CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG

Sentiment and Semantic Analysis and Urban Quality Inference using Machine Learning Algorithms

Master's thesis in Computer Science and Engineering

EMILY HO
MICHELLE SCHNEIDER

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022

Master's thesis 2022

Sentiment and Semantic Analysis and Urban Quality Inference using Machine Learning Algorithms

EMILY HO
MICHELLE SCHNEIDER

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2022

Sentiment Analysis and Urban Quality Inference using Machine Learning Algorithms

EMILY HO
MICHELLE SCHNEIDER

Supervisor: Yinan Yu, Computer Science and Engineering Department
Supervisor: Sanjay Somanath, Architecture and Civil Engineering Department of Chalmers

Examiner: Devdatt Dubhashi, Computer Science and Engineering Department

Master's Thesis 2022
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2022

Sentiment Analysis and Urban Quality Inference using Machine Learning Algorithms

EMILY HO
MICHELLE SCHNEIDER
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Qualitative interviews are conducted by researchers to gain a deeper understanding of people's opinions and perceptions about a specific topic. The analysis of such textual data is an iterative process and often time-consuming. To help researchers obtain an overview of the data and improve their coding process, the objective of this thesis was to investigate how state-of-art techniques can be used to classify sentiment and semantic orientation from qualitative interviews transcribed in Swedish. The results demonstrate that the implemented deep learning techniques, BERT and NER, are a possible and promising solution to achieve the stated goal. For the sentiment analysis, the Swedish BERT model KB-BERT was used to perform a multi-class classification task on a text sentence level into the three different classes: positive, negative, and neutral. For the semantic analysis, NER and String Search were used to perform multi-label classification to match domain-related topics to the sentence. The models were trained and/or evaluated on partially annotated datasets. Nevertheless, an important factor to consider is that these deep learning models are heavily data-driven and would need accurately annotated domain-specific data to reveal their full potential.

Keywords: Natural Language Processing, Machine Learning, Deep Neural Networks, Transformers, BERT, NER, Text Classification, Urban Planning, Qualitative Research, Interviews.

Acknowledgements

We would like to thank our supervisor Yinan Yu for giving us the opportunity to research about this thesis topic and for the great guidance through all the months. No matter what question we had, we could always rely on your quick answers, on your support and great feedback. Thank you Yinan!

Additionally, we thank Sanjay Somanath for introducing us to the topic of urban planning and for helping us to clarify the overall outcome. Thank you Sanjay for your clear communication and for making sure that we are thematically on the right track.

Michelle Schneider, Emily Ho, Gothenburg, July 2022

Contents

List of Figures

List of Tables

1 Introduction

Natural language processing (NLP) refers to the use of computational tools and techniques to understand, learn and produce human languages for different tasks or applications. In particular, along with rapid developments in machine learning (ML) and artificial intelligence (AI), applications of NLP include machine translation, speech-to-speech translation, identifying sentiments in reviews etc [3]. As such, a fundamental task within NLP is text classification. Text classification can use unsupervised, semi-supervised or supervised methods to label a certain entity, e.g. a word or sentence, into predefined classes. Within text classification, there are several approaches for example multi-class and multi-label classification that uses different techniques. Those can be rule-based or ML and deep learning (DL) methods. One important area within text classification and NLP is the analysis of sentiments and semantics of different texts [4, 5]. Sentiment analysis describes the investigation of a text-based on its polarity. This can be positive, negative, or neutral and therefore mirrors the emotions and opinions of the author. In comparison, semantic analysis extracts the main topics that have been said. Both sentiment and semantic analysis are used in a wide range of domains. This thesis will focus on its application in the field of urban planning.

Urban planning of cities considers the ecological, social, and economic aspects of city living. More specifically, a part of urban planning is to understand the perception of urban places by the residents in the city. One way to investigate this is by conducting and analysing qualitative interviews to gain a deeper understanding of people's attitudes and perceptions (sentiment) about a specific topic (semantic).

In this master's thesis, a collaboration with the Architecture and Civil Engineering Department of Chalmers (ACE), here also addressed as end-users, was formed to investigate the perception of different areas within Metropolitan Gothenburg. Interviews with more than 400 people were conducted to gather qualitative data to study how residents perceive their neighbourhoods. To help the end-user gain an overview of the interview data, visualisations such as tree maps and word clouds are used to show word frequencies. However, these methods do not consider the relationship between people, places and themes. Therefore, this paper will mainly focus on extracting the sentiment and semantic context from Swedish transcribed interviews using text classification technologies, such as Bidirectional Encoder Representations from Transformers (BERT), Named Entity Recognition (NER) and String Search, in the domain of urban qualities.

1.1 Aim

The primary goal of this master's thesis is to analyse the data from the interviews using statistical and/or ML-/ DL- algorithms. The results will be used to improve the understanding of the urban quality of the areas within Metropolitan Gothenburg. Hence, it is important to identify the sentiments and semantics within the interviews. In this respect, the overarching research problem can be stated as follows:

How can state-of-art techniques be adjusted and applied to classify sentiment and semantic orientation from qualitative interviews, in order to provide city planners and researchers with an accurate overview of the data?

This in turn will be split up into the following sub-questions:

1) Which state-of-art algorithms can be adopted and what are the pros and cons?

2) How can we visualise the semantic and sentiment analysis of interview data?

1.2 Limitations and Challenges

Since the raw data is in Swedish, our project will be limited to models trained in Swedish. Furthermore, the given dataset for sentiment analysis is not labelled, which makes it difficult to evaluate correctly. One solution was to create a semi-supervised dataset that had to be manually annotated. The annotations were done by several people, which increases the overall human error in the analysis as annotations can be subjective. The annotation of the sentiment is based on the written text and therefore linguistic expressions such as sarcasm or irony can be more difficult to detect. Also, for the semantic analysis only the test set was annotated into some of the predefined labels and therefore was only used for evaluation instead of model training. Overall for both analyses, annotating more or the entire dataset would be too time-consuming and beyond the time frame given for the scope of this thesis.

1.3 Thesis outline

In this thesis, Chapter 2 presents the general background information needed to understand the methodology used. First, general knowledge about urban planning is provided, highlighting its importance. Following is an introduction to text classification, which introduces different approaches to conduct sentiment and semantic analysis in its subsection. The approach chosen for this thesis is explained in detail in Section 2.5. Subsequently, Chapter 3 presents the implemented methodology step by step. Here, the dataset and the pre-processing steps are explained along with the description of the implementation of the final models. After, the results obtained are exhibited in Chapter 4, followed by the discussion and conclusion in Chapter 5.

2
Background

This chapter starts by illustrating the general overview of the importance of urban planning in Section 2.1. In Section 2.2, key concepts within text classification are described, and the study of sentiment and semantic analysis are explored and explained in Sections 2.3 and 2.4. Lastly, Section 2.5 gives an in-depth explanation of the architecture of the DL models along with a short explanation of the evaluation metrics used.

2.1 Urban Planning

The urban population has been growing rapidly around the world, where around 55% of the world's population lives in urban areas today. It is estimated that by 2050, this figure will increase to approximately 68% [6]. Already since 1950, the urban population has grown from 751 million to 4.2 billion people in 2018 [6]. This mass movement of the population from rural to urban areas and the resulting physical changes to the urban settings is often referred to as urbanisation [7]. Urbanisation has long been associated with positive effects such as reducing poverty, fostering human and technological development, and being the heart of economic growth in many nations. However, with the shift of the population from rural to urban areas, there is an increased need to build sustainable cities and public services such as transportation, waste management, housing, energy systems, and other infrastructure solutions. Many cities face challenges and increased pressure to ensure access to the above-mentioned infrastructure, as well as basic services such as employment, education, healthcare and a safe environment [6, 7, 8, 9]. This influx of humans could, for example, form the basis for inequality, poverty, environmental hazards such as pollution, road traffic, and communicable and non-communicable diseases. etc. [7]. Consequently, to compensate for the adverse effects of urbanisation, the sustainable development of cities depends greatly on integrated policies and good governance to improve the social, economic and environmental aspects of urbanisation [6, 8].

Studies have shown that urban planning positively contributes to the health and well-being of the urban population, improving the quality of urban living [10, 11]. Especially policies that regulate land use, connectivity, density, transport, and green infrastructure have been shown to be instrumental in improving health outcomes [9]. For example, in Sweden, green areas play the role of both air cleaners and noise reducers, while also creating meeting places for residents. Moreover, reducing car traffic in the city centres and building in favour of bikes and pedestrians has led to better health amongst the residents [8, 12]. Accordingly, understanding that the environment and spatial planning is an integral factor of health, and is imperative in both the conceptualisation of policy development and its solutions [9, 11, 12]. As Carmichael et al. [9] describe, the environment in which people spend their lives has a profound impact on their physical, mental, social, and economic well-being.

One way to support the policies and governance in creating healthy urban living in the local communities is to understand the resident's perception of their neighbourhood. Uncovering neighbourhood satisfaction and discovering the reasons why residents move, their desires and intentions to relocate is one method to inform the planning process of the neighbourhoods [12, 13]. Research into the residents' perception of their neighbourhoods has thereby facilitated the development of more informed public policies with the aim to improve each individual's life satisfaction and also the community's well-being [14].

2.2 Text Classification

Text classification is a fundamental task within the study of NLP, especially with the recent advancements in NLP and text mining along with the increasing amount of digital text documents available. Text classification refers to the task of classifying written text or documents into predefined categories and can be seen as a multidisciplinary field that covers computational linguistics, NLP, ML, and AI [15, 16]. Classical examples of text classification include, but are not limited to, topic labelling, sentiment analysis, NER, and news classification. Here, the common goal is to predict a predefined label for a given text, although the labels differ depending on the domain [17].

Text classification has not only been gaining traction in the scientific community but also within the business realm with the development of real-world applications that can leverage these text classification methods [17, 18, 19]. For example, text classification can improve capabilities within antispam systems, bot and fraud detection, and commercial document classification, as well as be used in recommendation systems by filtering what is shown to customers by excluding different features [15, 19, 20].

One research area in which text classification has become popular is in the social sciences and qualitative research methods, where computational social sciences have emerged as a field of study to support the analysis of the growing data sets [21, 22]. More specifically, ML techniques have been used to assist in the analysis of qualitative data by qualitative coding [21]. Qualitative research methods are often used in social sciences to probe the social environment of humans by discovering attitudes and beliefs on a specific topic through interviews, focus groups, and observations [23]. Once transcribed, the descriptive data will need to be analysed in order to derive meaningful insights and infer conclusions. This is usually done through qualitative coding, where descriptive or inferential labels are assigned to fragments of data in a systematic order. Such coding may assist in theory development, where the hypothesis and theories are constructed based on the patterns and shared characteristics of the coded data [21]. However, performing such manual coding is usually labour-intensive and time-consuming, especially in the era of big data [21, 22, 23]. As social scientists only sample and code a small part of the data, their analysis can be seen to be incomplete because inconsistencies in their theories may arise as a large part of their data are still unexplored [21]. Consequently, text classification methods can also be applied to assist in the identification and extraction of attitudes, opinions, and topics from these interview transcripts [22, 23].

2.2.1 Types of Text Classification

In general, text classification can be divided into different groups, such as multi-class or multi-label, depending on how many classes or labels are assigned to each text. Multi-class text classification systems assigns only one class from a set of more than two classes to each sample of text. Multi-label classification, on the other hand, assigns one or more classes from a set of classes to each sample of text [24].

2.2.2 Computational Methods in Text Classification

In the context of text analysis, there are different levels of granularity; document, paragraph, or sentence level [18]. The algorithms used for text classification systems would predict the relevant classes for either a full document, a single paragraph, or a single sentence. For sentiment analysis, there is another level of granularity, namely aspect-based. Document and sentence level for sentiment analysis focuses on the general polarity of each, such as whether the document or sentence expresses a positive, negative, or neutral stance. In comparison, aspect-based sentiment analysis focuses on the specific object the opinion is about [16, 20].

There are two main methodologies used for text analysis of different granularity; a) rule-based, and b) ML- and DL-based [16, 25]:

a) Rule-based: Rule-based systems are based solely on predetermined rules set by the model architect or language experts in this case. The classifier would be made up of a set of human-coded rules that would classify the data into one or multiple classes. In terms of sentiment analysis, the classifier would rely on a sentiment lexicon built on a set of word units and their sentiment tag of neutral, positive or negative [25, 26]. A naive method would thus be to count the opinionated words in the text based on the dictionary. If there are more positive than negative words, the overall sentiment would be positive, and vice versa.

b) ML- and DL-based: In contrast to rule-based methods, ML and DL are data-driven and rely on algorithms and statistical models. They learn patterns from past data to automatically make predictions for a given set of unseen data. There are three different approaches in which ML can be performed; supervised, unsupervised, and semi-supervised learning. Supervised learning is performed with the target value or label, and thus requires an annotated corpus on which the models train to learn the patterns for classification. Unsupervised learning, in contrast, is performed without a target value, which means that the model tries to make sense of the data by extracting the undiscovered patterns on its own. Semi-supervised learning then uses a small amount of labelled data to train models that would support the prediction of the larger set of unlabelled data [26].

Within the scope of this thesis, the focus will be placed on DL models that will be used for the task of sentiment and semantic analysis. A more detailed technical explanation of the specific DL methods implemented is found in Section 2.5.

2.2.3 Text Representation

When using text as input to the many ML and DL algorithms, it is important to pre-process and clean the text from any noise that may be present. For example, most raw text and documents would include unnecessary words such as stop words, slang, punctuations, or even misspellings that may have an adverse effect on the model performance [18]. The following is an explanation of text pre-processing steps that are relevant for this use case :

Lemmatisation: Lemmatising is the process of extracting the most basic word form (lemma) from each word [18]. This is done by replacing or removing the suffix, prefix, infix, or other inflexions of the word in question [18]. For instance, for the word '*moved*', the lemma would be '*move*'. Similarly, for Swedish, the lemma for the word '*flyttat*' is '*flytta*' [27]. Lemmatising words is also important for capturing the context of the word, and thus the overall meaning of the sentence [28]. For instance for the word '*leaves*', the lemma would either be '*leave*' or '*leaf*', depending on the context.

Tokenisation: Tokenisation is a method that separates a piece of text into smaller elements called tokens [18]. This step is a fundamental part of the pre-processing method as many NLP algorithms require the text input to be tokenised in order to be used, as it can then be transformed into a numerical data structure as the suitable input for the algorithms. Moreover, tokenisation is important in understanding which words are present in the piece of text since the meaning and context of a piece of text can be analysed through the words present in the text and also through the sequence of those words [18].

2.3 Sentiment Analysis

Sentiment analysis is a common example of multi-class text classification where written text is classified based on the expressed attitude, feelings, or opinions towards a certain entity [16, 29, 30, 31]. These subjective expressions are often categorised into positive, negative, or neutral classes. Sentiment analysis can be seen to be one of the most prolific research areas within computer science today [29]. This can be attributed to the growing availability of the opinions of the general public online on objects, brands, topics, ideas, or even products, which has been enabled by the evolution of Web technology [29, 31]. These opinions may be available in the form of blogs, social networks, and online forums as well as reviews on websites and e-commerce sites [18, 29]. As a result, the opportunity to capture these opinions has increased the interest in many different domains such as products and services, health care, financial services and politics [30, 31, 32]. For instance, research has revealed that sentiment analysis can be used to predict sales revenue for businesses or even the stock market fluctuations and trading strategies [33, 34, 35, 36].

Initial research within sentiment analysis first started off with rule-based systems, whereas the trend has moved on to using ML and now DL methods [37, 38]. A large number of articles have been published using traditional ML methods such as Naïve Bayes, Support Vector Machine, Decision Trees and Random Forests, and K-nearest Neighbour, with Support Vector Machine being the most successful model performed on this task [18, 37]. However, a major challenge with rule-based approaches and traditional ML models is that features are hand-crafted, where feature engineering and feature extraction are the most time-consuming process. The advantage of DL neural networks, on the other hand, is that they can automatically learn the features from the data [37, 38, 39]. Especially with the increasing amount of data generated, traditional ML-based models are inferior to DL models. The DL models can be trained to learn more key features from large-scale training data with multiple layers, automatically capturing complex and non-linear patterns in the data [37].

Some of the most popular DL models applied in sentiment analysis are Convolutional Neutral Network (CNN), Recursive Neural Network (RNN) including Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), Bi-directional Long Short-Term Memory (Bi-LSTM), and transformer-based networks [15, 17, 37]. As text classification tasks can also be considered sequential modelling tasks, RNNs are more frequently used compared to CNNs due to their ability to learn sequential associations, which is an important feature when dealing with semantic analysis of text [15, 18]. LSTM and Bi-LSTM models, which are based on RNN architectures, are capable of capturing long-term dependencies, taking into account the preceding and succeeding contexts in a text passage, and making achievements in the academic field [15]. Several studies even suggest the use of hybrid models for sentiment analysis, where a combination of DL models has performed better than individual models [39]. The hybrid model of CNN-LSTM has achieved 91% accuracy in detecting the fine-grained polarity of IMDB and Amazon reviews outperforming traditional DL and ML techniques [40].

