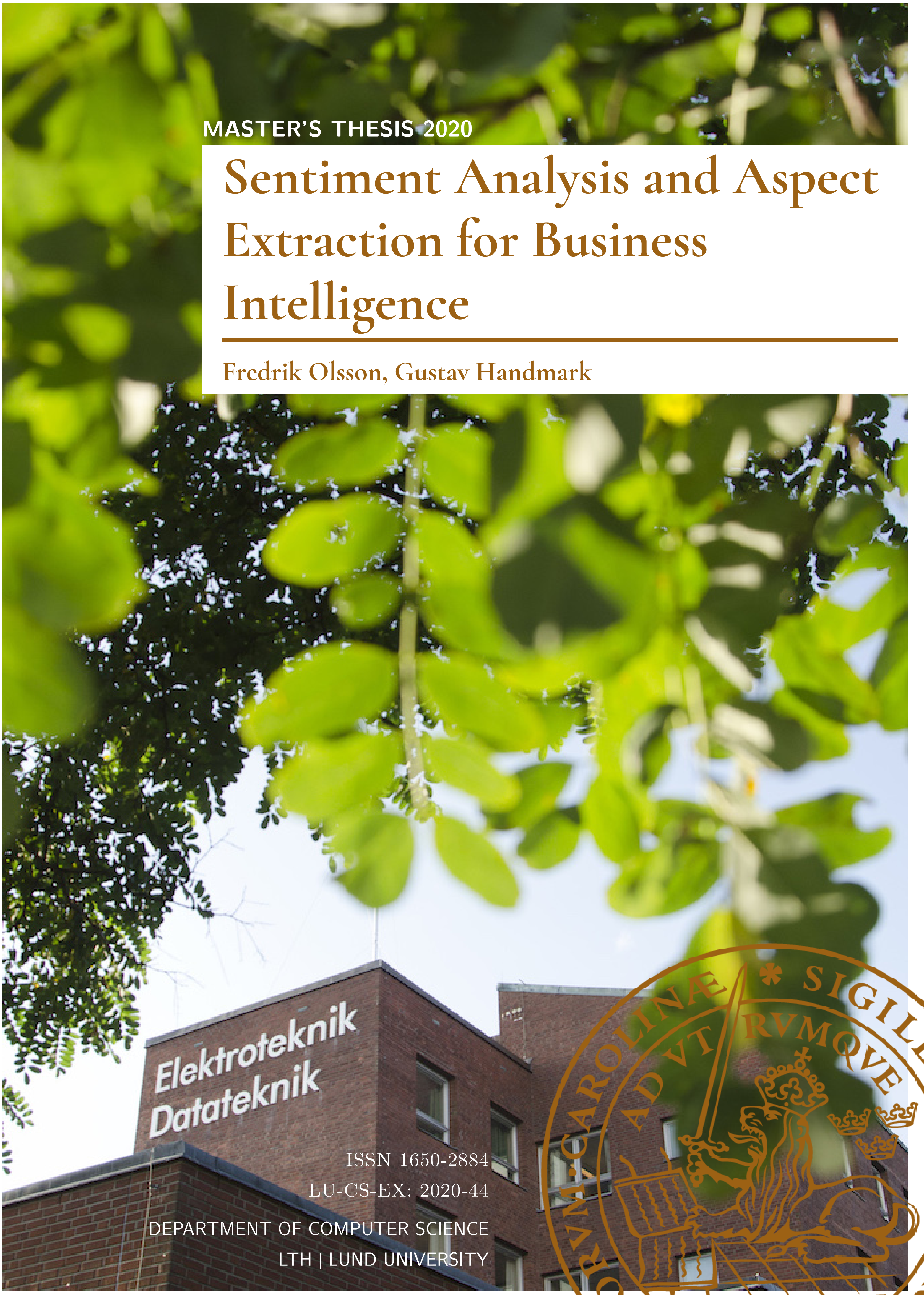
MASTER'S THESIS 2020

Sentiment Analysis and Aspect Extraction for Business Intelligence

Fredrik Olsson, Gustav Handmark



ISSN 1650-2884
LU-CS-EX: 2020-44

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2020-44

Sentiment Analysis and Aspect Extraction for Business Intelligence

Fredrik Olsson, Gustav Handmark

Sentiment Analysis and Aspect Extraction for Business Intelligence

(A practical approach based on NLP methods to automate the process of creating insights from public opinionated text data)

Fredrik Olsson
`ine15fol@student.lu.se`

Gustav Handmark
`jup14gha@student.lu.se`

June 8, 2020

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisors: Oskar Handmark, `oskar@backtick.se`
Pierre Nugues, `pierre.nugues@cs.lth.se`

Examiner: Jacek Malec, `jacek.malec@cs.lth.se`

Abstract

Data mining and predictive analysis are important instruments in business intelligence. This should lead to insights essential for identifying strategic business opportunities. The amount of publicly available data can be a daunting task to process manually, which is why automated approaches have become popular. In this thesis, we explore current state-of-the-art NLP techniques for processing company targeted customer reviews to provide meaningful and actionable insights. Our approach is two-fold. First we train, fine-tune, and evaluate multiple different models for sentiment analysis of review texts. Secondly, we conduct aspect-based opinion mining to extract fine-grained information in every review text. The results are aggregated and displayed in multiple graphs and informative tables allowing easy interpretation of the data and the captured trends. This is done for two languages, English and Swedish. We found that with current technology, we are able to train models that are effective and achieve good results on benchmarks.

