapter 2 the frequent occurrences of NSUs in dialogue and why this motivates research of these utterances. As for any dialogue system, such utterances need to be understood for natural and coherent conversations with humans. We continued by explaining the architecture of pre-trained transformer language models, such as BERT and GPT, that were used for experimentation in later chapters. As the transformer architecture has become state-of-the-art in most NLP tasks we want to test their ability to interpret NSUs in conversations. We also introduced some methods for interpreting language models, such as feature attribution and probing tasks.

A natural task for testing how well different NSUs can be distinguished from each other is by classifying NSUs, which was done in Chapter 3. We used the taxonomy of NSUs created by R. R. Fernández (2006), which contains a total of 15 classes. Prior to this work, and helping to form this taxonomy, a corpus study of fragments done by Fernandez and Ginzburg (2002) and later extended by R. R. Fernández (2006), created the corpus we used in the classification task, containing 1283 labeled NSUs samples from the British National Corpus (BNC).

Unlike previous classification methods (R. Fernández, Ginzburg, and Lappin 2007; Dragone and Lison 2016), we used raw text as input for our models, instead of extracted features from the dialogue. We also used dialogues from the corpus

where the antecedent was more than 1 turn away from the NSU, which differs from previous methods and makes this classification of NSUs more complicated.

Our experiment showed that language models are indeed able to capture linguistic properties of the contextual information and the NSU, allowing them to classify NSUs to some extent. We showed that a simple baseline model, using non-contextualized word embeddings, was not able to classify NSUs, and only converged to the majority class. Using stratified 10-fold cross validation, the classification model using BERT as its pre-trained language model, we were able to get a weighted f_1-score of 85.1%, and for DialoGPT the f_1-score was 81.0%. These results are both worse than previous methods, which were able to reach f_1-scores of 92.0% (R. Fernández, Ginzburg, and Lappin 2007) and 90.5% (Dragone and Lison 2016). However, the pre-trained models used contain "only" around 110-117 Million parameters. We can expect a jump in performance when using larger language models i.e. around 1 Billion parameters, or more, as we saw a significant jump in performance when increasing the size of the model in Chapter 4.

Feature attribution

Chapter 3 also used feature attribution to analyze contribution of the contextual information and the NSU, when classifying NSU in dialogues. We used models trained on the classification task, on a training set, and calculated the attribution scores for the test set. All feature attributions were calculated with respect to the true class label. We then found that in general the individual tokens in the NSU got higher scores than the individual tokens in the contextual information, which was true for both BERT and DialoGPT. However, the weighted average of the attribution scores for the contextual information was slightly higher than for the NSU. This is probably a consequence of the contextual information often containing a (much) higher number of tokens than the NSU, hence contributing more as a whole to the output, than the NSU.

We then saw examples of synthetic dialogues with small changes, to see how the feature attribution changes to small changes in dialogue. It seems like the language models have picked some specific words and symbols, and give these a higher attribution score when classifying them. However, these examples need to be studied in more detail e.g. by using additional examples for each class.

Question Answering NSU dataset

In Chapter 4, we introduced a benchmark dataset to test language models inference of NSUs in conversations. The dataset is a multiple-choice Question Answering dataset, where question posed in the context of a conversation has multiple associated answers, and the task is to select the correct one. The presented dataset will be available for further research and experimentation. This dataset was mostly based on conversations from the NSU corpus (BNC) (R. R. Fernández 2006), but also the DailyDialog corpus (Y. Li et al. 2017), as well two text conversation from the Ubuntu Dialogue Corpus (Lowe et al. 2015). Most of the conversations have been modified to some extent, and reused so they fall within another class of NSUs. The dataset hence contains a wide variety of NSUs, but can also be extended to contain more text messaging dialogues in the future.

In order to test how well the it is understood by humans, a user study was performed on the dataset. The user study resulted in an accuracy of 93.7% was achieved.

We tested the dataset using T5 language models of different sizes, trained on datasets on the UnifiedQA format. There were 5 models of different sizes ranging from about 60 Million to 11 Billion parameters. We saw a drastic increase in performance as the number of parameters increased, where the smallest model achieved an accuracy of 49.6% and the largest model got an accuracy of 87.4%, which is only 6.3% lower than the accuracy achieved by humans in the user study. These results suggest that the large models are able to infer the meaning of NSUs in conversations to some extent.