However, most recently, the state-of-the-art techniques used for sentiment analysis are the transformer models [1, 17]. Transformer models process all words simultaneously rather than sequentially. Yet, it is able to ensure that the same word has different representations depending on its position in the sentence, and it can also learn the dependencies between words due to the self-attention mechanism [17].

Using this approach, transformers have been shown to produce better results, while at the same time being less time-consuming due to their ability to parallelise in training [17, 39].

Although there is extensive research conducted within sentiment analysis, the majority of them have been modelled using English corpora [30, 37]. This results in a shortage of resources and tools, such as non-English datasets and benchmarks, making it difficult to build good sentiment classifiers for other languages such as Swedish [30, 38]. To mitigate these issues, studies have tried to leverage English sentiment analysis systems by translating a non-English corpus into English and then using existing classifiers to annotate the corpus [30]. However, this may induce errors due to the different linguistic and language differences. For example, unlike English, negative Swedish sentiment is most often written in definite forms using '*the*', while positive sentiments are expressed more frequently in indefinite forms such as '*a*' [41].

Not only is the modelling language important, but research has also shown that sentiment classification is highly sensitive to the domain, the specific topic and the context of the training data. As Liu [30] explains, words and the language constructs used for expressing opinions can differ in different domains. The same word in one domain may depict a positive sentiment, whilst for another domain may be negative. For example, the word '*easy*' in the domain of electronics might be positive *'The camera is easy to use'* while it might be negative in the domain of movies *'The end of the movie was easy to guess'* [42].

This domain-transfer problem is also a challenge to be addressed for this thesis as sentiment analysis on interview transcripts is less researched, more so in Swedish and urban planning.

2.4 Semantic Analysis

The goal of semantic analysis is to explore the meaning of human language. In the field of NLP, this involves evaluating and representing words and word relationships and investigating their meaning as an element. The aim here is to give the model the same word interpretation as humans have and to extract the context [43, 44]. To understand the meaning of sentences, there are two tasks involved in semantic analysis: Word Sense Disambiguation and Relationship Extraction [45].

Word Sense Disambiguation describes the ability of a model to identify the ambiguity of words. Human language is often associated with ambiguities, which means that a word can have different meanings depending on the context [46]. The following sentences will show an example:

(a) I can hear *bass* sounds.
(b) They like grilled *bass.*

Here the word *bass* is used in two different contexts and thus has different meanings: one refers to low-frequency sounds whereas the other refers to a specific type of fish. For a human, this is easy to deduce, but for a computer, it is a challenge. To process the context of texts, a computer must transform unstructured text information accordingly and feed it into clear data structures, which in turn must be analysed to detect the subliminal word connections and meanings [46].

The second task within Semantic Analysis is Relationship Extraction. The task here is to recognise different entities within a sentence and to extract their relationship to each other [47]. To understand it more clearly following example is given:

Gothenburg is in *Sweden.*

Here, the relationship extraction would notice that the *city: Gothenburg* is within the *country: Sweden* and that those two words are related to each other. As with Word Sense Disambiguation, it is relatively easy for a human to recognise the links between the city and the country, but to teach this to a computer, the marked entities and the context of the whole sentence must be used [47].

Depending on the desired result, semantic analysis can be used in different ways. One of the most common techniques is text extraction, which includes keyword extraction and entity extraction [45]. The aim of keyword extraction is to detect certain keywords within a document that are explicitly mentioned in the text [48, 49]. Here, keywords are a series of one or more words that represent the main content of a document in a condensed form [50]. The latter technique, entity extraction, attempts to identify all entities contained in a document. One method that comes into focus is Named Entity Recognition (NER), which is often used for semantic text extraction and is described in more detail in Section 2.4.1 below [51].

For the semantic analysis in this thesis, entity extraction is an approach that will be investigated within the urban planning domain.

2.4.1 Named Entity Recognition

NER is a task within NLP and semantic analysis where the objective is to detect and classify named entities in text [52]. A named entity refers to a word or a series of words that identifies an item with similar attributes from a collection of other items, such as the name of an organisation [53, 54]. For instance, *Chalmers* is a named entity of the entity type *organisation.* NER is then the process of recognising the names of the pre-defined semantic entities [53]. Examples of named entity types are person, location, organisation, dates, and times [52, 53, 54]. Depending on the domain of interest, these named entity types may vary. For example, in the medical domain, drug and disease names may rather be seen as entities [54]. Basic NER models typically distinguish between a small set of entity types and usually, one type per named entity. However, recently, NER tasks are able to consider a larger set of 100 or more distinct entity types, where a named entity may also be assigned to multiple types [52, 53]. Extraction of such information from text has been seen to be an essential pre-processing step for a number of downstream NLP applications such as Automatic Text Summarisation, Question Answering, Machine Translation, Information Retrieval etc. [53, 54, 55]. In terms of this thesis and use-case, understanding which areas of Gothenburg and Housing-related organisations have been mentioned in the interviews are of interest. As such, NER can be used to detect the named entities of organisations and locations.

Similar to other text classification models mentioned in Section 2.2.2, the algorithms proposed in the NER models can be classified into two different methods; rule-based and ML- and DL-based computational methods [55, 56]. Rule-based approaches are mainly based on semantic and syntactic rules to identify the entities and thus work well when the lexicon and rules are exhaustive [53]. As the rules are able to capture domain-specific properties of the language the models are able to obtain sufficient accuracy [54]. Quimbaya et al. [57] proposed a rule-based system for electronic health records to extract named entities such as diagnosis and treatment using a dictionary lookup. However, a drawback of such rule-based systems is that they can rarely be transferred to other domains or languages [53, 54]. Moreover, they are quite expensive as these models require human expertise within the domain and/or language [54]. Thus, recent NER models rely on the ML- and DL-based algorithms and approaches that can overcome the drawbacks of the earlier models [54].

The DL systems employed for the NER tasks in recent years have been found to yield substantial improvements [55]. Compared to the traditional ML algorithms based on heavy feature-engineering, the DL models have shown effectiveness with their ability to automatically detect useful features through nonlinear processing and ability to consider long-term dependencies [55, 56]. In particular, studies have shown that transformer-based models such as BERT, RoBERTa and XLNet, can outperform non-transformer-based models such as RNN, CNN, LSTM and other hybrid models as Bi-LSTM-CNN-CRF models within the NER task [52].

Additionally, there are many factors that may affect the performance of the NER models. For example, the language or domain in which the task is performed as well as the number of entity types to be identified [54]. Lack of resources such as a large and high-quality annotated dataset for certain languages and domains makes the NER task more challenging [53, 54]. Without the annotated data as input for training or evaluation, it may be difficult to classify a named entity correctly, especially when the text is ambiguous [53, 54]. For example, *Apple* may refer to an organisation as well as a common noun. Thus, many languages that are resource-poor may resort to the rule-based and traditional ML algorithms to execute the NER tasks [56]. The approach in which NER technique to use depends strongly on the availability of the labelled and annotated language resource [54, 56]. Especially for DL methods, a large data set is generally required for training the model [56].

2.5 Transformer

To understand the theory of the DL model used in this thesis the following Section gives an introduction to the transformer method. Here, the overall architecture of the transformer will be explained followed by a detailed description of its main components: self-attention and multi-head. In addition, the specific transformer model BERT, which was implemented for this thesis, is presented.

As mentioned in Section 2.3, the current state-of-the-art models used in text classification are transformer models. Transformers, introduced by Vaswani et al. [1], is a non-recurrent model based on the self-attention concept that creates dependencies between input and output globally. The main reasons the self-attention mechanisms are preferred are due to computational complexity, parallelisation, and the ability to learn long-range dependencies between words, which are all essential for building contextualised word embeddings [58]. The transformers' structure is built mainly of an encoder-decoder architecture where each component consists of so-called transformer blocks. In the original paper presented by Vaswani et al. [1], it consists of a stack of six transformer blocks but Alammar highlights that this number can be adjusted [59]. Additionally, the transformer blocks can be stacked on top of each other since the input and the output of each block have the same size [1]. Figure 2.1 represents the model architecture with an example of a transformer block in the encoder (left) and decoder (right) components.

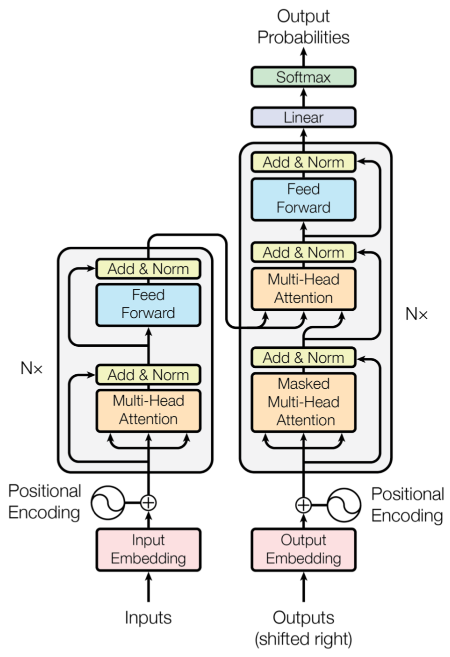

Figure 2.1: Transformer Architecture for one Transformer Block, each based on Vaswani et al. [1].

Encoder

Within each transformer block in the encoder component, there are two main modules: a multi-head self-attention layer, which is the key technology of the transformer model and will be further described in Sections 2.5.1 and 2.5.2, and a fully connected feed-forward layer [60]. Additionally, residual connections and normalisation layers are applied to both of the modules to improve the learning and training rate [61].

As it can be seen in Figure 2.1, the overall transformer process begins by converting the input sequence into embeddings - this step happens only in the very bottom encoder. Since transformer models do not contain any convolution or recurrent methods they have to add positional encoding to use the input sequence properly. Positional encoding is an added vector that keeps the information about the token's position within an input sequence and therefore has the same dimension as the embedded data [1]. From there, the residual connection is applied to the input data, which will be further processed in the first multi-head self-attention layer. Residual connections add the inputs of the lower layer to the generated self-attention output without having gone through the intermediate steps. This summation is then processed in the normalisation layer. Here, the mean and standard deviation of the input data to be normalised will be calculated. Then the mean is subtracted from the element in the input and divided by the standard deviation. The output of the normalisation layer can be further processed in the feed-forward layer or in the next transformer block [60]. Since the transformer blocks are stacked on each other, the created encoder output is used as input for the next encoder module; this process repeats till the sequence arrives at the last encoder block. The final output of the encoder stack represents the context that will be processed in the decoder module [59].

The transformer architecture makes it possible to apply parallelisation since embeddings can pass through the encoder layer almost individually. Few dependencies consist of the self-attention layer, but in the feed-forward layer, each embedding will be processed separately. Using parallelisation in the transformers leads to faster training times compared to recurrent encoder-decoder models [59, 60].

Decoder

The decoder architecture, shown in the right part of Figure 2.1, works similar to the encoder. In the original construction, it consists of a stack of six decoder blocks. Positional encoding is also added to the decoder input sequence to follow the order of the tokens [59]. The main difference between the encoder and decoder side comes with a third sub-layer. In addition to the general multi-head self-attention and feed-forward layer, a modified attention layer is implemented at the beginning of each decoder stack. Modification is done by masking. Since the whole input sequence is fed into the decoder structure, masking is used to prevent the model to gain knowledge of words that comes after the word currently processed at position i. This is done by setting the values of the following words to $-\infty$ [60]. Additionally, the embedding of the generated encoder output is moved by one position, which in combination with masking ensures that the prediction for the word at position i consists of known words at position smaller than i [1].

After the input sequence passed through all decoder steps, it reaches the linear layer which creates words out of the decoder output. This layer can be described as a fully connected neural network that resizes the decoder output vector into a much larger vector called logits. Usually, the logits vector size is equal to the dimension of the vocabulary. Finally, the softmax layer converts the scores from the logit vector into probabilities where all values are positive and add up to 1. The word that will be chosen as output for this timestamp is the cell with the highest value [59].

2.5.1 Self-Attention

The self-attention layer is used by the transformer to comprehend the relevance of other words in relation to the word that is currently processed [59]. For this purpose, a score is calculated that indicates how much attention other words receive when a word is encoded.

It does so, by first, creating three different matrices: Query (Q), Key (K) and Value (V), where $Q, K, V \in \mathbb{R}^{m \times n}$ and m represents the number of tokens and n represents the embedding dimension. Q and the matrix of the current focus, while K represents the previous input that is compared to Q, and V is used to calculate the outcome of Q [60]. The matrices are generated by multiplying the embedding matrix (X), where each row represents one word from the input sentence, with the respective weight matrix (W), which is trained during the training process, so that [59]:

$$Q = XW^Q; K = XW^K; V = XW^V$$

In a second step, each word of the input sequence is scored against the word of attention. This is achieved by taking the dot product Q times K. The outcome is divided by the square root of K's dimension, and softmax is applied to normalise. Afterwards, the result is multiplied with V to maintain the values of words that the model should focus on. Finally, the weighting values are summed, giving the output of the self-attention layer [59, 60].

$$output = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

2.5.2 Multi head

As seen in Figure 2.1, the multi-head attention is the actual self-attention layer. It copies the procedure of single self-attention, described in the previous Section and performs it several times. To be precise, eight different self-attention heads i, each with its own random assigned Query, Key and Value matrices W_i^Q, W_i^K, W_i^V, are executed in parallel [59, 60].

In general, multi-head is used to improve the model's capacity to focus on multiple positions in the input sequence, with the differently weighted parameters supporting the learning process to target different word relations. Overall, this procedure allows for a more detailed illustration of the word relationships. Processing eight self-attention layer in parallel ends up with eight different weighted matrices for one input sequence. Since the feed-forward layer expects only one input, the results have to be combined. This happens by concatenating and multiplying them with another weight matrix W^O which was developed in conjunction with the model [60].

2.5.3 BERT

BERT stands for Bidirectional Encoder Representations from transformers and is a language model that was introduced by Google researchers Devlin et al. in 2018 [2]. As the name implies, its purpose is to understand the meaning of words on both left and right sides of the sentence. The model is based on the multi-layer encoder architecture of the transformer model described in Section 2.5.

The BERT model comes with two different architectural sizes, which are in principle larger than the original encoder block: $\text{BERT}_{\text{BASE}}$ is a model consisting of a total of 12 transformer Encoder Blocks with 12 self-attention heads and a feed-forward network size of 768. $\text{BERT}_{\text{LARGE}}$ is a more comprehensive model which has an overall 24 transformer Encoder blocks, 16 self-attention heads and a network size of 1024 [2].

Input Representation

To forward the input text into the first encoder stack, three different embeddings per token must be generated.

The first embedding is the tokeniser, which splits the input into word pieces. This ensures that unknown words can be broken down and therefore be recognised and further processed by the model. For example, the English word *playing* will be divided into *[play]* and *[##ing]*. Additionally, the tokeniser adds the following special tokens to the embedding [2]: [CLS], [SEP], [PAD]. The classification token [CLS] is used to mark the start of each input sequence. The separation token [SEP] operates as a delimiter between sentences and marks the end of the sequence. Since all input sequences must have the same length, paddings tokens [PAD] are appended to shorter sentences.

The second additional embedding layer is called segment embedding. It helps the model to recognise where the separation tokens are placed to identify to which sentence and context the word of focus belongs. For that, the separation token saves the information for every token whether it belongs, e.g., to sentence A or sentence B. If the input sequence consists of a single sentence, the segmentation token will be the same for all tokens.

The last embedding layer is the positional embedding and is identical to the one described in the transformer section. As mentioned there, it supports the model by saving the information of relative position within an input sequence of each token. Finally, all three embeddings are summed up and generate the final embedding that can be further processed in the model. Figure 2.2 visualise the described steps.

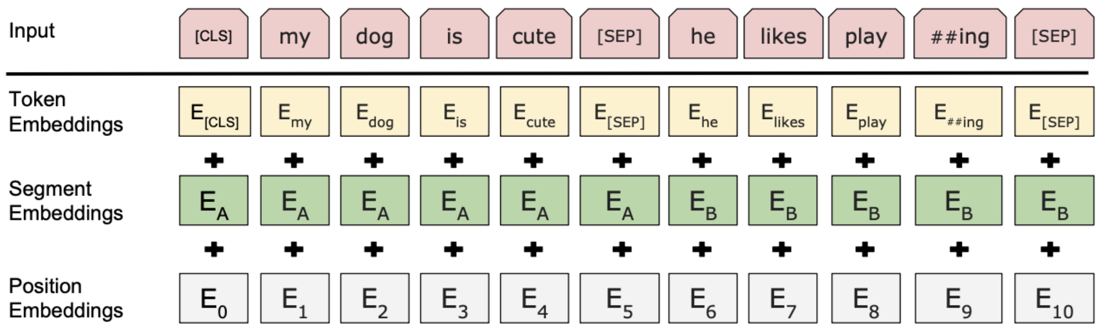
Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2.2: BERT input representation [2].