Keywords: NLP, sentiment analysis, aspect-based opinion mining, text classification, business intelligence, transformers

Acknowledgements

We would like to especially thank our supervisor Pierre Nugues for his support, constructive feedback and genuine interest in our project. We greatly appreciate the weekly meetings and the time he has devoted to helping us, always with a smile and encouraging words. We would also like to thank everyone at Backtick Technologies for allowing us to pursue our project with great freedom at an inspiring workplace. It has been a pleasure working alongside you on this project.

At last, we would also like to thank the teams behind open source libraries like `transformers`, `simpletransformers`, `spacy` and `tensorflow`. These tools have been of great help and made our project possible. We would also like to thank National Library of Sweden for releasing their pre-trained Swedish BERT model.

Contents

Chapter 1

Introduction

1.1 Background and context

One of the main goals of business intelligence is to provide a basis for making informed decisions and taking strategic action. By transforming raw data into meaningful and actionable insights, such a basis can be established. Digitization provides new resources for capturing large amounts of publicly available data. Such raw data may come from many different sources both internally and externally. Internal data usually consists of support tickets, e-mails, customer surveys, marketing materials, production statistics etc. External data includes, for example, publicly available financial data, news outlets, customer reviews (the data source that will be used in this thesis) and employee reviews. However, the vast amount of raw data available is usually an insurmountable task to process manually.

Much of this raw data is in the form of plain text. To automate parts of this process, we used natural language processing (NLP) in order to understand natural language in a computer environment. As such, the primary goal of this thesis is to investigate, evaluate, and create tools to yield insights about customer pain points, needs and wants, expectations and sentiment with regards to a certain enterprise, service, or product.

This information may then be used as a basis for competitive analysis for multiple enterprises within the same field. For example by evaluating key areas, where the company's performance is below or above average in regards to its competitors. This, in turn, yields actionable insights.

1.2 Problem

The problem we will try to solve is to create a general framework that, from a collection of user reviews, gives us insights and understanding of the business of a company. This will provide us with an informational basis for company analysis. More specifically, we would like to find consumer sentiments of

- The reviews as a whole;
- The specific aspects mentioned in the texts.

To accomplish this, we will develop machine learning models and pipelines for sentiment classification of texts as well as aspect-based opinion mining (or also known as aspect-based sentiment analysis). We will investigate different approaches and models to find a solution that works well.

As part of solving this problem, we will integrate the machine learning models, pipelines and results in a user interface. This user interface will serve as a platform for company analysis.

1.3 Reviews as a phenomenon and data source

A review is a formal or informal assessment of some material with the intent of expressing the author's view or opinions on it. Commonly, when it refers to products or services, it is a way to inform or influence future potential customers to make them more or less inclined to purchase the product or service. As reviews get more popular, there are obstacles with manually parsing the amount of information contained within multiple reviews. In this project, it is an excellent data source since it is easily obtainable and usually contains some sort of score, which can be used in supervised learning.

However, there are also some pitfalls that one should be aware of, especially with regards to fake reviews. It is often hard to distinguish between a fake review and a real one, as they carry the same message with regards to the meaning of the text.[1] From a customer viewpoint, fake reviews will often influence the customer in a non-beneficial way. To combat fake reviews, there is often some way to rate or grade reviews with regards to how helpful the review was (in essence, reviewing reviews).

Individual users may themselves have scores depending on how many reviews they have written, and which grade their reviews have received. Amazon has also added the *Verified Purchase* tag to reviews, where they have confirmed that the person reviewing a product actually purchased it, and did not receive it at a considerable discount.

However, fake reviews do not necessarily have an impact on the training and inference of a purely binary sentiment classifier as they are usually inseparable from a real review. Rather, fake reviews could, in this context, be seen as an addition to the training data in the same way as synthetic data is used. With regards to aspect-based sentiment analysis and the use-cases for business intelligence, one should be wary when drawing conclusions from review data if the dataset consists of a high amount of fake reviews.

[1]Which is also the purpose of a fake review. If it could be distinguished from a real review it would not be very useful

1.4 Contribution

This thesis will provide practical examples of how NLP methods can be used to analyze large amounts of text data to gain insights. More specifically, we will try to extract consumer sentiments expressed towards a company from text data such as user reviews.

The work has been done in collaboration, and more or less all sections have been written by both authors.

1.5 Related work

Here we present some examples of previous work related to the tasks *sentiment text classification* and *aspect-based sentiment analysis* – the same tasks we have dealt with ourselves in this work.

1.5.1 Sentiment text classification

Sentiment text classification is a common subtask within sentiment analysis. It consists of classifying texts into different sentiment categories; in the simplest case a positive or a negative class. Many approaches to solve this task with different model types has been made in recent years.

Kim (2014) used a convolutional neural network model trained on top of pre-trained word embedding vectors. Tai et al. (2015) instead approached the sentiment classification task using the recurrent network type Long Short-Term Memory (LSTM), commonly used to model sequential data (such as text).