5.1 Future work

The experiments presented in this thesis can be extended in multiple directions.

For the classification task, it would be interesting to see how larger language models perform, in order to see how much improvement we could get when increasing the number of parameters. A text-to-text language model trained on prompts, such as T5, can also be used to see if there is an increase in performance compared to the language models used in this thesis.

In this thesis we only scratched the surface of what could be done using feature attribution for explaining the models. In multi-turn conversation we could find the attribution scores for each turn when classifying a NSU, to see if some utterances get higher scores than others. Then, we might be able to see if the language model can capture some relationship between the antecedent and the NSU in conversations. This is especially interesting for conversations where

the antecedent is more than 1 distance away from the NSU. However, one could also include previous dialogue utterance before the antecedent and compare the attribution scores of these to the antecedent.

Feature attribution could also be used on the UnifiedQA models for the benchmark NSU dataset created in this thesis. When a question posed in the context of a conversation requires the NSU to be inferred (for answering the question correct), one could analyze the attribution scores from the model to see if it is attributing high scores to the NSU and the antecedent. If the model yields high attribution scores to the NSU and antecedent, it might mean that it is inferring the NSU from the given context. However, if there are some other utterances getting high attribution scores, it might mean that it is able to answer the question without inferring the NSU.

Another method for analyzing language models is to create a *probing task*. This probing task could simply be the classification of NSUs as performed in Chapter 3. However, we could analyze individual layers of a language model, by freezing the model and use the embeddings from intermediate layers when the probe is classifying NSUs. By doing this we might be able to find which layers that encodes most of the information needed in order to classify NSUs.

Finally, the benchmark NSU dataset presented in Chapter 4 can be extended by adding additional labels for which utterances contain NSUs, as well as their associated antecedent. It could also be interesting to include more conversation in the form of text messaging, as these are likely to give rise to other forms of NSUs.

Appendix A

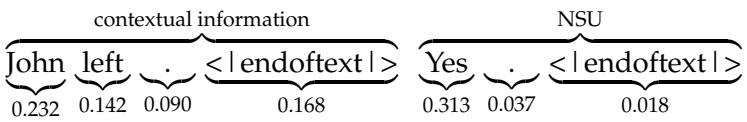
Classification and Feature Attribution

This appendix shows additional results from the classification task and feature attribution task in 3.

Feature Attribution

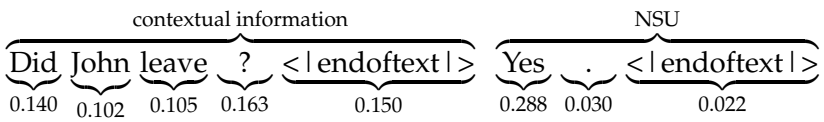
The following shows the attribution scores for DialoGPT with a dense classifier, using Integrated Gradients. It was trained on the NSU training data (3.1.1). A separation character, "Ġ", which is put in front of a words separated with spaces, has been removed for easier readability.

Ack



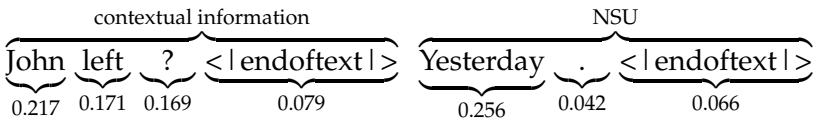
(A.1)

Ack



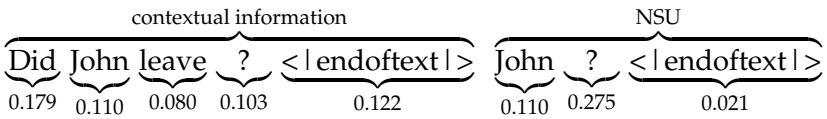
(A.2)

BareModPh



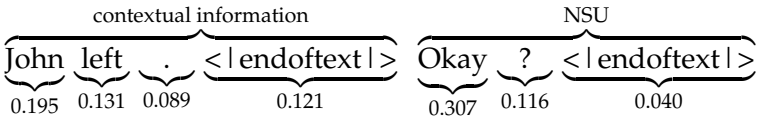
(A.3)