Pre-training

Because BERT has bidirectional connections, it would theoretically allow the word being predicted to recognise itself. Thus, it would render the process of prediction irrelevant, as the model can simply access the word it is looking for through the various connections without generating a learning factor. A solution to this problem is Masked Language Modelling (MLM), which is one of the two pre-training tasks used in the BERT model.

MLM masks the target words using the [MASK] token and uses the remaining ones as a context. This forces the model to rely on the meaning of the surrounding words and use them as the main source for prediction. For this process, 15% of the input tokens are determined as the target word. Once the words have been selected, the model runs through three different processes. In the example below, step (1) is used in 80% of the iterations, the token is replaced by the mask token. In step (2) another 10% replaces the token with a randomly chosen word, whereas the target word is visible to the model in the remaining 10%, as seen in step (3). The underlying goal of masking is to achieve generalisation of the model's predictive ability.

Example:
(1) The moon looks so bright, today -> The [mask] looks so bright, today
(2) The moon looks so bright, today -> The *cat* looks so bright, today
(3) The moon looks so bright, today -> The *moon* looks so bright, today

The second pre-training task is the so-called Next Sentence Prediction (NSP). As one can deduce from its name, the goal here is to determine which sentence follows the one in focus. Through this task, the model learns to recognise the relationships between different sentences in a better sense. A binary classifier task is posed to detect whether sentence B directly follows sentence A or not. Given that this is a binary task, pairs of sentences are used as input. As with MLM, the model passes through various iterations. In 50% of the cases, sentence B follows sentence A, in the remaining 50%, a sentence from the corpus is chosen at random.

2.6 Evaluation Matrices

In order to assess the efficiency of ML and DL models, accuracy, precision, recall, and the F1-score are commonly used as measurements in text classification, both for sentiment and semantic processes [62, 63]. This Section will explain the mentioned measurements in more detail.

The confusion matrix, as seen in Table 2.1, describes the performance of a model and introduces four different classification results. In this table, the predicted values are compared with the actual values for more precise comparison. The results True Positive (TP) and True Negative (TN) describe the correctly predicted classes for the actual positive or negative labels. False Positive (FP), on the contrary, indicates when the model predicts an actual negative element as positive. A similar thing happens with False Negative (FN), here an actually positive element is declared as negative [64].

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

Table 2.1: Confusion Matrix

Based on the Confusion matrix, accuracy, precision, recall and F1-score can be derived:

Accuracy describes how many elements were correctly predicted compared to the total number of elements. It takes the number of elements of the correctly classified elements (TP+TN) and divides it by the total number of elements (TP+TN+FP+FN).

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.1}$$

Precision describes how many elements the model has correctly classified as positive (TP) compared to the total amount of correctly and incorrectly predicted elements assigned to that class (TP+FP).

$$precision = \frac{TP}{TP + FP} \tag{2.2}$$

Recall describes how many elements the model has correctly classified as positive (TP) compared to the total number of elements present in the respective label (TP+FN). Thus, the value indicates the percentage of labels found by the model. Usually, data that achieves a high recall score ends up with a low precision Score and vice versa.

$$recall = \frac{TP}{TP + FN} \tag{2.3}$$

F1-score is the harmonic mean between recall and precision and thus balances both results and gives a result between 0 and 1. F1-score can be a good guide to unbalanced data distribution, as each label is given a weight based on its support, i.e., the number of instances within the target label [64]. The recall and precision scores are multiplied by each other and by two and then divided by the common addition.

$$\textit{F1-score} = \frac{2 * precision * recall}{precision + recall} \tag{2.4}$$

Macro Based Evaluation

In addition to the evaluation metrics mentioned above, there is an averaging method that can be applied to precision, recall and F1-Score, which are of interest in multi-label classification: macro average [65].

Macro average treats all given classes equally. Here, the predictions (TP, FP, TN, FN) are taken and the respective metric is calculated, then the results are added and divided by the number of classes [65].

$$\textit{Macro F1-score} = \frac{1}{N} \sum_{i=0}^{N} \text{F1-score}_i \tag{2.5}$$

3
Methods

In this Chapter, the general workflow of this thesis is explained. Firstly, Section 3.1 starts with a description of the dataset and moves on to explain the data pre-processing, cleaning methods, and the annotation process. The Sections 3.2 and 3.3 present the experimental setup for sentiment and semantic analysis before each step of the implementation of the models is detailed, respectively.

3.1 Dataset

The datasets used for the text classification tasks consisted of interview documents along with semantic classes and labels provided by the ACE department. There were in total 447 interview documents, where 245 documents (55%) were in the format of interview Transcripts and the other 202 documents (45%) were in a written Summary format. The list of semantic class target labels was also provided to check whether these sentences mentioned any of these target labels. The list consisted of eight semantic classes that included labels describing names of housing-related organisations, names of all neighbourhoods in Gothenburg, names of housing units, objects belonging to the neighbourhood, physical components within a unit, and rooms of a unit.

3.1.1 Data Pre-processing

In order to train the model using the raw interview documents and labels, they were first cleaned and pre-processed to make them suitable for the models.

3.1.1.1 Text Inputs

For the interview documents, the documents were first received in the format of Microsoft Word documents and had to be transformed into the Python string format for modelling. As sentiment and semantic classification were performed on the sentence level, each document was also split into individual sentences in string format. The following explains in detail the complete pre-processing method:

Firstly, the interview documents were manually checked and labelled according to whether they were in a Transcript or a Summary format as the cleaning would differ depending on the format. These labels were saved in the metadata .csv file and later parsed as a `DataFrame` object in Python. An example of an interview Transcript and Summary can be found in Appendix A.1 and Appendix A.2. It was important to separate the two formats because different models would be trained separately on them as Transcripts uses a first-person narrative while Summary a third-person narrative. Next, all the documents were converted from Microsoft Word documents to text documents (.txt files) containing plain text in the form of lines only using the `docx2txt` package from the Python Package Index (PyPI) [66].

Once in text format, the cleaning step for each document followed:

1) parentheses '()', hard brackets '[]', and any text in between were removed. This is because the information within the parenthesis and brackets were placeholders for laughter or inaudible words.
2) The text made up of unicode characters was then normalised using `unicode.normalize()` function.
3) The text was then divided into separate lines using the `splitlines()` function, and empty lines and tabs were also removed simultaneously.

This cleaning step was performed for both the interview Transcripts and the Summaries. Each document was saved in its corresponding dictionary with the key-value pair of interview ID and interview text. The following parts will explain the next cleaning step for the respective formats:

Interview Transcripts

During the manual check, it was found that an interview transcript only included responses to a questionnaire sent to interviewees before the interview. This particular document was removed from the datasets. For the rest of the interview Transcripts, the first few lines of documents started off with information about the interview, such as date, time and place, transcription notes and descriptions, the initials and full name of the interviewer and interviewee IDs, before the actual main body of what each speaker said. This descriptive information was also removed as only the words of each speaker are of interest for the modelling.

However, inconsistencies between the main body text were also found. Most transcribers (170 of 245 Transcripts) included the initials of both the interviewer and the interviewee ID before the speaker's words. 65 documents only included the initials of the interviewer whilst ten documents had the speaker identifier omitted completely. Since it was important to keep track of who the speaker was and what they said, the latter two groups of documents were modified to include the speaker identifiers. The ten documents without speaker identifiers were manually altered and labelled by the authors of this thesis, while the others were automatically changed. Lastly, each interview was saved into a dictionary consisting of two lists; the order of speaker and the corresponding sentences split by line.

Summary of interview

Through the manual check, it was discovered that two Summaries included the interviewer or transcriber's personal comments such as "*Min dator ville inte spara filen så texten försvann direkt efter jag skrivit*" and "*Kanske ska tas bort eftersom den är så kort?*" which translates into "*My computer did not want to save the file so the text disappeared.*" and "*Remove because it is so short?*" respectively. Therefore, these two Summaries were removed.

For the other interviews that were transcribed as Summaries, the first few lines of descriptive information about the interview were also removed, similar to the interview Transcripts. Some Summaries also included the responses to a questionnaire at the end of the text. These lines of text were also removed by searching for lines that start with "*Svar på enkät*", "*Enkätfrågor*", or "*Enkätsvar*", which all roughly translates into "*Questionnaire responses*".

In total after cleaning and preprocessing, there are 444 interview documents, 244 Transcripts, and 200 Summaries, consisting of 43,110 and 5,354 individual sentences, respectively.

3.1.1.2 Semantic Class Labels

The semantic class labels provided were a collection of semantic classes with an array of labels under them. There were in total eight different classes provided at the start, where two of the classes were supplementary lists of existing ones. The supplementary list was therefore merged with the original list to create a fully comprehensive one. For example, for the class *rooms of a housing unit*, there were two lists; one '*room_scale*' including a set of rooms of a unit, and another '*room_scale_extra*' including names of other possible rooms. As both lists consist of rooms in a unit, these two were merged. After merging supplementary lists, there were finally six different semantic classes; housing-related organisations (*org*), neighbourhoods in Gothenburg (*nbr_scale*), housing units (*unit_scale*), objects belonging to the neighbourhood (*nbr_sub*), physical components within a unit (*unit_sub*), and rooms of a unit (*room_scale*).

As labels can appear in different grammatical forms in the text, the labels for four of the semantic classes were lemmatised using the `Stanza` Python NLP package [27]. The example below illustrates the lemmatisation process conducted for this thesis. Here, both sentences refer to the kitchen, although the actual word is presented differently ("köket", "kök"). If the labels were not lemmatised, the class would have to include both "köket" and "kök", but now with only "kök", limiting the number of label instances in each semantic class.

Example:
1) Jag gillar köket.
2) Jag tycker att mitt kök är stort.

The two semantic classes that were not lemmatised were *org* and *nbr_scale*. This is because the labels under these classes were the names of organisations and neighbourhoods. Since the original name wants to be caught, lemmatisation was not needed.

In the final step of this pre-processing, all duplicate labels were removed for each of the six classes using the python `set()` function.

In total there were six semantic classes and a total of 190 semantic class labels; 7 for housing related organisations (*org*), 119 for neighbourhoods in Gothenburg (*nbr_scale*), 10 for housing units (*unit_scale*), 12 for objects belonging to the neighbourhood (*nbr_sub*), 27 for physical components within a unit (*unit_sub*), and 15 for rooms of a unit (*room_scale*).

3.1.2 Annotation Process

For both the semantic and sentiment analysis, the original dataset was unlabelled. This meant that each sentence did not have a target label for the topic discussed, such as the neighbourhood, or a target label for sentiment if the opinion was negative, positive or neutral. The following part will explain how the labelling was performed for the different analyses.

Sentiment Analysis

For sentiment analysis, as a semi-supervised dataset was needed for the training and evaluation of the DL methods, manual annotation had to be performed. The annotation was done in Google Excel Sheets by the authors of this master thesis. In the process, ten documents from the interviews were divided into their sentences. Of these ten documents, four are from the category Summaries and six from the category Transcripts. The annotation was done in a two-phase principle:

After each document was divided into individual sentences, each annotator classified each sentence into one of the three classes: positive, neutral, and negative. Thus, each sentence was initially assigned to two independent classes, one by each annotator. It should be mentioned that Swedish is the mother tongue of one of the annotators, whereas the second currently has basic knowledge. In this case, a translator platform was used to assist.

In the second step, the sentences that showed discrepancies were filtered. These were discussed by both annotators and a label was agreed upon. For the sentences that were classified the same by both annotators, that specific class became the final label automatically.

The reasoning for each label annotation was based on a few created rules: Overall, it was decided that sentences that mention a rent increase will be marked as negative. Rent reductions due to a move or similar are marked as positive. It was also decided that flats or rooms described as large should be marked as positive, small ones as negative, with the exception that the interviewee explicitly mentions that he or she would like to move to a smaller flat. Furthermore, it was recognised that energy efficiency is an important issue. Statements in favour of an energy-efficient flat were marked as positive, while neutral or negative statements were marked with the respective label. Sentences that addressed social aspects were also marked according to certain rules. Relocation due to closer proximity to family was considered positive, as was a communicative neighbourhood. If a neighbourhood appeared unsafe, this was certainly marked as negative. Questions were also raised about the communication of the renovation on the part of the housing companies. If respondents felt that they were sufficiently informed, this appeared positive. If the respondents could not reach the companies at times or were unhappy with the opening hours, this was recorded as negative.

On the other hand, there were some sentences that made the annotation difficult due to different sentiments within one sentence. See the following example in Swedish, the original language, and then translated into English:

"*Gillar området och trivs men det var verkligen grannarna som sabbade alltihopa.*"

"*Like the area and enjoy it but it was the neighbours who ruined everything.*"

As can be seen here, the sentence contains, for human understanding, two different statements and opinions. Since the fundamental aim of this thesis is to preserve the perception of the areas in the city, the neighbours have been subordinated in this example, and thus the sentence has been considered positive. It was deliberately decided against a neutral and thus balancing opinion of the sentence, as otherwise, the sentence would have drowned in the final visualisation, although it contains important opinions. Thus, either a positive or negative classification would be more appropriate here.

Semantic Analysis

As the models used for semantic analysis were unsupervised pre-trained learning models, a labelled dataset was not needed for training, but rather for the evaluation of the models. To check whether the topics of interest for the end-user were detected, the end-user themselves annotated a part of the test dataset, where the set was skimmed and the most concise labels were extracted. Additionally, the end-users provided a full list of target labels in case the annotated set did not cover all the labels that were of interest. This full list can be seen in Appendix A.3.

3.1.3 Final Dataset

After pre-processing and annotation, the entire dataset can be divided into annotated and non-annotated data, see Table 3.1. The latter has an overall 47,818 sentences where the majority (89.10%) belongs to the category Transcripts and a smaller amount of 5,212 sentences (10.90%) to Summaries. For the annotated dataset, it consisted of 705 sentences, where 561 (79.60%) belong to Transcripts and 144 (20.40%) to Summaries.

For both training and evaluating steps, the models used only the annotated dataset, as such, the labelled set was split into a training and test set. As seen in Table 3.2, within the category Summary, 89 sentences (61.81%) belong to the training set and 55 sentences (38.19%) to the test set. Within the category Transcripts of 561 sentences, 475 (84.67%) belong to the training and 86 (15.33%) to the test set. The distribution of the training and test dataset compared to the full dataset for the Summary and Transcript can be seen in Figure 3.1 and 3.2.

	Annotated	Non-annotated	Total
Summary	144	5,212	5,326
Transcripts	561	42,606	43,167
Total	705	47,818	48,493

Table 3.1: Final Dataset of Annotated and Non-annotated Sentences.

	Training Set	Test Set	Total
Summary	89	55	144
Transcripts	475	86	561
Total	564	141	705

Table 3.2: Training and Test Set for Summary and Transcript Category.

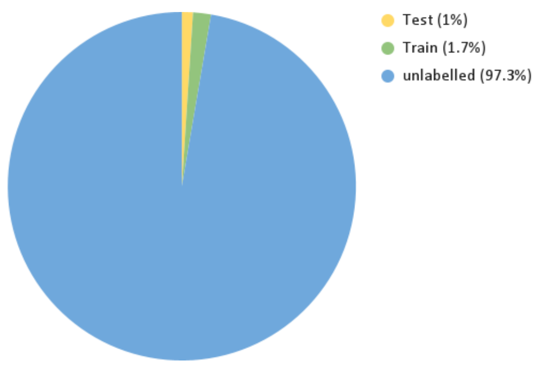

Figure 3.1: Data distribution for Summaries.

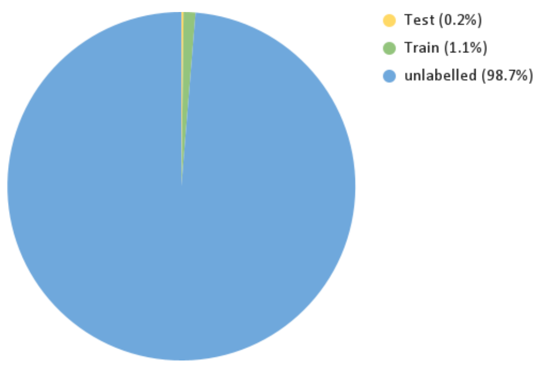

Figure 3.2: Data distribution for Transcripts.