More recently, many approaches have been based on different language models using the Transformers architecture. Devlin et al. (2018) tested their BERT model on various tasks including sentiment classification. Another example is Gong et al. (2019) who proposed using a variation of the language model XLNet by Yang et al. (2019), which they called BroXLNet.

1.5.2 Aspect-based sentiment analysis

The task of aspect-based sentiment analysis (or aspect-based opinion mining), which aims to find the sentiment expressed towards specific aspects, is a challenging task. There has been many attempts in recent years to tackle it, using both supervised and unsupervised methods.

Li et al. (2019) proposed a supervised approach that made use of the sentence level context representations from BERT by placing a task-specific classification layer on top of the BERT architecture. Sun et al. (2019) also made use of BERT in their solution. They however constructed what they called an *auxilary sentence* from the aspect to convert the task into a sentence-pair classification task like *question answering* and *natural language inference*.

In contrast to Li et al. (2019) and Sun et al. (2019), Anoop and Asharaf (2018) instead took an unsupervised approach to the aspect-based sentiment analysis task. They made use of the topic modeling algorithm called *latent Dirichlet allocation*, and more specifically the extracted keywords for each topic category. Giannakopoulos et al. (2017) took another unsupervised approach, namely to create a system that automatically creates labelled datasets for aspect-based sentiment analysis. Then they used these datasets to train a supervised model based on Bi-LSTM and *conditional random fields* (CRF).

Chapter 2
Approach

2.1 Method

2.1.1 CRISP-DM

As a methodology for solving the machine learning problems stated in the problem section above, we applied the *Cross-Industry Standard Process for Data Mining (CRISP-DM)*, which is an industry-proven data mining process developed by IBM.

CRISP-DM gives an overview of the data mining life cycle, as well as describing the typical phases of a data mining project and tasks involved in each phase. The process is divided into six phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment. These phases are not necessarily sequential, and projects may iterate back and forth between the different phases as needed (IBM, 2020).

Figure 2.1 shows an illustration of the CRISP-DM data mining life cycle.

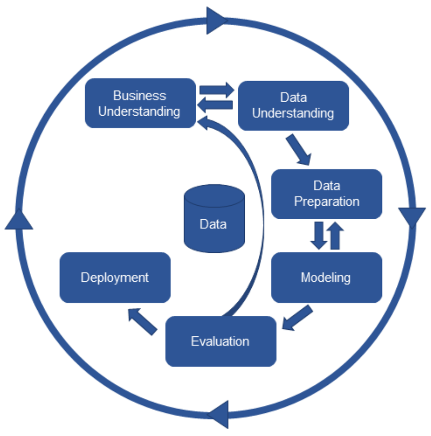


Figure 2.1: The CRISP-DM life cycle. After Akan Esen (2018)

Business understanding. This first part places focus on understanding the business perspective of the data mining process. What are the project objectives, expectations and requirements from a business perspective, and how these are transformed into a problem definition and preliminary plan for the data mining project. It is also useful to establish a business success criteria, i.e. which outcome is acceptable to the end-user (IBM, 2020).

Data understanding. This phase begins with an initial data collection and after that, it is mainly about having a closer look at the data in order to better understand it. Typical ways of dealing with this is data exploration using tables, graphs and various kinds of statistics. The data understanding phase is an important part in determining the quality of the data, and hence to avoid unexpected problems during the next phase – data preparation (IBM, 2020).

Data preparation. This phase of the data mining life cycle covers all activities involved in constructing the final dataset that will be used for modeling, starting from the initial raw data. The exact activities involved will vary depending on the company and data mining task, but examples are merging and aggregating datasets, data cleaning and splitting data into training, validation and test sets (IBM, 2020).

Modeling. During this phase different models are selected according to the data types available, the data mining goals and requirements. For example predictive models for sentiment analysis.

Usually it is necessary to test different models and in turn go back to the data preparation phase as different models may require a different structure. In this phase, it is necessary to have a firm business understanding so that the models are actually useful in achieving the goals of the data mining process. Furthermore, it is necessary to outline in which regards the models should be tested and to agree upon a few final models which can be evaluated more thoroughly in the next phase (IBM, 2020).

Evaluation. The final models agreed upon in the modeling phase should be evaluated with regards to the performance according to some performance measure (e.g. F1-score) on unseen data. We also need to verify if the results of the models are good enough to meet the business success criteria, and in turn decide which champion model we should deploy (IBM, 2020).

Deployment. The deployment phase moves the champion model to a production pipeline, where new unseen data is pre-processed in the same way as during the data preparation and modeling phase. If the models are accurate enough, the results should lead to new insights which help facilitate improvements within the organization (IBM, 2020).

2.2 Implementation

Training our models was mainly done in Google Colaboratory, which allows you to write and execute `python` code in your browser with zero configuration and free access to highend

GPUs. This sped up the training of our models significantly, and the project would not have been feasible without access to these GPUs.

For the training of our classification models, we have used the `transformers`[1] library which contains general-purpose architectures for natural language understanding with a wide range of pre-trained models which can be expanded upon. The library also incorporates both `tensorflow` and `pytorch` of which we used the latter. For aspect extraction and opinion mining, we used both a neural network model (BERT-based) and dependency parsing via the `spacy` library.