CE



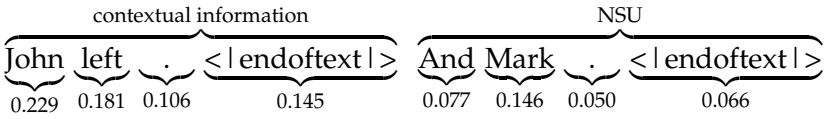
(A.4)

CheckQu



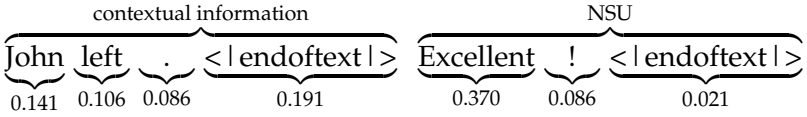
(A.5)

ConjFrag



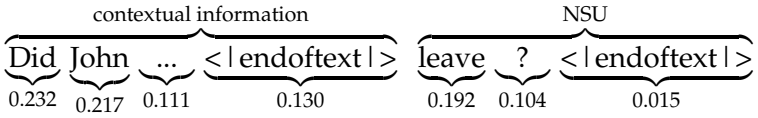
(A.6)

FactMod



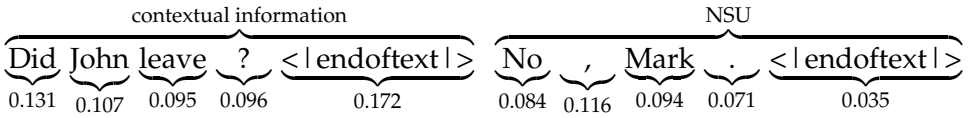
(A.7)

Filler



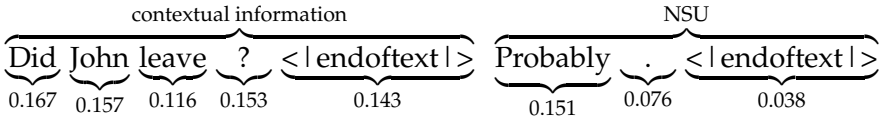
(A.8)

HelpReject



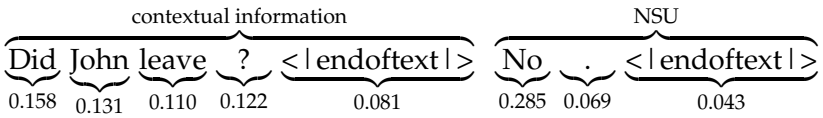
(A.9)

PropMod



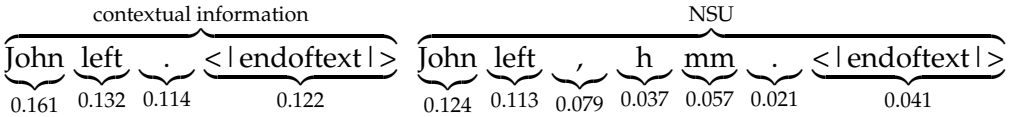
(A.10)

Reject



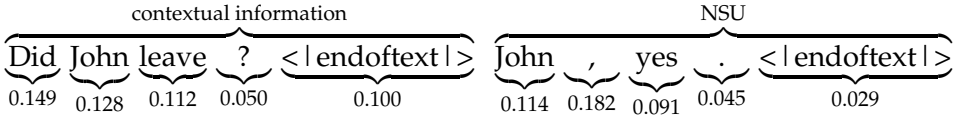
(A.11)

RepAck



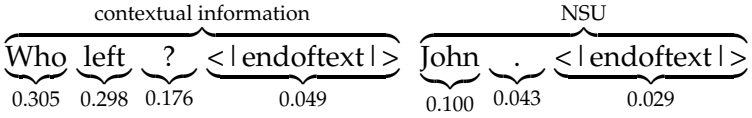
(A.12)

RepAffAns



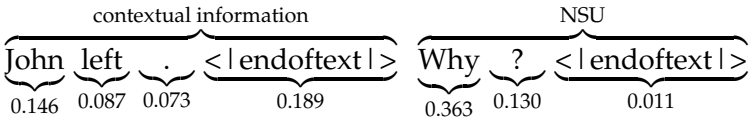
(A.13)

ShortAns



(A.14)

Sluice



(A.15)

Bibliography

[1] Asad Abdi, Norisma Idris, and Zahrah Ahmad. "QAPD: an ontology-based question answering system in the physics domain". eng. In: *Soft Computing* 22.1 (2016), pp. 213–230. ISSN: 1432-7643.