Sentiment Analysis

In the annotated dataset for sentiment analysis, 705 sentences were assigned one label. Overall, it contains a total amount of 112 positive, 424 neutral, and 169 negative sentences, as seen in Figure 3.3. More specifically, the Summary dataset contains 43 positive and almost the same amount of neutral and negative sentences (51 neutral, 50 negatives). Due to the fact that the Transcript dataset is larger, the amount of sentences is also bigger here: it has a total of 69 positive, 373 neutral and 119 negative sentences.

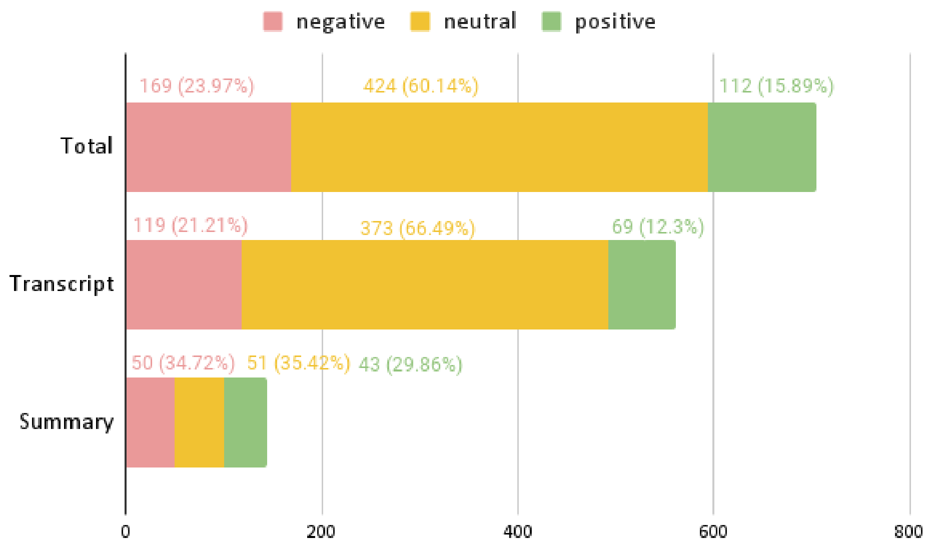

Figure 3.3: Data split into Transcripts and Summaries.

As mentioned earlier, the annotated dataset can be split into training and test sets, both for Summary and Transcript. For the category Summary, the training set shows 20 positive, 35 neutral, and 34 negative sentences; an almost similar distribution can be found in the test set with 23 positive and 16 neutral and negative sentences each. Details can be seen in Figure 3.4. Within the category Transcripts, the total dataset contains 12% of positive sentences while negative reaches around 20% and the majority lies within the neutral labels at 66.49%, as can be seen in Figure 3.5.

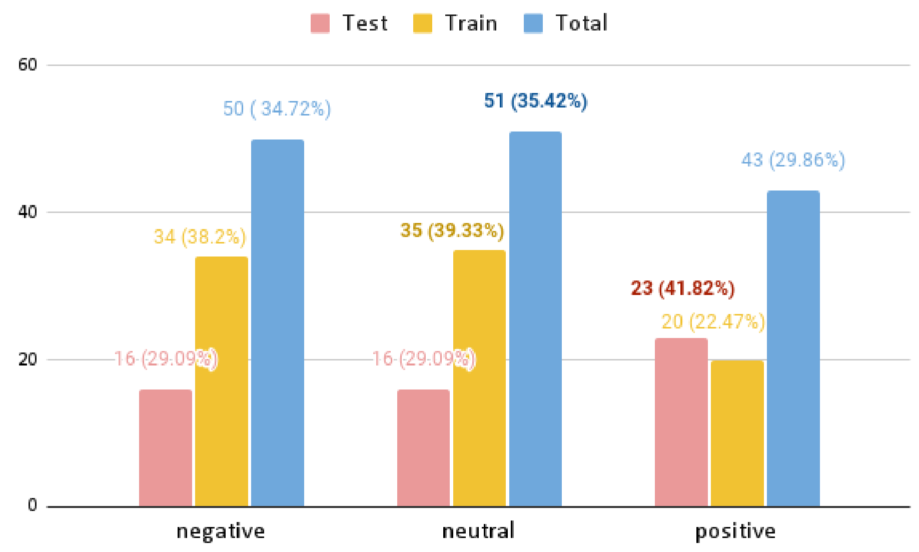

Figure 3.4: Sentiment data distribution for Summaries.

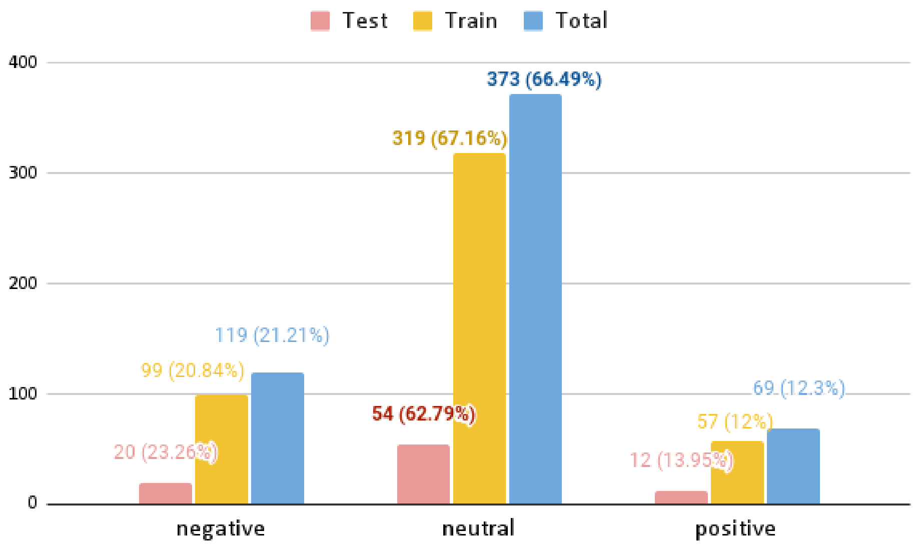

Figure 3.5: Sentiment data distribution for Transcripts.

The annotated training dataset was also separated into two different levels of difficulty, Dataset$_{\text{simple}}$ and Dataset$_{\text{complex}}$, to train and validate the final model in the later process. These steps will be explained in more detail in Section 3.2.2. The

Dataset$_{\text{simple}}$, includes those labels that contained opinion matches from the first annotation step, and the Dataset$_{\text{complex}}$, consists only of those sentences that were filtered and finally classified in the second step. Out of the total 705 annotated sentences, 595 sentences (84.40%) can be found in Dataset$_{\text{simple}}$, while a disagreement on 110 sentences (15.60%) belongs to Dataset$_{\text{complex}}$. Figure 3.6 illustrates a more detailed division into positive, neutral, and negative.

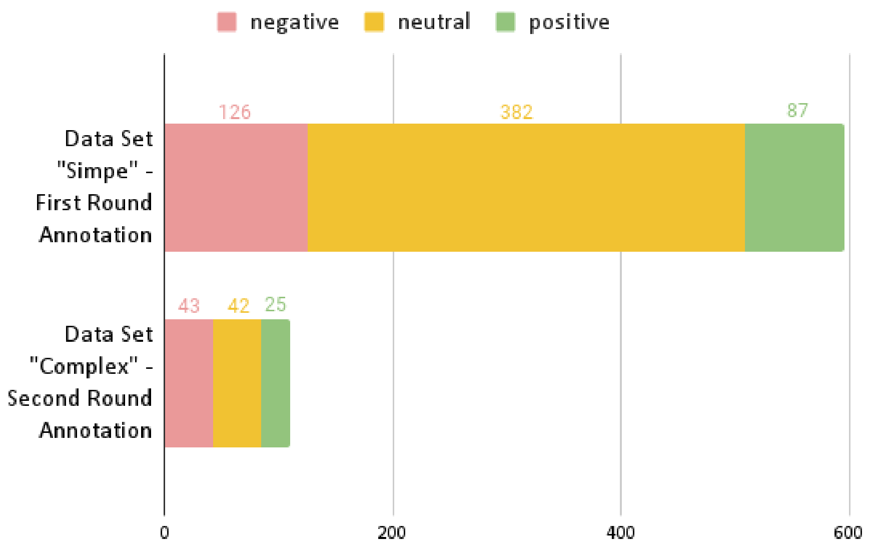

Figure 3.6: Two different complexities of Dataset created from Rounds of Annotation

Semantic Analysis

As the semantic models were unsupervised and did not require any training, the annotations were applied only to the test set. That is, only 55 Summary and 86 Transcript sentences were labelled by the end-user. A total of 190 semantic class labels, divided into six different classes, were provided, where a sentence could be labelled with either no or more than one label from a different or the same class. Table 3.3 illustrates the distribution of annotated labels per semantic class for both the Summary and the Transcript datasets.

For an overview of the semantic class labels provided, an exhaustive list of the semantic class target labels can be found in the Appendix A.3. In general, the *nbr_scale* class has the most amount of labels with 62.6% and 119 labels. The second biggest class is *unit_sub* which has 14.2% and 27 labels. *nbr_sub* with 6.3% and 12 labels belongs to one of the smaller classes. *roomscale* has 15 labels (7.9%), almost similar to *unit_scale* with 10 labels and 5.3%. The smallest class is *org* which contains 10 labels (3.7%), see Figure 3.7.

	Summary	Transcript	Total
nbr_scale	7	13	20
org	2	1	3
nbr_sub	8	5	13
room_scale	3	9	12
unit_scale	20	14	34
unit_sub	7	25	32
Total	47	67	114

Table 3.3: Semantic Annotated Labels

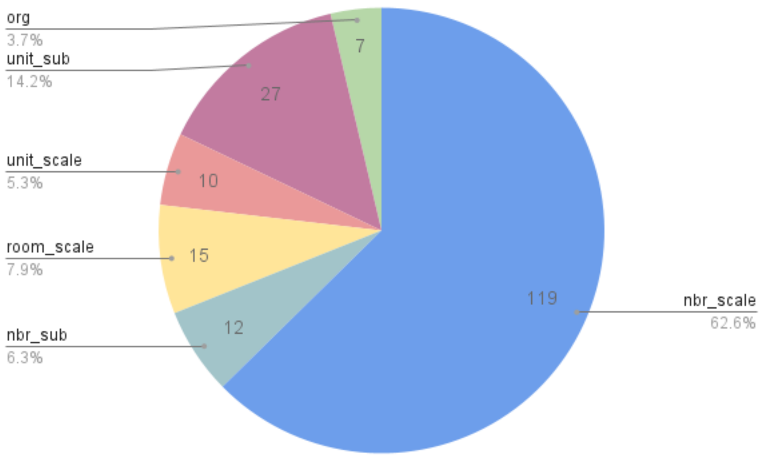

Figure 3.7: Semantic class distribution

To summarise the final dataset, the data used for text input can be divided into annotated and non-annotated data, where the first makes almost 1.5% of the total sum of sentences. Additionally, the sets were divided into two categories: Transcripts and Summary, as seen in Figure 3.3. From here, each category is further divided into test and train sets. Independent from this, specifically the sentiment analysis, the Dataset$_{\text{complex}}$ and Dataset$_{\text{simple}}$ were created, where the latter is used as the first training set and the former as the additional dataset used for hyperparameter-tuning. For the sentiment analysis, the annotations consisted of a positive, negative or neutral label. On the other hand, for the semantic analysis, 190 labels were provided, where none or multiple labels could be assigned to each sentence in the dataset.

3.2 Sentiment Model

In this Section, the implementation of the sentiment model will be discussed. A multi-class text classification model for sentiment analysis was implemented using a pre-trained BERT that was fine-tuned to suit the domain. A modified version of the text classifier by Tran [67] was implemented to fit multi-class classification. To highlight, in this thesis the model is implemented individually once on the Summary and once on the Transcript side.

Since the approach of this thesis follows a semi-supervised methodology and in that overall only 1.5% of the data is annotated, additional approaches had to be applied to ensure an accurate outcome. For this reason, a voting ensemble classifier was created with the aim of better performance compared to a single classifier. As the name suggests, a voting ensemble chooses the label based on the outcome of the soft majority of classifier predictions, i.e. each model predicts a probability for each class. These are accumulated and the highest value in one of the classes is selected.

Regarding the technical setup, the experiments and models were carried out on `Jupyter Notebooks` on the `Google Colaboratory` platform with an NVIDIA Tesla K80 GPU and 12GB of RAM.

3.2.1 Pre-trained BERT Models

The pre-trained Bert model used in this thesis was a language-specific BERT model for Swedish ("KB-BERT") developed by the KBLab at the National Library of Sweden (KB) [68]. The KB-BERT was pre-trained on around 15-20GB of text including approximately 260M sentences and 3400M tokens from diverse sources such as digitised newspapers, official reports, legal e-deposits, social media and Swedish Wikipedia articles [68].

The specific model used for pre-training was the `bert-base-swedish-cased (v1)` model, a BERT trained with the same hyperparameters as presented by Google. This model includes 12 encoder layers, 768 hidden layer sizes of the feed-forward network, and 12 attention heads with around 50K vocabulary size. It is also cased, meaning that there is no change to the input text. The other alternative would be uncased where the model processes each word as lower case tokens.

3.2.2 Implementation

Here a view of the sentiment model implementation is given. A graphically visualisation can be seen in Figure 3.8.

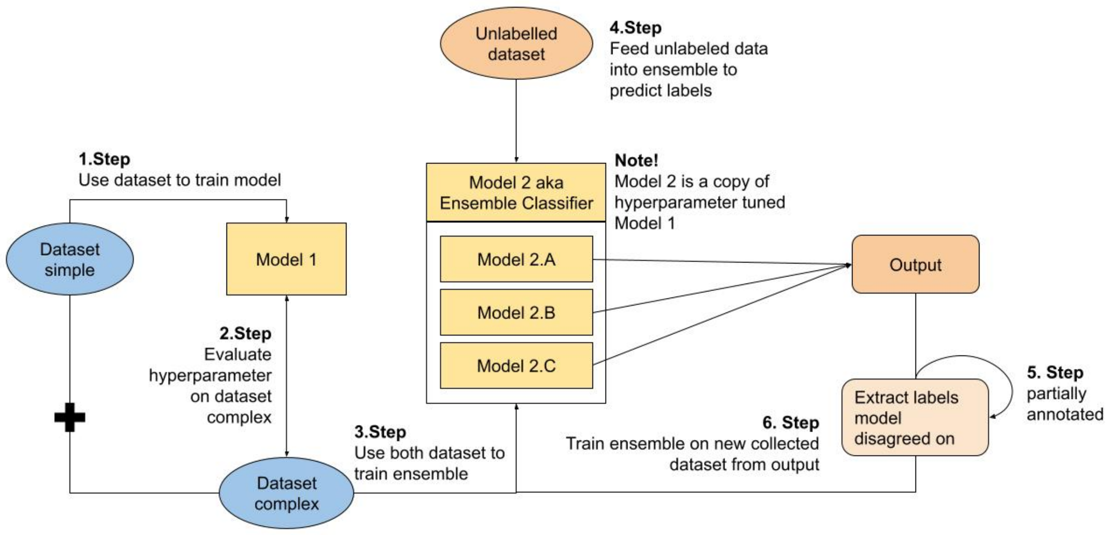
Unlabelled dataset
4.Step
Feed unlabeled data into ensemble to predict labels
1.Step
Use dataset to train model
Model 2 aka Ensemble Classifier
Note!
Model 2 is a copy of hyperparameter tuned Model 1
Dataset simple
Model 1
Model 2.A
Model 2.B
Model 2.C
Output
2.Step
Evaluate hyperparameter on dataset complex
5. Step
partially annotated
Extract labels model disagreed on
3.Step
Use both dataset to train ensemble
6. Step
Train ensemble on new collected dataset from output
Dataset complex

Figure 3.8: Visualisation of Model Process

Model 1

For loading the pre-trained KB-BERT model to Python, the Hugging Face's Transformers API was used as recommended by KB. Hugging Face is an open-source and platform provider for NLP technologies made by and available to the community, providing thousands of pre-trained models to perform tasks with text, vision, and audio [69].

The training data used for each of the Summary and Transcript categories were the respective Dataset$_{\text{simple}}$, consisting of only agreed labels from the annotators. This training set was split into a training and validation set of 90% training and 10% validation. For the Summary set, a random split was used, while for the Transcripts a stratified split was implemented due to the large imbalance of classes for the Transcript set, as seen in Section 3.1.3.