For real-time inference or predictions, we used a flask backend, where some of the models were pre-loaded. Pre-loading the models reduces the time for predictions. Unfortunately, this requires a large amount of RAM and it would benefit from being hosted in a cloud environment. The backend is coupled with a React frontend for visualization.

To explain the predictions of our classifiers, we have used *Lime*[2]. Lime will perturb the object of classification and create a local linear approximation around this object to explain which features are most important. For example, in a sentence of words, this will generate an importance score of each word so that we can determine what the model perceives as the most important words when making the prediction (Ribeiro et al., 2016).

[1]`https://github.com/huggingface/transformers`; we also used an extention to this library called `simpletransformers`; `https://github.com/ThilinaRajapakse/simpletransformers`

[2]`https://github.com/marcotcr/lime`

Chapter 3

Theory

3.1 Natural Language Processing (NLP)

Natural language processing, or NLP, deals with data in the form of natural language and is now tightly coupled with machine learning. NLP is the field where we try to give computers the ability to read, understand, and derive meaning from the human languages. Natural language data can be in terms of both written text and speech, but our work is focused on written text only.

3.1.1 Classical NLP techniques

First we look at some classical NLP techniques used when working with text data in machine learning.

Tokenization. Tokenization is the process of splitting a text into smaller parts such as words and terms, and these smaller parts are called *tokens*. It is a bit more complex than just splitting a text based on the spaces. Let us look at the example:

The service was great, but the food wasn't that good.

to illustrate it. Just splitting this sentence based on the spaces will yield:

```
["The", "service", "was", "great,", "but", "the", "food", "wasn't",
"that", "good."]
```

while tokenization results in:

```
["The", "service", "was", "great", ",", "but", "the", "food", "was",
"n't", "that", "good", "."]
```

As we can see, the negation in *wasn't* is treated separately as its own token. Whether punctuation signs become tokens or are completely disregarded may differ among various implementations.

Part-of-speech tagging. Part-of-speech tagging (POS tagging) means assigning different classes to the tokens in a text, based on the grammatical properties of the tokens. There exists a list of Universal POS tags, and some examples are ADJ (adjective), PUNCT (punctuation) and NOUN (Universal Dependencies, 2020b). Each token in the example text:

> The service was great, but the food wasn't that good.

is assigned with the following POS tags:

The - DET, service - NOUN, was - AUX, great - ADJ,
, - PUNCT, but - CCONJ, the - DET, food - NOUN
was - AUX, n't - PART, that - DET, good - ADJ, . - PUNCT

Lemmatization. Lemmatization is the process of reducing words to their base form, also called *lemma*. Some examples of this are

$$\begin{aligned} \textit{enjoyed} &\rightarrow \textit{enjoy} \\ \textit{best} &\rightarrow \textit{good} \\ \textit{deliveries} &\rightarrow \textit{delivery} \end{aligned}$$

This makes it possible to analyze every variant of a word as a single item. For example, if we look at the two sentences *I liked the food* and *I like the food*, they have the respective opinion pairs (food, liked) and (food, like). However, they carry exactly the same sentiment and when aggregating the results it would be good if they are considered to be equal. This can be achieved by replacing *liked* with its lemma *like*.

Dependency parse trees. A dependency parse tree can be used to describe the syntactic structuring of a sentence. Apart from the tokens in the sentences, it includes the POS tags for each token as well as the existing relations between tokens. These relations exist between each token and a head word, also knows as a relation between dependent and governor. There exist a list of Universal dependency relations, and two examples are *amod* (adjectival modifier) and *obj* (object) (Universal Dependencies, 2020a). In the dependency parse tree, these relations are displayed as arrows going from the governor to the dependent. Figure 3.1 shows a parse tree of the example:

> The service was great, but the food wasn't that good.

generated by the `spacy` parser model.

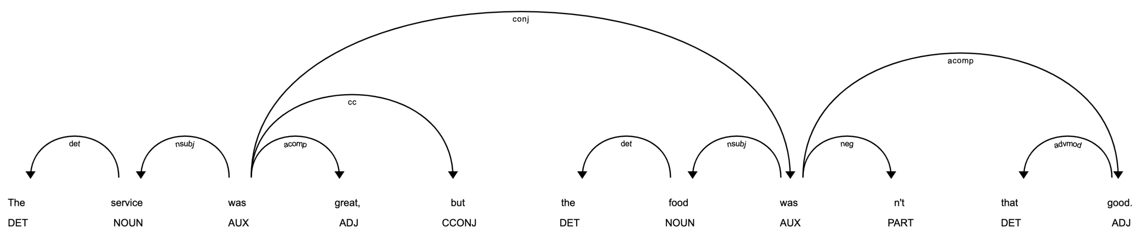


Figure 3.1: Dependency parse tree

Sentiment lexicons. A sentiment lexicon is a collection of words and phrases typically used to express positive or negative sentiments. Words like *good*, *enjoy*, *disappointed* and *horrible* are most likely present in a sentiment lexicon. It is also common that these sentiment lexicons have some kind of floating sentiment polarity score such that *amazing* has a higher positive score than *good*. Furthermore, the lexicon is often able to deal with adverbial modifiers so that *very bad* gets a higher negative score than *bad*.