[2] Guillaume Alain and Yoshua Bengio. *Understanding intermediate layers using linear classifier probes*. 2018. arXiv: 1610.01644 [stat.ML]. URL: https://arxiv.org/abs/1610.01644.

[3] Nicholas Asher. *Logics of conversation*. eng. Cambridge, 2003.

[4] Sebastian Bach et al. "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation". In: *PLOS ONE* 10.7 (July 2015), pp. 1–46. DOI: 10.1371/journal.pone.0130140. URL: https://doi.org/10.1371/journal.pone.0130140.

[5] Stephen Bach et al. "PromptSource: An Integrated Development Environment and Repository for Natural Language Prompts". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 93–104. URL: https://aclanthology.org/2022.acl-demo.9.

[6] David Baehrens et al. "How to Explain Individual Classification Decisions". In: *Journal of Machine Learning Research* 11.61 (2010), pp. 1803–1831. URL: http://jmlr.org/papers/v11/baehrens10a.html.

[7] Ellen L Barton. *Nonsentential constituents : a theory of grammatical structure and pragmatic interpretation*. eng. Amsterdam, 1990.

[8] Yonatan Belinkov. *Probing Classifiers: Promises, Shortcomings, and Alternatives*. 2021. arXiv: 2102.12452 [cs.CL]. URL: https://arxiv.org/abs/2102.12452.

[9] Emily M. Bender and Alexander Koller. "Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 5185–

5198. DOI: 10.18653/v1/2020.acl-main.463. URL: https://www.aclweb.org/anthology/2020.acl-main.463.

[10] Leo Breiman. "Random Forests". In: *Machine Learning* 1 (Oct. 2001), pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[11] A. B. Brown. "A Proof of the Lebesgue Condition for Riemann Integrability". In: *The American Mathematical Monthly* 43.7 (1936), pp. 396–398. DOI: 10.1080/00029890.1936.11987865. eprint: https://doi.org/10.1080/00029890.1936.11987865. URL: https://doi.org/10.1080/00029890.1936.11987865.

[12] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. DOI: 10.48550/ARXIV.2005.14165. URL: https://arxiv.org/abs/2005.14165.

[13] Lou Burnard. *Reference Guide for the British National Corpus (World Edition)*. Oxford Universtity Computing Services. 2000. URL: http://www.natcorp.ox.ac.uk.

[14] Yong-gang Cao et al. "Automatically extracting information needs from complex clinical questions". In: *Journal of Biomedical Informatics*. Vol. 43. 6. 2010, pp. 962–971. DOI: https://doi.org/10.1016/j.jbi.2010.07.007.

[15] Sandra Carberry. *Plan recognition in natural language dialogue*. eng. Cambridge, Mass, 1990.

[16] Yulong Chen et al. *AdaPrompt: Adaptive Model Training for Prompt-based NLP*. 2022. DOI: 10.48550/ARXIV.2202.04824. URL: https://arxiv.org/abs/2202.04824.

[17] Eunsol Choi et al. "QuAC: Question Answering in Context". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 2174–2184. DOI: 10.18653/v1/D18-1241. URL: https://www.aclweb.org/anthology/D18-1241.

[18] William W. Cohen and Yoram Singer. "A Simple, Fast, and Effective Rule Learner". In: *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*. AAAI '99/IAAI '99. Orlando, Florida, USA: American Association for Artificial Intelligence, 1999, pp. 335–342. ISBN: 0262511061.

[19] Alexis Conneau, Douwe Kiela, et al. “Supervised Learning of Universal Sentence Representations from Natural Language Inference Data”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 670–680. DOI: 10.18653/v1/D17-1070. URL: https://aclanthology.org/D17-1070.

[20] Alexis Conneau, German Kruszewski, et al. “What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 2126–2136. DOI: 10.18653/v1/P18-1198. URL: https://www.aclweb.org/anthology/P18-1198.