To apply pre-trained KB-BERT, the training and validation datasets had to be tokenised, as described in Section 2.5.3 explaining the input representation of text to BERT. For tokenisation, the same tokeniser provided by KB-BERT as the model has a fixed vocabulary and a specific way of handling out-of-vocabulary words. These were loaded using the class and method `AutoTokenizer` and `from_pretrained()` from Hugging Face Transformers. The method `encode_plus` was then used to 1) split text into tokens, 2) add special tokens [CLS], [SEP] and [PAD] to each sentence, 3) convert tokens into indexes of the existing tokeniser vocabulary, 4) pad and truncate the sentence to a constant length, specified as the maximum length for input sequence, and lastly 5) create the attention mask. Since using the maximum length for the input sequence of 512 tokens would have been computationally expensive, the number was updated to 194 tokens. This number was found by finding the longest sentence input sequence from both the Summary and Transcript dataset.

Once the training and validation sets were tokenised, they were passed into the multi-class BERT classifier (Model 1) with a pre-trained KB-BERT model to extract the last hidden layer and then a single-hidden-layer feed-forward neural network as the classifier. For the fine-tuning of the weights and learning rate of the model, an AdamW optimiser was created, where the weight decay and learning rate were optimised separately. Additionally, the warmup linear optimiser schedule was used, where a higher learning rate is set for the initial stages of learning but over time decreases linearly in order to reach optimum better. At each epoch, the training performance was measured using the validation dataset with Accuracy and Loss as the defining metric.

Fine-tuning

Model 1 was also fine-tuned based on Dataset$_{\text{complex}}$. This dataset was also tokenised accordingly as explained in the Section above. Different numbers of epochs, batch sizes, and learning rates were investigated. Based on the original BERT model [2], the recommended fine-tuning values were as follows:

```
Batch size:  16, 32
Learning rate (Adam):  5e-5, 3e-5, 2e-5
Number of epochs:  2, 3, 4
```

Except for the number of epochs, the hyperparameters followed the original paper. A set of random combinations of these hyperparameters was chosen as an exhaustive search would have been too time-consuming and not in the scope of technical possibilities when using Google Colaboratory.

Model 2 - Ensemble Classifier

Once Model 1 was fine-tuned according to the best hyperparameters, it was trained again on the full dataset consisting of Dataset$_{\text{simple}}$ and Dataset$_{\text{complex}}$. See Figure 3.8 for illustration.

As mentioned above, an ensemble classifier, Model 2, was implemented to improve reliability, especially with the low number of annotations. Therefore, three different BERT models, Model 2.A, 2.B, and 2.C, based on Model 1 was created using different seeds to control randomness and reproducibility. The predicted label was then determined by the soft majority vote of the three classifiers. The predicted labels and sentences were then each saved in a list.

To further improve the reliability, Model 2 was trained again on another retrospectively annotated dataset. This dataset was created from the non-annotated dataset, where Model 2 was used to predict the full unlabelled set for both Summary and Transcripts. Then, the sentences on which the three classifiers did not agree were further checked manually by the authors. Due to the time constraint of this thesis, only a partial part of the disagreed labels were annotated by the authors using the same procedures as explained in Section 3.1.2. These later became the retrospectively annotated dataset that was further used to train each of the three classifiers in Model 2.

3.2.3 Evaluation

Once the fine-tuning and training of the model is completed, the next step is to evaluate the multi-class sentiment model. The different metrics used for evaluation were precision, recall, (macro) F1-score, and Accuracy which are based on the confusion matrix. More specifically, for the Summaries a weighted average of the F1-score was used, while for the Transcripts a macro F1-score was applied for evaluation. These metrics were calculated using the Python `sklearn.metrics` library. The explanation of each metric can be found in Section 2.6.

In addition to the mathematical evaluation methods, a qualitative evaluation will be conducted. Since this project is tailored to the end-users needs, they have to evaluate it over a period of time with different datasets to properly judge the application approach. Thus, due to the scope of this thesis, only mathematical evaluation will be discussed.

3.3 Semantic Model

To discover the semantics, the topics the interview documents mentions, two different models were used, NER and a naive String Search. The NER model was implemented using a pre-trained DL model, while String Search followed a string matching algorithm to find if a string is found within a larger piece of string or text. The String Search was implemented as a complement to the NER. The technical setup is the same as the semantic analysis, where the experiments and models were carried out on `Jupyter Notebooks` on the `Google Colaboratory` platform with an NVIDIA Tesla K80 GPU and 12GB of RAM.

3.3.1 Pre-trained Models

NER BERT Model

The pre-trained NER model used was a Swedish BERT model fine-tuned for NER for Swedish, also developed by KBLAB at KB, the National Library of Sweden [68]. This model was trained on the Stockholm-Umeå Corpus (SUC) 3.0 dataset from 2012, which is composed of a collection of Swedish text with approximately one million tokens and 74,000 sentences [70]. The specific model used was the `bert-base-swedish-cased-ner` model, with the same parameters as the BERT model used for the sentiment analysis. That is 12 encoder layers, 768 hidden layer sizes of the feed-forward network, and 12 attention heads with around 50K vocabulary size. The model was cased and also trained with whole word masking. For the NER tagging, eight different entity types were applied, time (*TME*), organisations (*ORG*), personal names (*PRS*), locations (*LOC*), events (*EVN*), work art (*WRK*), and measures (*MSR*), and objects (*OBJ*).

3.3.2 Implementation

NER

For NER, the model was loaded using the `NER` pipeline from HuggingFace. The model was then used to classify each sentence into different tags. For this use-case and thesis, only the entity types location *(LOC)* and organisation *(ORG)* are of interest. Therefore, if the sentence contained tokens that were tagged with *LOC* and *ORG*, these were saved in a dictionary with key-value pair of sentence index numbers and then the individual tokens as values.

String Search

In order to extract the semantic class labels, which were introduced in Section 3.1.1.2, from the text, a String Search was implemented. The aim of the String Search is, as can be deduced from the name, to search for certain strings within a text and to output both the word and the index position. In this work, the String Search was carried out separately for each class and thus a total of six times.

First, to capture the lemmatised labels of *'unit_sub','unit_scale','nbr_sub'*, and *'room_scale'* in the text, the input text also had to be lemmatised. For the two other classes (*'org', 'nbr_scale'*), the original text was inserted.

Two inputs are entered into the String Search: a list of the labels of interest and a list of sentences to be analysed. Both input text and labels were also lower cased so that the word format matches. If the label was found in the sentence, matched by the string, the index of the position of the sentence in the list, as well as the label, was saved. Similar to the NER output, these were saved into a dictionary with key-value pair of sentence index numbers and then the individual tokens as values.

3.3.3 Evaluation

In order to evaluate the performance of the NER and String Search model, the given partially annotated dataset introduced in Section 3.1.3 was used as the target value. Due to this, only Accuracy is considered in this thematic context. If the target word is in the word list extracted by the model, the model performance is considered successful.

4

Results

This chapter includes a comprehensive explanation of the results obtained by the different developed classifiers for sentiment and semantic analysis. First, in Section 4.1 performance results from the Sentiment Model are presented from both the hyperparameter tuning and the predictions on the unlabelled dataset and the final test set. Then, in Section 4.2, the model performance for the semantic analysis is described. Lastly, Section 4.3 presents a visualisation of the predicted labels for both the semantic and sentiment models of the Summary set.

4.1 Sentiment Analysis

4.1.1 Hyperparameter Tuning

For both Summary and Transcripts datasets, Model 1 was hyperparameter tuned based on their respective Dataset$_{\text{complex}}$. The results of the hyperparameters that were examined can be found in table 4.1. It can be seen that for the Summary category, the highest F1-score, 0.55, occurs at *epoch* = 10 and *learning rate* = 2e-5. In comparison, it can be seen that for Transcript interviews the highest macro F1-score, 0.56, occurs at *epoch* = 10 and *learning rate* = 5e-5. These best performing hyperparamenters were then used for three classifiers in Model 2.

Hyperparameters		**Summary**	**Transcript**
epoch	learning rate	F1	macro F1
10	5e-5	0.51	**0.56**
10	3e-5	0.44	0.52
10	2e-5	**0.55**	0.50
8	5e-5	0.44	0.48
8	2e-5	0.45	0.53
4	5e-5	0.51	0.50
4	2e-5	0.40	0.45

Table 4.1: Hyperparameter Tuning Results.

4.1.2 Prediction of Unlabelled Dataset

Table 4.2 depicts the number of instances that the three classifiers within the Model 2 agreed or disagreed on for the unlabelled dataset.

		Summary	Transcript	Total (%)
Agreed	Negative	1,484	5,654	7,138
	Neutral	2,325	26,501	23,826
	Positive	997	3,420	4,417
	Total	4,786	35,575	40,361 (84.41%)
Disagreed	Total	426	7031	7,457 (15.59%)
Grand Total		5,212	42,606	47,818 (100.00%)

Table 4.2: Model Agreements on Unlabelled Set.

Out of the disagreed labels, 301 sentences of the Summary category, and 401 sentences of the Transcript category were annotated again by the authors. Tables 4.3 and 4.4 then show the sentiment classes originally predicted from Model 2 compared to the new human annotations. These, annotated sentences were retrospectively used for training the Model 2 again, as explained in the implementation of the semantic model in Section 3.2.2.

Table 4.3 shows that within the Summary interviews, the model was able to correctly assign 45.45% (50 of 110) as actual negative sentences. In the neutral class, 17.74% (11 of 62) were correctly classified and in the positive class, 27 of 129 sentences (20.93%). Most predicted positive sentences were classified by the model as neutral, whereas most neutral sentences were labelled as positive. The majority of negative sentences were assigned to the correct class and performed the best among the classes.

		Model 2			
		Negative (%)	Neutral (%)	Positive (%)	Total
Annotators	Negative	**50 (45.45%)**	42	18	110
	Neutral	15	**11 (17.74%)**	36	62
	Positive	44	58	**27 (20.93%)**	129
	Total	109	111	81	301

Table 4.3: Summary Annotation Results.

Table 4.4 presents the results for the interview Transcripts. It can be seen that of a total of 93 actually negative sentences, 55 (59.14%) were added to the correct class.

More than half (57.39%) of the neutral sentences could also be correctly classified. The same holds for the positive class, where 41 of 78 (52.56%) positive sentences were correctly classified. It is also noticeable that in the Transcript interviews the highest number of predictions were made in the respective correct class. Overall, the negative class performed the best.

		Model 2			
		Negative (%)	Neutral (%)	Positive (%)	Total
Annotators	Negative	**55 (59.14%)**	26	12	93
	Neutral	60	**132 (57.39%)**	38	230
	Positive	13	24	**41 (52.56%)**	78
	Total	128	182	91	401

Table 4.4: Transcript Annotation Results.

4.1.3 Evaluation on the Test set

This Section presents the results of the predictions on the test set for the Summary and Transcript categories made by Model 2 without and with training on the retrospective annotations. The results are illustrated in a confusion matrix and classification report for each category and model.

4.1.3.1 Model 2 - without Retrospective Annotated Training

The tables in this Section showcase the performance of Model 2 without the training of the retrospective annotations from the unlabelled dataset.

To start off, the Summary dataset had an accuracy and weighted F1-score of 0.60 and 0.61, respectively, for the multi-class classification as seen in Table 4.5. More precisely in terms of accuracy, the model classified the neutral class the best, where it classified 62.50% of total neutral labels correctly. The positive and negative classes had a smaller accuracy score of 60.87% and 56.25% each. This is illustrated in Table 4.6. On the other hand, the positive class had a higher precision score (0.82) which contributed to the highest F1-score of the classes at 0.70. There was a distinct difference in F1-score compared to the negative and neutral classes each had a low score of 0.53 and 0.56, due to the low precision of those classes.

	Precision	Recall	F1	Amount
Negative	0.50	0.56	0.53	16
Neutral	0.50	0.62	0.56	16
Positive	0.82	0.61	**0.70**	23
Accuracy			**0.60**	55
Macro Avg	0.61	0.60	0.59	55
Weighted Avg	0.64	0.60	**0.61**	55

Table 4.5: Summary Classification Report without Retrospective Annotations.

		Predicted			
		Negative (%)	Neutral (%)	Positive (%)	Total
Actual	Negative	**9 (56.25%)**	5	2	16
	Neutral	5	**10 (62.50%)**	1	16
	Positive	4	5	**14 (60.87%)**	23
	Total	18	20	17	55

Table 4.6: Summary Test Set Confusion Matrix without Retrospective Annotations.

For the Transcript test set, Model 2 had an overall accuracy of 0.87 and a macro F1-score of 0.82 (Table 4.7). The neutral sentiment class can be seen to perform the best with the highest score for accuracy, precision and recall at 0.92, 0.94 and 0.93, respectively. Correspondingly, the other two sentiment classes also exhibited good performance scores. In table 4.8, with regard to the accuracy, the negative class had a score of 80.00%, where 16 out of 20 classes were correctly predicted, while the positive class had a score of 75.00% (9 of 12 correct predictions).

	Precision	Recall	F1	Amount
Negative	0.80	0.80	0.80	20
Neutral	0.94	0.93	**0.93**	54
Positive	0.69	0.75	0.72	12
Accuracy			**0.87**	86
Macro Avg	0.81	0.83	**0.82**	86
Weighted Avg	0.88	0.87	0.87	86

Table 4.7: Transcript Classification Report without Retrospective Annotations.

		Predicted			
		Negative (%)	Neutral (%)	Positive (%)	Total
Actual	Negative	**16 (80.00%)**	2	2	20
	Neutral	2	**50 (92.60%)**	2	54
	Positive	2	1	**9 (75.00%)**	12
	Total	20	53	13	86

Table 4.8: Transcript Test Set Confusion Matrix without Retrospective Annotations.

4.1.3.2 Model 2 - with Retrospective Annotated training

First, Table 4.9 and Table 4.10 presents the confusion matrix and the corresponding classification report for the predictions on the Summary dataset. It can be seen that the model predicted the neutral most accurately. 75.00% were correctly classified in the neutral class, while positive and negative classes were correctly assigned 69.56% and 50.00% of the labels, respectively. However, the F1-score was the highest for the positive class at 0.73, as it had both a high precision and recall performance. There is, however, a rather negligible difference between the macro and weighted average of the different scores as the classes were balanced. Overall, the model had an accuracy performance of 0.65 and a weighted F1-score of 0.65.

		Predicted			
		Negative (%)	Neutral (%)	Positive (%)	Total
Actual	Negative	**8 (50.00%)**	5	3	16
	Neutral	2	**12 (75.00%)**	2	16
	Positive	3	4	**16 (69.56%)**	23
	Total	13	21	21	55

Table 4.9: Summary Test Set Confusion Matrix.

	Precision	Recall	F1	Amount
Negative	0.62	0.50	0.55	16
Neutral	0.57	0.75	0.65	16
Positive	0.76	0.70	**0.73**	23
Accuracy			**0.65**	55
Macro Avg	0.65	0.65	0.64	55
Weighted Avg	0.66	0.65	**0.65**	55

Table 4.10: Summary Classification Report.

Table 4.11 and Table 4.12 depicts the performance of Model 2 on the Transcript test set, where it can be seen that the overall accuracy of the model was 0.85 while the macro F1-score was 0.80. More specifically, from the total of 86 sentences, 13 out of 20 sentences (65.00%) could be correctly classified in the negative class. Also in the neutral class, the model shows good values: here the majority (96.30%) could be predicted correctly and therefore performs among the classes the best. In the positive class, 9 of 12 sentences (75.00%) were correctly assigned. The F1-score was also the highest for the neutral class at 0.92. However, it should also be noted that there is a significant difference between the weighted and macro average of the F1-score, where the class imbalance plays a considerable role in this output.

		Predicted			
		Negative (%)	Neutral (%)	Positive (%)	Total
Actual	Negative	**13 (65.00%)**	5	2	20
	Neutral	0	**52 (96.30%)**	2	54
	Positive	1	2	**9 (75.00%)**	12
	Total	14	59	13	86

Table 4.11: Transcript Test Set Confusion Matrix.

	Precision	Recall	F1	Amount
Negative	0.93	0.65	0.76	20
Neutral	0.88	0.96	**0.92**	54
Positive	0.69	0.75	0.72	12
Accuracy			**0.85**	86
Macro Avg	0.83	0.79	**0.80**	86
Weighted Avg	0.87	0.86	0.86	86

Table 4.12: Transcript Classification Report.

4.2 Semantic Analysis

In this section, the results of the test set of semantic methods (NER and String Search) are presented. Table 4.13 shows the results of the Summary and Table 4.14, the results of the Transcripts. It must be emphasised, that in these results only those labels that were marked in the annotated set are investigated. Table 4.15 shows the number of all labels found per class compared to the annotation.