3.1.2 Word vector embeddings

It is not possible to use the raw text data just as it is for machine learning, since machine learning models require numerical input values. Some kind of numerical representation of the text data is therefore needed for the NLP models. Classic approaches to solve this problem are one-hot encoding, bag-of-words or TF-IDF. An alternative, more current approach, is to use so called word embeddings. They usually yield a better performance.

By using word embeddings, we represent the words in the text as vectors in a large vector space. Word embedding models have been trained on very large corpora to create the vector space in a way that words with similar contexts will be close to each other in the vector space. The idea is that words like *burger* and *pizza*, or *green* and *yellow* will have similar word vectors. It is also possible to perform arithmetic operations on the word vectors, such as

$$\textit{Queen} + \textit{Man} - \textit{Woman} = \textit{King}$$

or computing similarity scores between word vectors.

There exist several word embedding models, on many different languages. Some common ones are Google's word2vec[1], Facebook's fastText[2] and Stanford's GloVe[3].

3.2 Machine Learning

Machine learning has been, and will continue to be, a widely discussed topic both in academia and in the industry. Machine learning is responsible for numerous technical advances during recent years, especially in the fields of image and speech recognition, autonomous cars, and natural language processing. In general, machine learning refers to a field of studies, where algorithms are applied on a set of data to identify patterns and get better at identifying these

[1]https://code.google.com/archive/p/word2vec/
[2]https://fasttext.cc/
[3]https://nlp.stanford.edu/projects/glove/

patterns through learning. The following definition, coined by Mitchell (1997) is often used to describe learning:

> A computer program is said to learn from experience $\boldsymbol{E}$ with respect to some class of tasks $\boldsymbol{T}$ and performance measure $\boldsymbol{P}$, if its performance at tasks in $\boldsymbol{T}$, as measured by $\boldsymbol{P}$, improves with experience $\boldsymbol{E}$.

3.2.1 Classification

One of the most typical machine learning tasks is classification. In classification, we train a model to perform the task of assigning new observations to a specific pre-defined class, based on an annotated training set containing data points with known class belongings. The number of classes is fixed but can be arbitrarily many. In the simplest case, we have just two different classes, and this case is usually referred to as binary classification.

When training a binary classification model, one typically uses the binary cross-entropy function (also called log loss function) as loss function, i.e. the function which the model tries to minimize.

Given training data points $(\mathbf{x}_1, ..., \mathbf{x}_n)$ and corresponding true target classes $(y_1, ..., y_n)$, where $y_i \in \mathcal{Y} = \{0, 1\}$, we train the binary classification model to minimize the following loss function:

$$L = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1 - y_i)(1 - \log(\hat{y}_i))\right],$$

where $\hat{y}_i$ is the model output for $\mathbf{x}_i$.

Evaluation metrics

In order to evaluate the performance of machine learning models, we need some kind of numerical evaluation metric. For classification, a common one is accuracy, which is defined as:

$$\text{Accuracy} = \frac{\text{Correct classifications}}{\text{Total classifications}}$$

i.e. the fraction of correctly classified data points.

While accuracy is intuitive and easy to understand, it has some drawbacks as an evaluation metric for classification. It does not say anything about the performances on the different classes, which is particularly a problem when dealing with skewed datasets. A more sophisticated measure that is often used instead, is the so called F1-score. The F1-score is based on precision and recall, and are for the both classes in a binary classification problem defined as follows:

	Predicted 0	Predicted 1
Actual 0	True negatives (TN)	False positives (FP)
Actual 1	False negatives (FN)	True positives (TP)

$$\text{Precision class 0} = \frac{TN}{TN+FN}, \quad \text{Precision class 1} = \frac{TP}{TP+FP}$$
$$\text{Recall class 0} = \frac{TN}{TN+FP}, \quad \text{Recall class 1} = \frac{TP}{TP+FN}$$

and the F1-score for each class is then defined as:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

It is then common to use the macro averaged F1-score as the final evaluation metric, which is the arithmetic mean of the F1-score for each class:

$$\text{Macro F1-score} = \frac{\text{F1-score class 0} + \text{F1-score class 1}}{2}$$

Logistic regression

The logistic regression model is perhaps the simplest one for binary classification and is often used as a first baseline model. The model output $\hat{y}_i$ of the data point $\mathbf{x}_i = (x_1, ..., x_m)$, from the logistic regression model is

$$\hat{y}_i = \varphi\left(w_0 + \sum_{j=1}^{m} w_j x_j\right) = \frac{1}{1 + \exp\left(-(w_0 + \sum_{j=1}^{m} w_j x_j)\right)}$$

where $\mathbf{w} = (w_0, w_1, ..., w_m)$ are the model weights. The function φ is the sigmoid function (also known as the logistic function), and it transforms the weighted sum of input values $\sum_{j=1}^{m} w_j x_j$ so that the model output is on the interval $[0, 1]$.

From a neural network point of view, the logistic regression model can be created by an input layer followed by a single output node with the sigmoid function as activation function.