[21] Cristian Danescu-Niculescu-Mizil, Michael Gamon, and Susan Dumais. “Mark my words!” In: *Proceedings of the 20th international conference on World wide web - WWW '11*. ACM Press, 2011. DOI: 10.1145/1963405.1963509. URL: https://doi.org/10.1145%2F1963405.1963509.

[22] Poulami Debnath, Shubhashis Sengupta, and Harshawardhan M Wabgaonkar. “Identifying, Classifying and Resolving Non-Sentential Utterances in Customer Support Systems”. In: (2018). URL: http://semdial.org/anthology/Z18-Debnath_semdial_0006.pdf.

[23] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://www.aclweb.org/anthology/N19-1423.

[24] Paolo Dragone and Pierre Lison. “Classification and Resolution of Non-Sentential Utterances in Dialogue”. In: *Italian Journal of Computational Linguistics* 2.1 (2016), pp. 45–62. URL: http://paolodragone.com/files/pdf/dragone2016classification.pdf.

[25] Raquel Fernandez and Jonathan Ginzburg. “Non-Sentential Utterances in Dialogue: A: Corpus-Based Study”. In: *Proceedings of the Third SIGdial Workshop on Discourse and Dialogue*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 15–26. DOI: 10.3115/1118121.1118124. URL: https://www.aclweb.org/anthology/W02-0203.

[26] Raquel Fernández, Jonathan Ginzburg, and Shalom Lappin. *Classifying Non-Sentential Utterances in Dialogue: A Machine Learning Approach*. 2007. DOI: 10.1162/coli.2007.33.3.397. URL: https://www.aclweb.org/anthology/J07-3005.

[27] Raquel Rovira Fernández. "Non-sentential utterances in dialogue: Classification, resolution and use". PhD thesis. King's College London, 2006. URL: https://staff.fnwi.uva.nl/r.fernandezrovira/papers/RaquelFernandez_PhD.pdf.

[28] Jonathan Ginzburg and Ivan A. Sag. *Interrogative investigations : the form, meaning, and use of English interrogatives*. eng. Stanford, Calif, 2000.

[29] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, May 2010, pp. 249–256. URL: https://proceedings.mlr.press/v9/glorot10a.html.

[30] Bert F. Green et al. "Baseball: An Automatic Question-Answerer". In: *Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference*. IRE-AIEE-ACM '61 (Western). Los Angeles, California: Association for Computing Machinery, 1961, pp. 219–224. ISBN: 9781450378727. DOI: 10.1145/1460690.1460714. URL: https://doi.org/10.1145/1460690.1460714.

[31] Riccardo Guidotti et al. "A Survey of Methods for Explaining Black Box Models". In: *ACM Comput. Surv.* 51.5 (Aug. 2018). ISSN: 0360-0300. DOI: 10.1145/3236009. URL: https://doi.org/10.1145/3236009.

[32] Somil Gupta, Bhanu Pratap Singh Rawat, and Hong Yu. "Conversational Machine Comprehension: a Literature Review". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 2739–2753. DOI: 10.18653/v1/2020.coling-main.247. URL: https://www.aclweb.org/anthology/2020.coling-main.247.

[33] Sanda Harabagiu, Steven Maiorano, and Marius Pasca. "Open-domain textual question answering techniques". In: *Natural Language Engineering* 9 (Sept. 2003), pp. 231–267. DOI: 10.1017/S1351324903003176.

[34] Karl Moritz Hermann et al. “Teaching Machines to Read and Comprehend”. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS’15. Montreal, Canada: MIT Press, 2015, pp. 1693–1701.

[35] John Hewitt and Percy Liang. “Designing and Interpreting Probes with Control Tasks”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 2733–2743. DOI: 10.18653/v1/D19-1275. URL: https://www.aclweb.org/anthology/D19-1275.

[36] John Hewitt and Christopher D. Manning. “A Structural Probe for Finding Syntax in Word Representations”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4129–4138. DOI: 10.18653/v1/N19-1419. URL: https://www.aclweb.org/anthology/N19-1419.

[37] Minlie Huang, Xiaoyan Zhu, and Jianfeng Gao. *Challenges in Building Intelligent Open-domain Dialog Systems*. 2020. arXiv: 1905.05709 [cs.CL]. URL: https://arxiv.org/abs/1905.05709.

[38] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. “What Does BERT Learn about the Structure of Language?” In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 3651–3657. DOI: 10.18653/v1/P19-1356. URL: https://www.aclweb.org/anthology/P19-1356.