In Table 4.13 it can be seen that for the Summary interviews the highest value of NER is in the '*org*' category (0.86%). String search, on the other hand, achieved an accuracy value of 1 in three (*nbr_scale*, *org*, *room_scale*) of the six classes. The lowest value is in the category '*unit_sub*' with an accuracy of 0.71%.

Table 4.14 shows that for the Transcript interviews, NER could achieve an accuracy value of 1 in the category '*org*'. The String Search method could only achieve this result in the categories (*org, room_scale*) twice in the Transcripts. The lowest value for the string search is also in the *unit_sub* category with a value of 0.76%.

As described in the introduction of this section, Table 4.15 shows the comparison between the annotated labels and the total number of labels that NER and the

		Accuracy	Predicted	Actual Labels
NER	nbr_scale	0.86	6	7
	org	0.50	1	2
String Search	nbr_scale	**1.00**	7	7
	org	**1.00**	2	2
	nbr_sub	0.75	6	8
	room_scale	**1.00**	3	3
	unit_scale	0.95	19	20
	unit_sub	0.71	5	7

Table 4.13: Summary Semantic Results.

		Accuracy	Predicted	Actual Labels
NER	nbr_scale	0.69	9	13
	org	**1.00**	1	1
String Search	nbr_scale	0.77	10	13
	org	**1.00**	1	1
	nbr_sub	0.80	4	5
	room_scale	**1.00**	9	9
	unit_scale	0.93	13	14
	unit_sub	0.76	19	25

Table 4.14: Transcript Semantic Results.

string search method could find. As seen, the models may have found more labels than were actually annotated. The bold numbers highlight the number of additional labels found by the model. NER was able to find more labels in both categories in the Transcripts, while in the Summaries more labels were found in one category (*nbr_scale*).

The String Search method was extracting additional labels in four classes (*org*, *room_scale*, *unit_scale*, *unit_sub*) in the Summaries, whereas in the Transcripts it found extra labels in only three classes (*org*, *nbr_sub*, *unit_scale*). In total, both models found more labels than originally given. In the Summary dataset, NER recorded a +1 label and in the Transcripts +6. String Search detected in the total of six different classes +19 additional labels in the Summaries and +23 additional in the Transcripts. The biggest increase in labels happened with the String Search method for both interviews in the class '*unit_scale*' (+10 Summary, +14 Transcript).

		Summary		Transcripts	
		Actual	Model	Actual	Model
NER	nbr_scale	7	**9**	13	**18**
	org	2	1	1	**2**
	Total	9	10 (+1)	14	20 (+6)
String Search	nbr_scale	7	7	13	11
	org	2	**3**	1	**6**
	nbr_sub	8	7	5	**17**
	room_scale	3	**7**	9	4
	unit_scale	20	**30**	14	**28**
	unit_sub	7	**12**	25	24
	Total	47	66 (+19)	67	90 (+23)

Table 4.15: Difference between Labels of Model and given Dataset.

4.3 Visualisation for End-Users

As mentioned in the Introduction, the ACE department, also the end-users, could use tree maps and word clouds for visualising the interview data. This however did not take into consideration the semantic and sentiment relationship between people, places and themes. With the predicted semantic and sentiment labels from the models implemented, the results can be used to help the end-user to investigate these relationships by transforming the results into visualisation.

Firstly, to clarify, the end-user can be divided into reference and stakeholder groups. The stakeholders are those who will use the models in their work, while the reference group consists of members that contribute to the thesis outcome and make sure that the stakeholder's needs are satisfied. Therefore, when creating the visualisations, workshops were facilitated with the reference group to align the expectations and technical possibilities. A final workshop with the stakeholders will be planned to explain how to use the visualisation results in depth.

First Draft

In Figure 4.1, a first draft of the visualisation for a selected Summary is shown. Here, the x-axis represents the number of sentences within the interview. The visualisation of the semantic models can be seen in the left y-axis and the bar chart. The stacked bars reflect how many labels of the respective topic were found in which sentence. The predicted sentiment can be read from the line shown and from the right y-axis. A value of -1 represents a negative semantic, 0 a neutral and 1 a positive one. The visualisation was implemented by using the Python `matplotlib` library and graphical decisions, such as colours, where made by the `matplotlib` stylesheet `'ggplot'`.

For example, in the presented figure, it can be seen that numerous labels from the class *unit_scale* were found in many sentences. For example, it can be seen that sentence number nine contains three different classes and a total of four labels. Furthermore, it can be seen that the neutral sentiment class occurs rather little and positive and negative are represented equally.

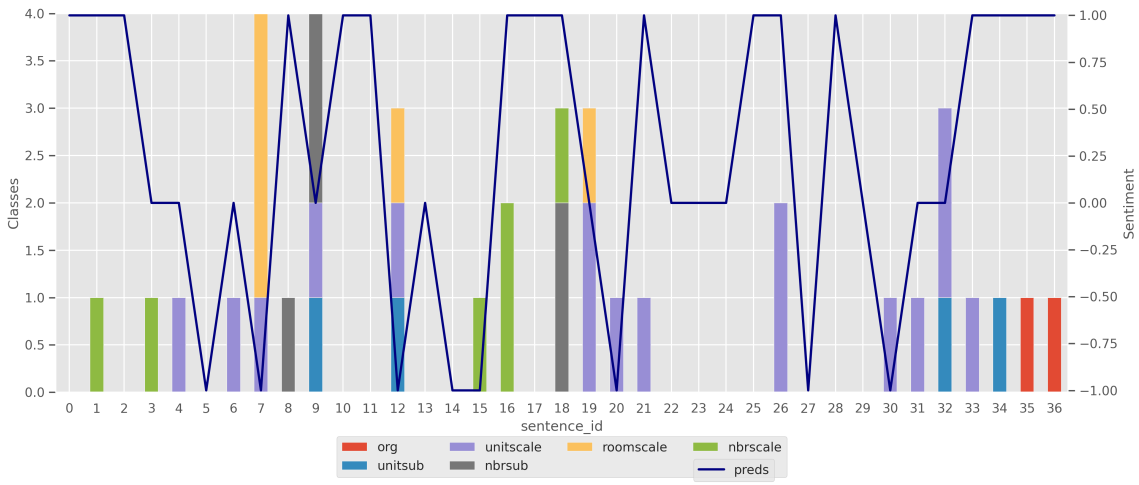

Figure 4.1: Draft visualisation

An example of the semantic and sentiment labels found for the sentences can be seen in Table 4.16 below.

Semantic Labels	Sentiment Labels	Sentence
Wiselgrensplatsen, Centrala Göteborg	Positive	Läget är guld, bara fem minuter till Wieselgrensplatsen och sen ytterligare några minuter in till centrala Göteborg.

Table 4.16: Model Prediction Results.

Final Visualisation

After taking into consideration continuous feedback given by the reference group, the final visualisation of the relationship between the semantic and sentiment orientation and the text is given below in Figure 4.2.

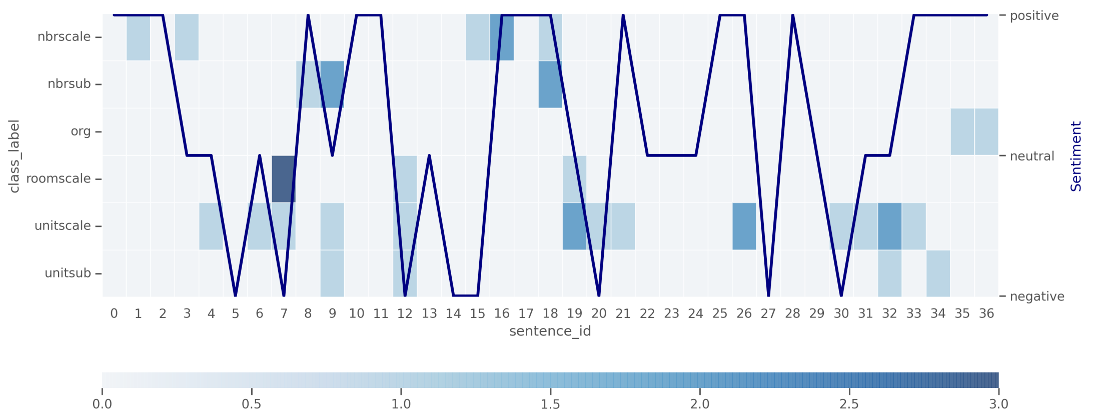

Figure 4.2: Final visualisation

Here, in Figure 4.2, the final visualisation for the same Summary as the draft visualisation is illustrated. The semantic classes and the number of labels mentioned within each class and sentence are now shown as a heatmap instead, where the darker the colour, the more instances of this class' labels have been mentioned. The colour bar at the bottom of the graph further describes this range of minimum and the maximum number of labels found per class per sentence. The x-axis still represents the number of sentences within the interview, and the right y-axis the Sentiment. However, the labels of the right y-axis have now been changed to positive, neutral and negative. This visualisation can then take into consideration the difference between the number of labels as it is scaled based on the colour now. From the draft visualisation, if some sentences had significantly more labels than others, the 'smaller' sentence would be drowned out as the graph is not scaled.

This visualisation was implemented by using the Python visualisation libraries `matplotlib` and `seaborn`.

Examples of visualisation for other interview samples can be found in Appendix A.4 and A.5.

5 Conclusion

In this chapter, first, the methods and results applied in this thesis will be discussed. Furthermore, the research questions proposed in Section 1.1 will be addressed and answered in Section 5.2 as well as a discussion of future work.

5.1 Discussion

Two models were developed to classify the sentiment and semantic orientation from qualitative interviews. This section presents a discussion of the results of the individual models.

5.1.1 Assumptions

First and foremost, in regards to the interview dataset provided by the ACE, different assumptions were made for the cleaning and pre-processing of the dataset as this could have had a significant impact on the training and evaluation of the models. The main assumption was that all the transcribed interviews contain only Swedish words. Through a quick manual check when annotating and processing the datasets, only fully Swedish interviews were found. However, not all 444 interviews were examined in detail, therefore, in reality, there may have been some interviews containing English words or are conducted in English. This concerns both the interview datasets that are transcribed as Transcripts and when interview Summaries contain direct quotes of the interviewees. This may in particular affect the performance of the sentiment DL models, as the retrospective annotated dataset may have included non-Swedish words and fed into the model for training again, which may not have been accurate.

Another assumption of the dataset is that there is no slang in the different datasets. This assumption however mainly concerns the semantic model where the String Search method was performed, as the different semantic target labels did not take into account their "slang" version.

As mentioned in the methodology, during the pre-processing of the dataset inconsistencies in how the interviews were transcribed were found within each Summary and Transcript dataset. The implementation of the pre-processing and cleaning of the text was based on the manual check that was performed. It was assumed that the deviations that were found were the only actual differences between the text and that the text would only be structured in these specific ways, but there may have been some instances where the variances were missed.

Additionally, in terms of the Sentiment model, since the real sentiment target values were not generated by professional annotators, a key assumption is that the three derivative classifiers within the Model 2 could be seen as substitutes for annotators. This is further emphasised in Table 4.2 and Figure 3.6. The three derivative models agreed on 84.41% and disagreed on 15.59% of the labels, similarly, the two authors of this paper agreed on 84.40% and disagreed on 15.60%. It can therefore be argued that the reliability of the three models can be equivalent to humans.

5.1.2 Sentiment Analysis

Table 5.1 presents a Summary of the model performance in terms of accuracy and F1-score for the different sentiment models. As the final evaluation score of the models, the Transcript model performed the best with an accuracy of 0.85 and F1-score of 0.80. The Summary model, on the other hand, only had an accuracy and F1-score of 0.65. It can also be seen that the final Transcript model performed worse after adding in the retrospective annotations while the final Summary model performed better with more annotations. These differences in performance could be attributed to the amount and class distribution of the datasets used for fine-tuning and evaluating the models.

	Retrospective Annotations			
	Without		With	
	Accuracy	F1-score	Accuracy	F1-score
Summary	0.60	0.61	0.65	0.65
Transcript	0.87	0.82	0.85	0.80

Table 5.1: Metric Comparison for Sentiment Models.

To further explain, the class distribution of negative, neutral, and positive of the annotated dataset can be seen to be significantly different for the Summary and Transcript models. As illustrated in Figure 3.3, the Summary dataset has a balanced distribution between the negative, neutral and positive classes, whilst the Transcript dataset mainly consists of the neutral class at 66.49%. For that reason, the weighted average is looked at when evaluating the F1-scores for the Summary, while the macro F1-score is inspected for Transcripts. This is because all classes are equally important, no matter the number of instances per class, as such the macro F1 is used for Transcripts and weighted for Summary. This is further emphasised in the Classification Reports of the test sets for both models and training datasets (with or without the Retrospective annotations). There is a negligible difference between the macro averaged F1-score and the weighted average F1-score for the Summary datasets, as seen in Table 4.6 and Table 4.9, and therefore, the class distribution does not have a big impact on the results. On the other hand, as seen in the Classification Reports for the Transcript class in Tables 4.8 and 4.11, there is around a 4-7% difference between the macro and weighted average for the precision, recall and F1-score. This stresses that the model classifies sentences that belong to the classes with a larger distribution with higher confidence. In this case, the model is highly biased towards the neutral class. The neutral class had the highest F1-score and the highest precision and recall score, at 0.93 and 0.92 for both training sets.

The class distribution may also have an effect on the performance of the models for the data sets with and without the retrospective annotations. As stated earlier, from Table 5.1, the Summary model performed better with the additional annotations. However, the Transcript model performed worse. Referring here to Table 4.11, this was attributed to the model now predicting five actual negative sentences as neutral, resulting in a lower precision for the neutral and lower recall for the negative class. In comparison to Table 4.7 only two actual negatives were predicted as neutral. It can be argued that with more training instances in the neutral class, the model will overfit towards the majority class and thus predict the instances as the neutral class. Consequently, more annotations may not always lead to better results. This also highlights a weakness of the DL models, where it is a black box and one cannot fully understand how they predict.

In supervised learning for ML and DL models, the quality and quantity of the target labels are also important to consider, as the models will learn based on the target labels provided. In this thesis, for both the Summary and Transcript dataset, the annotated set does not even reach 2-3% of the total data, as illustrated in Figures 3.1 and 3.2. Moreover, the performance of the sentiment models is all based on the annotations from the authors as the target value. The annotation process could thus have influenced the evaluation results. It should be emphasised that the authors do not have the expertise within qualitative coding for social sciences and are not linguistic experts. Therefore, the validity of the model can be questioned. On the other hand, in this use-case, whether the model actually performs well or not depends on the end-user.

5.1.3 Semantic Analysis

For the semantic analysis, a list with a total of 190 labels was provided, which can be divided into six different classes. In addition, the test set, which contains 141 sentences, was annotated with corresponding labels. Theoretically, each set could contain between zero and 190 labels, resulting in a total of between zero and 26,790 labels. Since the annotated data set does not represent complete target values, but merely serves as a sample value, the validity of the results is open to discussion. Nevertheless, they provide informative insights into the performance of the models, which are discussed in more detail below.

Comparing the performance of NER and String Search, it should be noted that only the classes *nbr_scale* and *org* can be compared. This is because NER is only used to identify the entities *LOC* and *ORG* which are the *nbr_scale* and *org* semantic classes, as mentioned in Section 3.3.2. For the Summary Category, Table 4.13 shows that String Search has an accuracy of 1 in both classes, whereas NER does not reach this target value. The highest value is 0.86 in the *nbr_scale* class, whereas in the *org* class just half the value of String Search was achieved (0.5 NER, 1 String Search). The accuracy score may exaggerate its reflection of the poor performance as in terms of the count of labels, NER did not manage to identify only one label in each class. However, it can still be concluded that with the given data, String Search performs better than NER.

Table 4.14 for the Transcripts shows a performance change for NER. Those classes that performed poorly in the Summaries (*org*) now have an ideal value of *accuracy* = 1. This score has also remained the same for String Search. At the same time, the value in the previously stronger class (*nbr_scale*) for NER drops to a value of 0.69. Although the value of String Search in the class *nbr_scale* also drops, this model shows an overall higher accuracy value. It can therefore be assumed that String Search also outperforms NER in the Transcript category.

Possible reasons why NER performed worse than String Search in both interview categories in the *nbr_scale* class could be due to the tokenisation of the words by NER. A thorough investigation revealed that NER divides the word '*centrala Göteborg*' into two individual tokens, '*centrala*' and '*Göteborg*'. Since the given label is seen as a single token, the two would not match. As a result, it would be marked as a missing word and thus lower the accuracy. This could also be the reason why the evaluation value in the *nbr_scale* class is significantly lower in the Transcripts than in the Summaries. In the Summaries, 7 actual labels are given, 6 of which were found by the model and thus achieved an accuracy value of 0.86. The Transcripts have an additional four *nbr_scale* labels and thus a total of 13, of which only 9 were found and so the accuracy value dropped to 0.69. A possible reason could be that the additional four words are those that were split up by the model and thus could not be found.