3.2.2 Neural networks & recurrent networks

Neural networks are a subset in the field of machine learning. Neural networks are formed by a multitude of nodes, each node represents an input-output function which forwards the output to the next node in the network. These nodes are often layered with multiple nodes in each layer, which in turn creates a network as the layers are linked. As the output of one layer is forwarded to the next layer, information can propagate through the network.

There are many different types of neural network configurations, i.e. how the nodes are connected to form a network. One of them is a feed-forward network, which has connections in only one direction. Every node receives input from a previous node and delivers its output to a node further along. The output of one node is dependent on the node activation function, the input from the previous node, and its weight (which determines the strength and sign of the output). Examples of activation functions are hard threshold (1 or 0), a logistic function, or rectified linear unit (ReLU). While learning, we will update the weights of the neural network based on the output and corresponding loss. During training the goal is to minimize the loss (Russell and Norvig, 2016).

Another example is recurrent neural networks (RNNs), which will have a feedback loop, where the output is fed back as an input, so each new input may be accompanied by a hidden state calculated from a previous input. As such, the output of the network for a given input can be dependent on its previous state, and thus supports short term memory. An example of a task where this is useful is on sequences of information, such as video clips. When the network is processing a frame in the video clip, it will have information about the previous frames that it has already processed. The same applies when considering words in a sentence, where the context depends on the previous word (Russell and Norvig, 2016).

LSTM

An improvement on the simple node in an RNN is the Long short-term memory (LSTM) unit. The LSTM network allows the network to store information, outside of the normal flow of the RNN, in a LSTM unit (also called gated cell). This allows the unit to forget previous information when it is no longer relevant, for example when processing a new sequence of information. This unit replaces the previously discussed node. Simplified, this unit contains a few main components. An input/input gate, which takes the current input and the input from a previous time step. A forget gate, which ensures that the LSTM is capable of resetting the memory contents when they are no longer relevant or belong to a time which is too far into the past. A memory cell with a state that is calculated (dependent on the forget gate) whether to forget information based on previous input and update state based on new information from the current input. The last component is the output/output gate, which controls what information flows from the LSTM unit (Singh, 2017).

There are several different versions and configurations of LSTM networks, one of them is bi-directional LSTM networks, which allows sequential information to be processed both with regards to previous time steps, but also with regards to future time steps. For example, when processing a sentence of words, where the context of a specific word is dependent both on the words before and after it.

3.3 Transformers

3.3.1 Language models

Understanding natural language is by no means an easy task. Languages includes complex grammatical rules, can be ambiguous (one sentence can have different meanings depending on context), new words are added and the language changes as time passes. Previously we have discussed POS tagging and dependency parse trees. In this chapter, we will study a more recent development. Large semi-supervised neural networks can be utilized to create models that perform very well at general language understanding. These language models can be described as models that predict the probability distribution of language and language expressions. These general models can in turn be modified to achieve very good results in classification, sentiment analysis, question answering and next-sentence prediction (Russell and Norvig, 2016).

The models below owe their success to two modern technological advancements: *attention* and *transformers*.

3.3.2 The transformer architecture

In LSTM networks and RNNs, the input is sequential and the previous output is fed back to the network and used again during the next time step. Two problems that arise from this configuration is that it can be hard to parallelize computations and RNNs can have a hard time learning long-range dependencies. Since the input is sequential, the GPU basically has to wait for the result of a previous iteration to complete. As such, the mentioned networks requires considerable computation power and time during training for any substantial amount of data.

To alleviate this problem the transformer model was proposed in a paper published in 2017. The model relies entirely on attention mechanisms instead of recurrence. The attention mechanism is not unique to the transformer architecture and has been used in previous models. However, the transformer architecture was the first to rely solely on attention and fully connected feed forward layers, which allows for parallelization in contrast to sequence-aligned RNNs (Vaswani et al., 2017).

Encoder & decoder

Before describing attention and the transformer model in more detail, we need to introduce two new concepts: *encoder* and *decoder*.

Imagine we have a sequence-to-sequence operation, where we want to translate a sentence in English,

I like this restaurant but not their food

to

Jag tycker om den här restaurangen men inte deras mat

in Swedish. The input and output have different lengths (8 for the English sentence, and 10 for the Swedish sentence), so we need a model capable of mapping sequences of different length to an output which may have a different length than the input.

The idea behind the encoder-decoder architecture is that the encoder will take the input sequence (using word embedding to map strings to numeric values), process it into an intermediate vector representation which will be the initial input state to the decoder. The decoder will in turn produce the output sequence we are looking for. The model is trained to maximize the likelihood of the correct output sequence. The intermediate encoder vector representation will ideally capture every useful piece of information regarding the input sequence (Kostadinov, 2019).

In almost all cases, the encoder-decoder architecture is used in conjunction with RNNs or LSTM networks, as they support sequences and memory. When the encoder processes our input sentence, one word at a time, it will result in multiple hidden states (the feedback states) which depend on the previous hidden state for each word in the sequence. The final hidden state (which combines the previous hidden states), called the intermediate encoder vector representation, is then the input to the decoder (Sutskever et al., 2014).