[39] Ginzburg Jonathan. *The Interactive Stance*. Oxford University Press, 2012.

[40] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. “A Convolutional Neural Network for Modelling Sentences”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 655–665. DOI: 10.3115/v1/P14-1062. URL: https://aclanthology.org/P14-1062.

[41] Daniel Khashabi, Yeganeh Kordi, and Hannaneh Hajishirzi. *UnifiedQA-v2: Stronger Generalization via Broader Cross-Format Training*. 2022. DOI: 10.48550/ARXIV.2202.12359. URL: https://arxiv.org/abs/2202.12359.

[42] Daniel Khashabi, Sewon Min, et al. "UNIFIEDQA: Crossing Format Boundaries with a Single QA System". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 1896–1907. DOI: 10.18653/v1/2020.findings-emnlp.171. URL: https://aclanthology.org/2020.findings-emnlp.171.

[43] Narine Kokhlikyan et al. *Captum: A unified and generic model interpretability library for PyTorch*. 2020. DOI: 10.48550/ARXIV.2009.07896. URL: https://arxiv.org/abs/2009.07896.

[44] Oleksandr Kolomiyets and Marie-Francine Moens. "A survey on question answering technology from an information retrieval perspective". In: *Information Sciences* 181.24 (2011), pp. 5412–5434. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2011.07.047. URL: https://www.sciencedirect.com/science/article/pii/S0020025511003860.

[45] Vineet Kumar and Sachindra Joshi. "Non-sentential Question Resolution using Sequence to Sequence Learning". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 2022–2031. URL: https://www.aclweb.org/anthology/C16-1190.

[46] Teven Le Scao and Alexander Rush. "How many data points is a prompt worth?" In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, June 2021, pp. 2627–2636. DOI: 10.18653/v1/2021.naacl-main.208. URL: https://aclanthology.org/2021.naacl-main.208.

[47] Geoffrey Leech, Roger Garside, and Michael Bryant. "CLAWS4: The Tagging of the British National Corpus". In: *COLING 1994 Volume 1: The 15th International Conference on Computational Linguistics*. Kyoto, Japan, Aug. 1994. URL: https://aclanthology.org/C94-1103.

[48] Jiaqi Li et al. "Molweni: A Challenge Multiparty Dialogues-based Machine Reading Comprehension Dataset with Discourse Structure". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 2642–2652. DOI: 10.18653/v1/2020.coling-main.238. URL: https://aclanthology.org/2020.coling-main.238.

[49] Xiaoya Li et al. "Dice Loss for Data-imbalanced NLP Tasks". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020,

pp. 465–476. DOI: 10.18653/v1/2020.acl-main.45. URL: https://aclanthology.org/2020.acl-main.45.

[50] Yanran Li et al. "DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset". In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Taipei, Taiwan: Asian Federation of Natural Language Processing, Nov. 2017, pp. 986–995. URL: https://www.aclweb.org/anthology/I17-1099.

[51] Pierre Lison and Jörg Tiedemann. "OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Portoro, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 923–929. URL: https://aclanthology.org/L16-1147.

[52] Pengfei Liu et al. *Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing*. 2021. DOI: 10.48550/ARXIV.2107.13586. URL: https://arxiv.org/abs/2107.13586.

[53] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG].

[54] Ryan Lowe et al. "The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems". In: *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Prague, Czech Republic: Association for Computational Linguistics, Sept. 2015, pp. 285–294. DOI: 10.18653/v1/W15-4640. URL: https://aclanthology.org/W15-4640.

[55] William Merrill et al. *Provable Limitations of Acquiring Meaning from Ungrounded Form: What will Future Language Models Understand?* 2021. arXiv: 2104.10809 [cs.CL].

[56] Julian Michael. *To dissect an octopus: Making sense of the form/meaning debate*. 2020. URL: https://blog.julianmichael.org/2020/07/23/to-dissect-an-octopus.html (visited on 07/23/2020).

[57] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. DOI: 10.48550/ARXIV.1301.3781. URL: https://arxiv.org/abs/1301.3781.

[58] Juri Opitz and Sebastian Burst. *Macro F1 and Macro F1*. 2021. arXiv: 1911.03347 [cs.LG]. URL: https://arxiv.org/abs/1911.03347.