In general, looking at the overall amount labels that the models found in Table 4.15, both NER and String Search were able to identify additional labels in the test set. In the Summary category, NER finds +2 labels in the *nbr_scale* class, while String Search finds no additional labels in the same class. On the other hand, in the *org* category, NER found one value less than actually given, while String Search finds a +1 label. Similar performance can be seen in the Transcripts. Here as well, NER finds more additional labels in *nbr_scale* and less in *org* compared to String Search, whereas it is the other way round for String Search (*nbr_scale* -2, *org* +5). Possible reasons why NER performs worse in *org* than String Search could be that research has shown that NER does not recognise the organisation '*Poseidon*'. Based on this, it can be assumed that the word is simply not included in the vocabulary, as in this case, it is a very specific local company in Gothenburg rather than being part of the general scope of a vocabulary. It is interesting to see here that both models perform differently in the classes and can thus complement each other well.

To conclude this section of the discussion, although NER performs worse overall than String Search in terms of accuracy for both Summaries and Transcripts for the given labels, it performs better than String Search when it comes to finding additional labels for the *nbr_scale* class. Additionally, Table 4.15 shows that both models complement each other well when it comes to finding additional information. It should also be mentioned that the NER has not been fine-tuned and therefore has potential for improvement, but this would require a data set that has been fully annotated with the relevant NER tags of the respective domain. Nevertheless, the results show that String Search serves as a good backbone for not missing additional labels.

5.2 Conclusion

To conclude, as stated in the Aim in Section 1.1, the original aim of this thesis was to analyse the interview data provided by the ACE using statistical, ML and/or DL technologies based on their sentiment and semantics. The following conclusion will thus be addressed through the research questions presented in Section 1.1. For this purpose, the sub-questions are answered first, from which the answer to the main research question can be derived.

Sub-question: 1) Which state-of-art algorithms can be adopted and what are the pros and cons?

For the semantic analysis of this thesis, BERT was implemented (see Chapter 3) and its performance was investigated in Chapter 4. One advantage of BERT and the introduced Hugging Face library is that numerous pre-trained models are available. However, due to the fact that the data in this thesis used the Swedish language, it limits the number of options. Although pre-trained models can be used very well in their original state, a lot of annotated data is needed to fine-tune and adapt them to the domain. As shown in Section 3.1.2, this, in turn, requires qualified staff and concrete rules, as obstacles can arise in label annotation. The model is only as reliable as the annotators of the target labels. A similar problem occurs when using the NER method. Again, a fully detailed annotated dataset is needed to fit the model to the domain. However, as discussed in Section 3.3.3 it was not possible within the scope of this thesis. In addition, the String Search method was implemented, which serves as a backbone to NER and whose performance was discussed in Section 5.1.2 of Chapter 5.

To answer the sub-question finally, in this thesis we have shown that the implementation of BERT, NER and the String Search is one successful way to extract the sentiment and semantics from interview data.

Sub-question 2) How can we visualise the semantic and sentiment analysis of interview data?

The answer to this sub-question is given in Section 4.3. As requested by the end-user, the sentiment and semantics are visualised for each individual interview. Generally, it can be stated that all information is presented within one image. For the technical implementation, the Python library `matplotlib` was used.

Research Question: How can state-of-art techniques be adjusted and applied to classify sentiment and semantic orientation from qualitative interviews, in order to provide city planners and researchers with an accurate overview of the data?

Accordingly, the answer to the final research question is partially included in sub-question 1 and 2. Overall, to implement state-of-art techniques to classify sentiment and semantic orientation from qualitative interviews, it is important to adapt the model accordingly to the given data. This happens through steps such as data pre-processing and cleaning, model fine-tuning, and proper evaluation. An important factor in the performance of the models is based on the given dataset, as DL models are heavily data-driven. Specifically, accurate target values are needed to properly fine-tune. In this thesis, for the sentiment model and semantic NER model, BERT was used as the core algorithm to train the models. Additionally, due to missing real semantic target values, String Search was implemented in the semantic model as a backbone method.

5.2.1 Future Work

In terms of future work, in this project, there are some areas of improvement that can be made on the dataset as well as the individual sentiment and semantic analysis.

Firstly, the main reason why interviews were conducted to uncover the perception is to avoid excluding certain demographics. An alternative approach would have been to only include data from online sources, such as social media. However, this would lead to social and economic exclusion. For example, this would assume that every person in the urban area would have digital access as well as the know-how of using digital products. Nevertheless, adding online data could be a course of action that can be further investigated.

Second, a proper and larger annotated dataset for both sentiment and semantic classifiers would most probably increase the training and evaluation performance. That is, using expert knowledge to annotate and label the dataset would decrease the overall human error that was created in this project. To continue the investigation on a sentence level, the data has to be split accordingly. Furthermore, each sentence should be labelled with the respective sentiment (negative, positive, neutral) and the semantic labels (house, room, etc.) if they occur within the text.

Third, when creating this annotated dataset, it is also important to take into consideration the way the interview documents are transcribed. As mentioned in the Data Pre-processing and also Assumptions Sections, the interview data obtained did not follow a strict format making the cleaning and pre-processing of the data more complex and tedious. Therefore, for future work, it may be good to consider transcribing the interviews in a standard format, so that all are transcribed in the same uniform way whether it is done by one or more people. This way, the data could be more structured and informative for both the analysis and use in different ML and DL techniques. With less variation and assumptions made about the underlying interview data, the reliability and validity of the results could also be seen to increase.

Fourth, as mentioned in the discussion, the semantic NER models could be fine-tuned if there is data that would allow for training in this specific domain. For example, by including the other semantic classes such as nbr_sub, room_scale, unit_scale and unit_sub. However, this data would need to be annotated on a word-level, where each mention of the entity has to be tagged in the sentence.

Fifth, for the sentiment analysis, Transcript training and test set could contain a more equal distributed data. Although an appropriate evaluation metric such as the macro F1 was used to assess the results of the classes equally, the model tends to overfit towards the largest sentiment class and therefore might affect the result more than assumed.

Sixth, the sentiment model is currently only based on one specific pre-trained BERT model, the KB-BERT. Other pre-trained BERT models on the Swedish corpus could be investigated and used for further exploration.

Lastly, another area for investigation would be to fine-grade the analysis from a sentence to a word level. Here, one could research the word relationship to other words, for example, using aspect-based sentiment analysis and also perform word sense disambiguation. However, this would mean that a more specific and correctly labelled dataset would be needed where targets and other grammatical indicators are annotated for the full text.

Bibliography

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[3] J. Hirschberg and C. D. Manning, “Advances in natural language processing,” *Science*, vol. 349, no. 6245, pp. 261–266, 2015.

[4] H. Zhang, W. Gan, and B. Jiang, “Machine learning and lexicon based methods for sentiment classification: A survey,” in *2014 11th web information system and application conference*, pp. 262–265, IEEE, 2014.

[5] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, “Deep learning–based text classification: a comprehensive review,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–40, 2021.

[6] “68% of the world population projected to live in urban areas by 2050, says un | un desa department of economic and social affairs.” `https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html`, May 2018.

[7] M. A. Kuddus, E. Tynan, and E. McBryde, “Urbanization: a problem for the rich and the poor?,” *Public health reviews*, vol. 41, no. 1, pp. 1–4, 2020.

[8] “Urban planning.” `https://smartcitysweden.com/focus-areas/urban-planning/`, May 2021.

[9] L. Carmichael, T. G. Townshend, T. B. Fischer, K. Lock, C. Petrokofsky, A. Sheppard, D. Sweeting, and F. Ogilvie, “Urban planning as an enabler of urban health: challenges and good practice in england following the 2012 planning and public health reforms,” *Land use policy*, vol. 84, pp. 154–162, 2019.

[10] M. Selim, “The effects of planned urban development on the comfort of healthy urban living,” in *2021 Third International Sustainability and Resilience Conference: Climate Change*, pp. 286–292, IEEE, 2021.

[11] H. Barton and M. Grant, “Urban planning for healthy cities,” *Journal of urban health*, vol. 90, no. 1, pp. 129–141, 2013.

[12] B. Giles-Corti, A. Vernez-Moudon, R. Reis, G. Turrell, A. L. Dannenberg, H. Badland, S. Foster, M. Lowe, J. F. Sallis, M. Stevenson, *et al.*, “City planning and population health: a global challenge,” *The lancet*, vol. 388, no. 10062, pp. 2912–2924, 2016.

[13] H. S. Andersen, “Why do residents want to leave deprived neighbourhoods? the importance of residents’ subjective evaluations of their neighbourhood and its reputation,” *Journal of Housing and the Built Environment*, vol. 23, no. 2, pp. 79–101, 2008.

[14] C. L. Ambrey, C. M. Fleming, and M. Manning, “Perception or reality, what matters most when it comes to crime in your neighbourhood?,” *Social indicators research*, vol. 119, no. 2, pp. 877–896, 2014.

[15] G. Liu and J. Guo, “Bidirectional lstm with attention mechanism and convolutional layer for text classification,” *Neurocomputing*, vol. 337, pp. 325–338, 2019.

[16] Q. T. Ain, M. Ali, A. Riaz, A. Noureen, M. Kamran, B. Hayat, and A. Rehman, “Sentiment analysis using deep learning techniques: a review,” *Int J Adv Comput Sci Appl*, vol. 8, no. 6, p. 424, 2017.

[17] A. Gasparetto, M. Marcuzzo, A. Zangari, and A. Albarelli, “A survey on text classification algorithms: From text to predictions,” *Information*, vol. 13, no. 2, p. 83, 2022.

[18] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, “Text classification algorithms: A survey,” *Information*, vol. 10, no. 4, p. 150, 2019.

[19] J. Howard and S. Ruder, “Universal language model fine-tuning for text classification,” *arXiv preprint arXiv:1801.06146*, 2018.

[20] S. Poria, E. Cambria, and A. Gelbukh, “Aspect extraction for opinion mining with a deep convolutional neural network,” *Knowledge-Based Systems*, vol. 108, pp. 42–49, 2016.

[21] N.-C. Chen, M. Drouhard, R. Kocielnik, J. Suh, and C. R. Aragon, “Using machine learning to support qualitative coding in social science: Shifting the focus to ambiguity,” *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 8, no. 2, pp. 1–20, 2018.

[22] M. Parmar, B. Maturi, J. M. Dutt, and H. Phate, “Sentiment analysis on interview transcripts: An application of nlp for quantitative analysis,” in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1063–1068, IEEE, 2018.

[23] M. Rambocas, J. Gama, *et al.*, “Marketing research: The role of sentiment analysis,” tech. rep., Universidade do Porto, Faculdade de Economia do Porto, 2013.

[24] G. Tsoumakas and I. Katakis, “Multi-label classification: An overview,” *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 3, no. 3, pp. 1–13, 2007.

[25] C. S. Khoo and S. B. Johnkhan, “Lexicon-based sentiment analysis: Comparative evaluation of six sentiment lexicons,” *Journal of Information Science*, vol. 44, no. 4, pp. 491–511, 2018.

[26] A. Abirami and V. Gayathri, “A survey on sentiment analysis methods and approach,” in *2016 Eighth International Conference on Advanced Computing (ICoAC)*, pp. 72–76, IEEE, 2017.

[27] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, "Stanza: A python natural language processing toolkit for many human languages," *arXiv preprint arXiv:2003.07082*, 2020.

[28] D. Khyani, B. Siddhartha, N. Niveditha, and B. Divya, "An interpretation of lemmatization and stemming in natural language processing," *Shanghai Ligong Daxue Xuebao/Journal of University of Shanghai for Science and Technology*, vol. 22, pp. 350–357, 2020.

[29] H. H. Do, P. Prasad, A. Maag, and A. Alsadoon, "Deep learning for aspect-based sentiment analysis: a comparative review," *Expert Systems with Applications*, vol. 118, pp. 272–299, 2019.

[30] B. Liu, "Sentiment analysis and opinion mining," *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[31] Sonia, "Opinion mining techniques and its applications: A review," in *Proceedings of First International Conference on Computing, Communications, and Cyber-Security (IC4S 2019)* (P. K. Singh, W. Pawłowski, S. Tanwar, N. Kumar, J. J. P. C. Rodrigues, and M. S. Obaidat, eds.), (Singapore), pp. 549–559, Springer Singapore, 2020.

[32] T. A. Rana and Y.-N. Cheah, "Aspect extraction in sentiment analysis: comparative analysis and survey," *Artificial Intelligence Review*, vol. 46, no. 4, pp. 459–483, 2016.

[33] Y. Liu, X. Huang, A. An, and X. Yu, "Arsa: a sentiment-aware model for predicting sales performance using blogs," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 607–614, 2007.

[34] M. Joshi, D. Das, K. Gimpel, and N. A. Smith, "Movie reviews and revenues: An experiment in text regression," in *Human language technologies: The 2010 annual conference of the North American chapter of the Association for Computational Linguistics*, pp. 293–296, 2010.

[35] S. Asur and B. A. Huberman, "Predicting the future with social media," in *2010 IEEE/WIC/ACM international conference on web intelligence and intelligent agent technology*, vol. 1, pp. 492–499, IEEE, 2010.

[36] W. Zhang and S. Skiena, "Trading strategies to exploit blog and news sentiment," in *Fourth international aAAI conference on weblogs and social media*, 2010.

[37] A. Yadav and D. K. Vishwakarma, "Sentiment analysis using deep learning architectures: a review," *Artificial Intelligence Review*, vol. 53, no. 6, pp. 4335–4385, 2020.

[38] R. Catelli, S. Pelosi, and M. Esposito, "Lexicon-based vs. bert-based sentiment analysis: A comparative study in italian," *Electronics*, vol. 11, no. 3, p. 374, 2022.

[39] S. T. Kokab, S. Asghar, and S. Naz, "Transformer-based deep learning models for the sentiment analysis of social media data," *Array*, p. 100157, 2022.

[40] A. U. Rehman, A. K. Malik, B. Raza, and W. Ali, "A hybrid cnn-lstm model for improving accuracy of movie reviews sentiment analysis," *Multimedia Tools and Applications*, vol. 78, no. 18, pp. 26597–26613, 2019.

[41] N. Palm, "Sentiment classification of swedish twitter data," 2019.

[42] F. Wu and Y. Huang, "Sentiment domain adaptation with multiple sources," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 301–310, 2016.

[43] J. Brownlee, *Deep learning for natural language processing: develop deep learning models for your natural language problems.* Machine Learning Mastery, 2017.

[44] S. A. Salloum, R. Khan, and K. Shaalan, "A survey of semantic analysis approaches," in *The International Conference on Artificial Intelligence and Computer Vision*, pp. 61–70, Springer, 2020.

[45] G. for Geeks, "Understanding semantic analysis – nlp," 2021.

[46] R. Navigli, "Word sense disambiguation: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 2, pp. 1–69, 2009.

[47] J. Leng and P. Jiang, "A deep learning approach for relationship extraction from interaction context in social manufacturing paradigm," *Knowledge-Based Systems*, vol. 100, pp. 188–199, 2016.

[48] J. Huang, "Ce standard documents keyword extraction and comparison between different machinelearning methods," 2018.

[49] S. Beliga, "Keyword extraction: a review of methods and approaches," *University of Rijeka, Department of Informatics, Rijeka*, vol. 1, no. 9, 2014.

[50] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic keyword extraction from individual documents," *Text mining: applications and theory*, vol. 1, pp. 1–20, 2010.

[51] S. Ghannay, A. Caubriere, Y. Esteve, A. Laurent, and E. Morin, "End-to-end named entity extraction from speech," *arXiv preprint arXiv:1805.12045*, 2018.

[52] C. Lothritz, K. Allix, L. Veiber, T. F. Bissyandé, and J. Klein, "Evaluating pretrained transformer-based models on the task of fine-grained named entity recognition," in *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 3750–3760, 2020.

[53] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 50–70, 2020.

[54] A. Goyal, V. Gupta, and M. Kumar, "Recent named entity recognition and classification techniques: a systematic review," *Computer Science Review*, vol. 29, pp. 21–43, 2018.

[55] V. Yadav and S. Bethard, "A survey on recent advances in named entity recognition from deep learning models," *arXiv preprint arXiv:1910.11470*, 2019.

[56] P. Sun, X. Yang, X. Zhao, and Z. Wang, "An overview of named entity recognition," in *2018 International Conference on Asian Language Processing (IALP)*, pp. 273–278, IEEE, 2018.