With our example sentence, we have to process the sequence one word at a time, while keeping the dependencies in memory. For example, the models need to know that the second last word *their* refers to the word *restaurant* at position four when predicting the word *deras* in the output sequence at step nine. As the sequence length becomes longer and longer, we may need to remember dependencies at time steps much further in the past, which is harder and harder to do.

Attention

To solve this problem, *attention* was introduced as a way to model dependencies regardless of their position in the input and output sequences, and to avoid the problem of having to capture every dependency within one single vector (the intermediate encoder vector representation in the previous section) (Britz, 2016).

Attention introduces attention weights which are calculated based on the current decoder state. Essentially this allows the decoder to look back at the input sequence and choose which hidden states to use based on what is most important at this current step. The decoder has access to all hidden states during different steps of the encoder, not just the last hidden state as was the case in the intermediary encoder vector representation, and chooses the most relevant ones based on the attention weights (Britz, 2016).

So instead of having the encoder try to capture every dependency in one single vector representation, we let the model decide which hidden states are actually useful at this step in the sequence. The attentions weights are trainable just like the weights in a regular neural network (Britz, 2016).

Transformers

At this point, attention has allowed us to better capture long-range dependencies. However, we still have a problem with the sequential nature of RNNs. We cannot parallelize the computations during training, as the hidden states will depend on the previous inputs in the sequence.

The *transformer* model was primarily proposed to allow parallelization and reduce training costs (and we will see in the following sections how this has allowed large models to train on extremely large sizes of training data). Instead of RNN or LSTM networks, the transformer is comprised of encoders and decoders relying entirely on attention and fully connected feed-forward network layers, which makes it easy to parallelize. On a high level, the transformer provides the possibility to see the entire input sequence all at once, instead of sequentially. All the while capturing dependencies through the attention mechanism. The model architecture for the encoder and decoder layers used in the transformer model can be seen in Figure 3.2 (Vaswani et al., 2017).

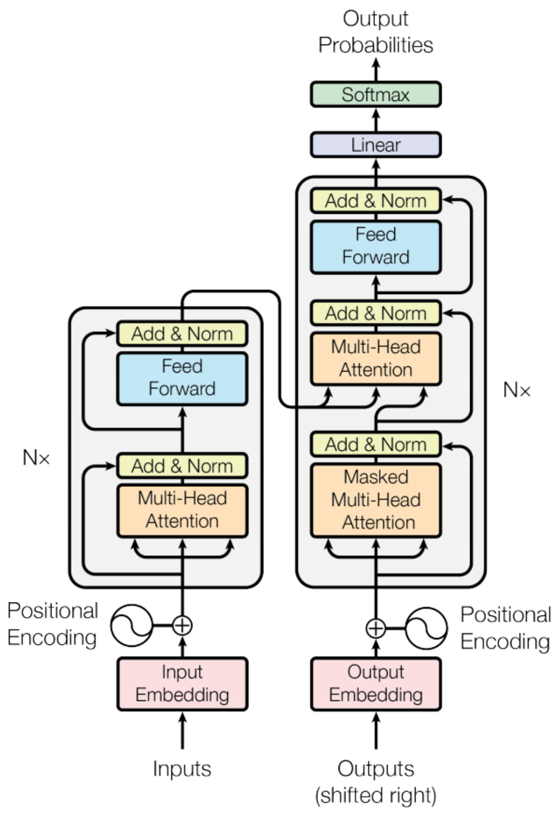


Figure 3.2: Transformer model architecture. After Vaswani et al. (2017)

In the leftward part of the picture, we see the encoder layer, which in turn is comprised of input embeddings, positional encodings, something called *Multi-Head Attention* which incorporates self-attention, layer normalization and a feed forward network.

The decoder has basically the same components, but with an added sub-layer, *Masked Multi-Head Attention* that performs multi-head attention on the previous decoder states. The complete model is comprised of six encoders and six decoders. So in reference to Figure 3.2, you stack six instances of the left side of the figure on top of each other to get the encoder stack, then feed the output from the last encoder to the first decoder in the decoder stack. The number of stacked layers can be modified (Vaswani et al., 2017).

The figure can be hard to understand at first glance, but keep in mind that the main goal is to allow parallelization of computations and to remove the need for sequential processing. In the following, we will describe the most important component, which is the Multi-head attention module (Vaswani et al., 2017).

In the Multi-head attention module, self-attention is a way to understand the relevance of other words in the sequence with regards to the current word that is being processed. It can be described with an example. Say we want our model to translate between two languages, and the input is

The animal didn’t cross the street because it was too tired.

In this case, we have to consider that, for our model, the word *it* may refer to both the animal and the street (for us this is trivial, we know a street cannot be tired). When the encoder processes the word *it*, self-attention allows the encoder to look at different words in the sentence which can lead to a better encoding for this word (Alammar, 2018).

Simplified, we score each word of the input sequence against the word we are currently processing to determine the best encoding for that word. The scores determine how much attention to give other parts of the input when encoding a specific word, this is done for every word in the sequence. *Multi-head* simply means that we do this multiple times. Each head has its own attention weights which are initialized differently and tuned during training. The result of the individual heads are then weighted together. This will allow the model to better capture different dependencies (Alammar, 2018).