[59] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. DOI: 10.48550/ARXIV.1912.01703. URL: https://arxiv.org/abs/1912.01703.

[60] Matthew Peters et al. "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. URL: https://www.aclweb.org/anthology/N18-1202.

[61] Christopher Potts. *Is it possible for language models to achieve language understanding?* 2020. URL: https://chrisgpotts.medium.com/is-it-possible-for-language-models-to-achieve-language-understanding-81df45082ee2 (visited on 10/05/2020).

[62] David M. W. Powers. *What the F-measure doesn't measure: Features, Flaws, Fallacies and Fixes*. 2019. arXiv: 1503.06410 [cs.IR].

[63] Alec Radford, Karthik Narasimhan, et al. "Improving language understanding by generative pre-training". In: (2018). URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[64] Alec Radford, Jeff Wu, et al. "Language Models are Unsupervised Multitask Learners". In: (2019). URL: https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[65] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2019. DOI: 10.48550/ARXIV.1910.10683. URL: https://arxiv.org/abs/1910.10683.

[66] Pranav Rajpurkar et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264. URL: https://aclanthology.org/D16-1264.

[67] Abhilasha Ravichander, Yonatan Belinkov, and Eduard Hovy. *Probing the Probing Paradigm: Does Probing Accuracy Entail Task Relevance?* 2021. arXiv: 2005.00719 [cs.CL]. URL: https://arxiv.org/abs/2005.00719.

[68] Siva Reddy, Danqi Chen, and Christopher D. Manning. "CoQA: A Conversational Question Answering Challenge". In: *Transactions of the Association for Computational Linguistics* 7 (Mar. 2019), pp. 249–266. DOI: 10.1162/tacl_a_00266. URL: https://www.aclweb.org/anthology/Q19-1016.

[69] Radim ehek and Petr Sojka. "Software Framework for Topic Modelling with Large Corpora". English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. http://is.muni.cz/publication/884893/en. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[70] Alan Ritter, Colin Cherry, and Bill Dolan. "Unsupervised Modeling of Twitter Conversations". In: Human Language Technologies - North American Chapter of the Association for Computational Linguistics (HLT-NAACL), Jan. 2010. URL: https://www.microsoft.com/en-us/research/publication/unsupervised-modeling-of-twitter-conversations/.

[71] Alan Ritter, Colin Cherry, and William B. Dolan. "Data-Driven Response Generation in Social Media". In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, July 2011, pp. 583–593. URL: https://aclanthology.org/D11-1054.

[72] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. *A Primer in BERTology: What we know about how BERT works*. 2020. arXiv: 2002.12327 [cs.CL]. URL: https://arxiv.org/abs/2002.12327.

[73] Stephen Roller et al. "Recipes for Building an Open-Domain Chatbot". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, Apr. 2021, pp. 300–325. URL: https://www.aclweb.org/anthology/2021.eacl-main.24.

[74] Victor Sanh et al. *Multitask Prompted Training Enables Zero-Shot Task Generalization*. 2021. DOI: 10.48550/ARXIV.2110.08207. URL: https://arxiv.org/abs/2110.08207.

[75] David Schlangen. *A coherence-based approach to the interpretation of non-sentential utterances in dialogue*. 2003. URL: https://pub.uni-bielefeld.de/download/1992167/2910547/Coherence-Based_Approach.pdf.

[76] David Schlangen. "Towards Finding and Fixing Fragments—Using ML to Identify Non-Sentential Utterances and their Antecedents in Multi-Party Dialogue". In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Ann Arbor, Michigan: Association for

Computational Linguistics, June 2005, pp. 247–254. DOI: 10.3115/1219840.1219871. URL: https://aclanthology.org/P05-1031.

[77] David Schlangen and Alex Lascarides. "The interpretation of non-sentential utterances in dialogue". In: *Proceedings of the Fourth SIGdial Workshop of Discourse and Dialogue*. 2003, pp. 62–71. URL: https://www.aclweb.org/anthology/W03-2106.

[78] Burr Settles. "Active learning literature survey". In: (2009).

[79] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. "Learning Important Features Through Propagating Activation Differences". In: (2017). DOI: 10.48550/ARXIV.1704.02685. URL: https://arxiv.org/abs/1704.02685.