[57] A. P. Quimbaya, A. S. Múnera, R. A. G. Rivera, J. C. D. Rodríguez, O. M. M. Velandia, A. A. G. Peña, and C. Labbé, "Named entity recognition over electronic health records through a combined dictionary-based approach," *Procedia Computer Science*, vol. 100, pp. 55–61, 2016.

[58] K. Mishev, A. Gjorgjevikj, I. Vodenska, L. T. Chitkushev, and D. Trajanov, "Evaluation of sentiment analysis in finance: from lexicons to transformers," *IEEE access*, vol. 8, pp. 131662–131682, 2020.

[59] J. Alammar, “The illustrated transformer,” 2018.

[60] D. Jurafsky and J. H. Martin, “Speech & language processing,” 2021.

[61] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[62] A. Hassan and A. Mahmood, “Convolutional recurrent deep learning model for sentence classification,” *IEEE Access*, vol. 6, pp. 13949–13957, 2018.

[63] Z. Cao, S. Li, Y. Liu, W. Li, and H. Ji, “A novel neural topic model and its supervised extension,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, 2015.

[64] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* " O’Reilly Media, Inc.", 2019.

[65] Z. C. Lipton, C. Elkan, and B. Narayanaswamy, “Thresholding classifiers to maximize f1 score,” *arXiv preprint arXiv:1402.1892*, 2014.

[66] “Python package index - pypi.” `https://pypi.org/`.

[67] C. Tran, “Tutorial: Fine-tuning bert for sentiment analysis - by skim ai.” `https://skimai.com/fine-tuning-bert-for-sentiment-analysis/`, Dec 2020.

[68] M. Malmsten, L. Börjeson, and C. Haffenden, “Playing with words at the national library of sweden–making a swedish bert,” *arXiv preprint arXiv:2007.01658*, 2020.

[69] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, *et al.*, “Huggingface’s transformers: State-of-the-art natural language processing,” *arXiv preprint arXiv:1910.03771*, 2019.

[70] S. University and U. University, “Stockholm-umeå corpus 3.0.” `https://spraakbanken.gu.se/en/resources/suc3`, 2012.

A

Appendix

A.1 Example of Interview Transcript

Telefonintervju med PXXX 2021-05-11
Samtalets längd: 6 minuter
Intervjuare: XYZ
Ljudfil finns

XYZ: Varför valde du att flytta från Träskilsgatan?
För att jag köpt ett hus
XYZ: Bor du ensam eller är det flera i hushållet?
Jag bor själv.
XYZ: Har det något med renoveringen att göra att du flytta just nu?
Jag vet inte vad det är för renovering du pratar om.
XYZ: På Träskilsgatan har de väl bytt fönster eller håller på att byta fönster, och sedan ska man göra en stamrenovering och renovera badrummen.
Det visste jag faktiskt inte.
XYZ: Det är ganska stort. Det kan hända att de har börjat i en ände och inte hunnit kommit dit där du bor ännu.
Ok det fick jag ingen information om när jag bodde där.
XYZ: Så du har redan flyttat?
Ja.
XYZ: Får jag lov att fråga vart du köpt hus någonstans?
Varberg.
XYZ: Så då är det jobb och andra saker som har bytts också då?
Jag jobbar fortfarande i Göteborg.
XYZ: Jag förstår. Det blir pendling i stället.
Ja det kan man säga.
XYZ: Då vet du inte om man skulle göra något för att spara energi när man renoverade vid Träskilsgatan.
Nej jag flyttade i slutet på februari och då hade jag inte hört något om fönster eller stambyte. Den lägenheten som jag bodde i var ganska nedgången. Jag vet ingenting om renoveringarna tyvärr.
XYZ: Hur stor var lägenheten som du flyttade ifrån?
Den var 52 någonting kvadratmeter.
XYZ: En tvåa. Har du större nu när du har köpt hus?
XYZ: Är det högre kostnader nu än vad det var innan?

Ja det är högre kostnader nu än det var innan. Det var ganska billig hyra där på Träskilsgatan.
XYZ: Är det någonting du kommer sakna från träskilsgatan?
Nej det tror jag inte. Läget var ju bra, men annars tror jag inte att jag kommer sakna något.
XYZ: Vi brukar ställa lite demografiska frågor till vår studie. Har du möjlighet att svara på hur gammal du är, vilken sysselsättning du har och din utbildningsnivå?
Jag är 35 år gammal. Anställd på heltid. Har gått gymnasiet och eftergymnasial yrkesutbildning.
XYZ: Var utbildningen 1,5 år eller 3?
Den var 1,5.
XYZ: Var det någon anledning till att du flyttade till Varberg? Hade du några kontakter där redan eller var det där du hittade något som verkade trevligt?
Jag tycker det är fint i Varberg och det är trevligt att vara nära naturen.
XYZ: Jag ska inte störa dig mycket mer för du kände inte till renoveringen så mycket. Du är helt anonym i denna studie och har fått nummer 441. Vi har bara din gamla adress här och vet inte om du vill att vi skickar dig någon information om vår studie. I så fall behöver jag en ny postadress eller möjligtvis en e-postadress.
Nej jag är inte intresserad faktiskt om jag ska vara ärlig.
XYZ: Förstår det eftersom vi mest fokuserar på de som flyttar på grund av renoveringen. Det är intressant för vi har fått ett antal adresser och då behöver vi veta de olika anledningarna till att folk flyttar.
Vad finns det för mål med detta?
XYZ: Det finns en debatt om att hyresgäster känner sig bortträngda när det renoveras.
För att det blir dyrare då eller?
XYZ: Exakt. För att det blir dyrare, men också andra anledningar som exempelvis störningar. Vi har en del sådana också som flyttar bara därför. Jag får tacka dig för att du tog dig tid att svara på detta och önska dig lycka till i det nya boendet.
Ja tack ska du ha.
XYZ: Ha det fint!
Hej.

A.2 Example of Interview Summary

PXX 181029

Telefonintervju då personen är avflyttad och inte har bott i lägenheten på ett år.

PXX bor i Västerås sedan ett år tillbaka och har under tiden hyrt ut lägenheten till vänner. De behövde den dock inte längre och därför säger hon upp den.

Hon skaffade lägenheten i Frölunda 2012. Hon hade då köat länge, i mer än fem år, och tycker att hon hade tur som fick den. Hon var den första tiden i köandet student men hade sen fått jobb och inkkomst och tror att det spelade in i att hon fick den.

Hon tyckte mycket om lägenheten, en trea på 68 m2. Läget nära Frölunda torg var bra, en stor omsluten innergård gjorde det lugnt och tyst. Det känns nära till allt, både med buss och spårvagn.

Från början bodde hon där med sin sambo, men när det tog slut behöll hon lägenheten. Nu har hon flyttat till Västerås och flyttat in till sin nye kille i en ägd lägenhet och bildat familj. Hon kommer inte att flytta tillbaka till Göteborg. Kostnaden är ungefär densamma för det nya boendet som för det gamla.

I lägenheten har hon tapetserat om det lilla sovrummet och bett Bostadsbolaget att tapetsera det stora. Det var slitet. Lägenheten var annars inte renoverad så länge hon bodde där och när hon pratat med andra i huset har det inte varit renoverat på många år. Hon har hört att det ska renoveras, kanske kök och bad, men det är oklart när det ska hända. Hon är positiv till renoveringen. Det behöver fräschas upp. Vissa har renoverat själva menar hon, de som har råd och inte orkar vänta på att allt renoveras. Tittar man på bottenplan kan man se att vissa har renoverat köken t ex. Annars har Bostadsbolaget bara renoverat om man bett om något särskilt eller om det har blivit något fel, t ex vattenläckor. Hon har inte hört det pratas om oro för högre hyror i samband med renovering.

Bostadsbolaget har funkat jättebra, särskilt kvartersvärden. De har alltid ställt upp.

Området känns tryggt. Hållplatserna efter Frölunda torg åker man helst inte efter att det blivit mörkt och går inte ensam där på kvällen.

A.3 Semantic Class Labels

Housing related organisations (org):
['BB', 'Bostadsbolaget', 'Västra Linden', 'Gravlunds', 'Familjebostäder', 'Poseidon', 'Vesterstaden']

Physical components within a unit (unit_sub):
['frys', 'fasad', 'spis', 'bovärd', 'kakel', 'hiss', 'balkong', 'tap', 'värme', 'ventilation', 'trapphus', 'halvtrappa', 'golv', 'kökslucka', 'kyl', 'fläkt', 'varmvatten', 'vägg', 'hyresgäst', 'kylskåp', 'fönster', 'energihushålla', 'golv', 'stambyte', 'garage', 'bricka', 'rör']

Housing units (unit_scale):
['bostadsrätt', 'kolonistuga', 'hus', 'bottenplan', 'byggnad', 'lägenhet', 'lägenhetshus', 'provlägenhet', 'rum', 'hyresrätt']

Objects belonging to the neighbourhood (nbr_sub):
['natur', 'promenera', 'underhållning', 'fasad', 'kollektivtrafik', 'bus', 'kollektiv', 'grann', 'utemiljö', 'hunddagmatte', 'förskola', 'grusgång']

Rooms of a unit (room_scale):
['matrum', 'stam', 'badrum', 'sovrum', 'källare', 'garderob', 'tvättstuga', 'vardagsrum', 'kök', 'hall', 'trapphus', 'balkong', 'miljörum', 'rum', 'vind']

Neighbourhoods in Gothenburg (nbr_scale):
['Stigberget', 'Skärgården', 'Kvillebäcken', 'Frölunda Torg', 'Vasastaden', 'Kaverös', 'Annedal', 'Smörslottsgatan', 'Länsmansgården', 'Grimmered', 'Linnarhult', 'Hisingen', 'Eriksbo', 'Kannebäck', 'Grevegården', 'Hjällbo', 'Svartedalen', 'Långedrag', 'Skattegården', 'Gamlestaden', 'Centrala Göteborg', 'Södra Kortedala', 'Näverlursgatan', 'Sanna', 'Brunnsbo', 'Tuve', 'Kärralund', 'Järnbrott', 'Gårdstensberget', 'Beväringsgatan', 'Inom Vallgraven', 'Vårmånadsgatan', 'Lunden', 'Mölndal', 'Ramberget', 'Slättadamm', 'Norra Kortedala', 'Tofta', 'Olskroken', 'Majorna', 'Askim', 'Lundby', 'Skogome', 'Överås', 'Bratthammar', 'Kalendervägen', 'Östra Bergsjön', 'Redbergslid', 'Västra Bergsjön', 'Bergum', 'Johanneberg', 'Säve', 'Björlanda', 'Kärra', 'Högsbohöjd', 'Södra Biskopsgården', 'Jättesten', 'Näset', 'Skår', 'Hovås', 'Björkekärr', 'Kyrkby', 'Masthugget', 'Hammarkullen', 'Norra Biskopsgården', 'Arendal', 'Landala', 'Agnesberg', 'Torpa', 'Stampen', 'Doktor Westrings gata', 'Husargatan', 'Kville', 'Eketrägatan', 'Krokslätt', 'Guldheden', 'Hjuvik', 'Nolered', 'Skanstorget', 'Haga', 'Långängen', 'Kortedala', 'Bagaregården', 'Änggården', 'Skåne', 'Flatås', 'Ängås', 'Gunnilse', 'Kungsladugård', 'Gårdsten', 'Lorensberg', 'Billdal', 'Olivedal', 'Lövgärdet', 'Utby', 'Jättestensgatan', 'Ruddalen', 'Rambergsstaden', 'Guldringen', 'Högsbo', 'Kärrdalen', 'Fiskebäck', 'Kallebäck', 'Angereds Centrum', 'Kyrkbyn', 'Rödbo', 'Heden', 'Hagen', 'Skälltorp', 'Eriksberg', 'Lindholmen', 'Backa', 'Rannebergen', 'Skäpplandsgatan', 'Wieselgrensplatsen', 'Kålltorp', 'Härlanda', 'Önnered', 'Högsbotorp']

A.4 Additional Visualisations of Interview Documents

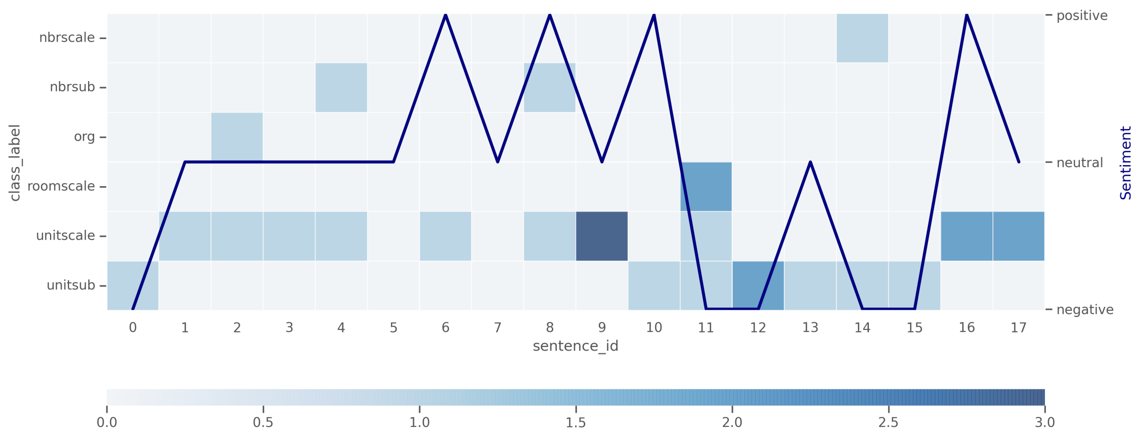

Figure A.1: Heatmap for Summary Version B

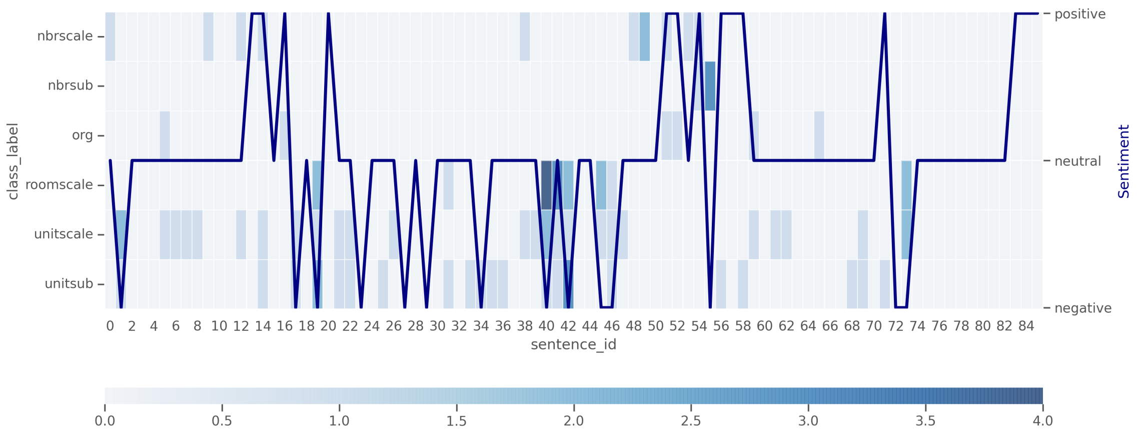

Figure A.2: Heatmap for Interview

A.5 Descriptive Analysis of Semantic Class Labels

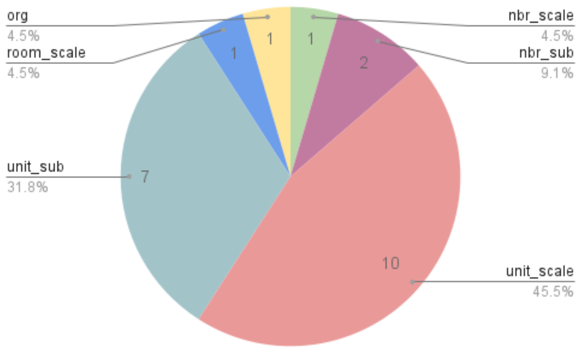

Figure A.3: Label Distribution for Summary Version B

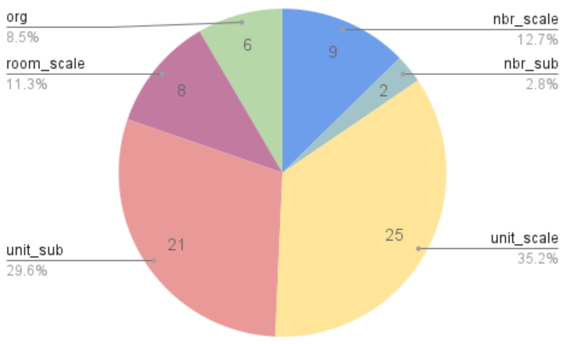

Figure A.4: Label Distribution for Interview