[80] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. 2013. DOI: 10.48550/ARXIV.1312.6034. URL: https://arxiv.org/abs/1312.6034.

[81] Hui Su et al. "Improving Multi-turn Dialogue Modelling with Utterance ReWriter". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 22–31. DOI: 10.18653/v1/P19-1003. URL: https://www.aclweb.org/anthology/P19-1003.

[82] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. *Axiomatic Attribution for Deep Networks*. 2017. DOI: 10.48550/ARXIV.1703.01365. URL: https://arxiv.org/abs/1703.01365.

[83] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. *Gradients of Counterfactuals*. 2016. DOI: 10.48550/ARXIV.1611.02639. URL: https://arxiv.org/abs/1611.02639.

[84] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. *Sequence to Sequence Learning with Neural Networks*. 2014. DOI: 10.48550/ARXIV.1409.3215. URL: https://arxiv.org/abs/1409.3215.

[85] Jörg Tiedemann. "News from OPUS A collection of multilingual parallel corpora with tools and interfaces". In: 2009.

[86] Jörg Tiedemann. "Parallel Data, Tools and Interfaces in OPUS". In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey: European Language Resources Association (ELRA), May 2012, pp. 2214–2218. URL: http://www.lrec-conf.org/proceedings/lrec2012/pdf/463_Paper.pdf.

[87] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[88] Alex Wang, Yada Pruksachatkun, et al. *SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems*. 2020. arXiv: 1905.00537 [cs.CL]. URL: https://arxiv.org/abs/1905.00537.

[89] Alex Wang, Amanpreet Singh, et al. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355. DOI: 10.18653/v1/W18-5446. URL: https://www.aclweb.org/anthology/W18-5446.

[90] Pengfei Wei, Lu Zhenzhou, and Jingwen Song. "Variable importance analysis: A comprehensive review". In: *Reliability Engineering & System Safety* 142 (2015), pp. 399–432. ISSN: 0951-8320. DOI: 10.1016/j.ress.2015.05.018. URL: https://www.sciencedirect.com/science/article/pii/S0951832015001672.

[91] Thomas Wolf et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2019. DOI: 10.48550/ARXIV.1910.03771. URL: https://arxiv.org/abs/1910.03771.

[92] Yongqin Xian et al. *Zero-Shot Learning – A Comprehensive Evaluation of the Good, the Bad and the Ugly*. 2017. DOI: 10.48550/ARXIV.1707.00600. URL: https://arxiv.org/abs/1707.00600.

[93] Yi Yang, Wen-tau Yih, and Christopher Meek. "WikiQA: A Challenge Dataset for Open-Domain Question Answering". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 2013–2018. DOI: 10.18653/v1/D15-1237. URL: https://aclanthology.org/D15-1237.

[94] Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. "CrossFit: A Few-shot Learning Challenge for Cross-task Generalization in NLP". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 7163–7189. DOI: 10.18653/v1/2021.emnlp-main.572. URL: https://aclanthology.org/2021.emnlp-main.572.

[95] Matthew D Zeiler and Rob Fergus. *Visualizing and Understanding Convolutional Networks*. 2013. DOI: 10.48550/ARXIV.1311.2901. URL: https://arxiv.org/abs/1311.2901.

[96] Yizhe Zhang et al. "DIALOGPT : Large-Scale Generative Pre-training for Conversational Response Generation". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Online: Association for Computational Linguistics, July 2020, pp. 270–278. DOI: 10.18653/v1/2020.acl-demos.30. URL: https://www.aclweb.org/anthology/2020.acl-demos.30.

[97] Alice Zheng. *Feature engineering for machine learning : principles and techniques for data scientists*. eng. Beijing, 2018.

[98] Fengbin Zhu et al. *Retrieving and Reading: A Comprehensive Survey on Open-domain Question Answering*. 2021. DOI: 10.48550/ARXIV.2101.00774. URL: https://arxiv.org/abs/2101.00774.

[99] Yukun Zhu et al. *Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books*. 2015. arXiv: 1506.06724 [cs.CV].

[100] Fuzhen Zhuang et al. *A Comprehensive Survey on Transfer Learning*. 2019. DOI: 10.48550/ARXIV.1911.02685. URL: https://arxiv.org/abs/1911.02685.