In Figure 3.3, we can see the result of two heads when encoding the word *it*. The orange colors represent the attention of first head, and the green colors the attention of the second head. We can see that one head is mainly focusing on what *it* refers to, in this case *the animal*,

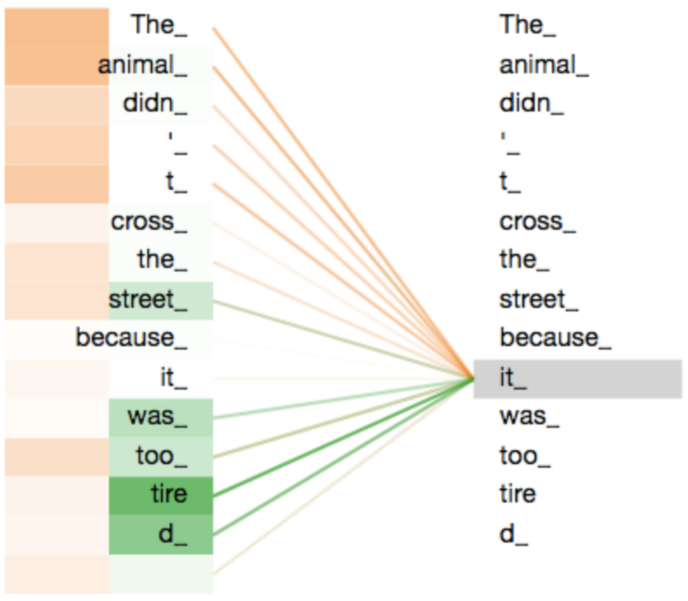


Figure 3.3: Focus of two different heads when encoding the word *it*.
After Alammar (2018)

and the other head is focusing on that it was tired. Multiple heads simply improve the models capabilities of encoding words (Alammar, 2018).

If we go back to Figure 3.2, we can see that multi-head attention occurs at three different places. In the encoder module (the left side) and in two places in the decoder module, at the *masked multi-head attention* and where the output of the encoder feeds into the decoder.

We have described above how the multi-head attention module works in the encoder. In the decoder, the multi-head attention module which processes the encoder output helps the decoder to focus on the relevant parts of the input sequence (similarly to how the attention mechanism works in a typical encoder-decoder model). During training, where the real output is known and fed to the decoder, the masked multi-head attention module allows the decoder to look at previous output words up until the current word it is predicting, with the same self-attention mechanism described for the encoder (Vaswani et al., 2017).

For instance, when translating one sentence to another language, the decoder has the capability of looking back at the already predicted words in the output sentence as well as the encoded input sentence when predicting the next output word. But since the real output is known and fed to the decoder during training, the words after the one it is currently predicting needs to be masked, as it is not supposed to have information available that occurs later in the sequence (Alammar, 2018).

Note that this does not conflict with parallelizing the training, as the computations can be made on different segments regardless, you simply mask different words. The output of the decoder will be a list of probabilities of words which could be the next word in the output sequence and the word with the highest probability will be chosen. The process is finished when the decoder reaches a specific <end of sequence> tag (Alammar, 2018).

Finally, there is one more important thing in the model architecture. Since the model does not process the input sequentially and as there is no recurrence, in contrast to RNNs and LSTMs, the model needs a positional vector to describe the order of the input. This is done at the bottom of the encoder and decoder stacks. (Vaswani et al., 2017)

3.3.3 Implementations

The transformer model has resulted in numerous new approaches for solving NLP tasks. In this section, we will explore three such variants which are used directly in this project: the BERT model (2018), a derivative of BERT known as RoBERTa (2019), and XLNet (2019).

BERT

Bidirectional Encoder Representations from Transformers (BERT) was developed by Google Research which, at its release, obtained state-of-the-art results on multiple NLP tasks, such as GLUE[4] (General Language Understanding Evaluation) and SQuAD question answering test. The model architecture of BERT is a multi-layer bidirectional transformer, which is based on the original transformer model discussed previously. In fact, the use of the transformer in BERT is almost identical to the original. The BERT base model uses 12 layers of transformer blocks, a hidden layer size of 768, 16 self-attention heads (Devlin et al., 2018).

The input to the BERT model can be a single sentence or a pair of sentences. The reason for allowing sentence pairs is that is simplifies and improves the performance some specific tasks such as question answering and sentence similarity. In Figure 3.4, we can see that the input token sequence to BERT consists of three different embeddings. The positional embeddings depicted in the bottom of the figure are used to describe the order of the input (same as in the original transformer model). The Segment embeddings are used to track whether the word belongs to sentence one or sentence two (when using sentence pairs). The token embedding is a representation of each word using the WordPiece embedding, which has a 30,000 token vocabulary and will split words such as “playing” to “play” and “##ing”. The first token in the input is always a reserved token [CLS] and [SEP] is used to separate sentence pairs (Devlin et al., 2018).

[4]At the point of writing, BERT is currently at placement 22 on the leaderboard for GLUE, while RoBERTa holds place 10 and the human baseline holds place 12. XLNet does not have a placement rank as it is not evaluated on one of the 12 tasks, but on the other tasks its performance is just shy of RoBERTa.

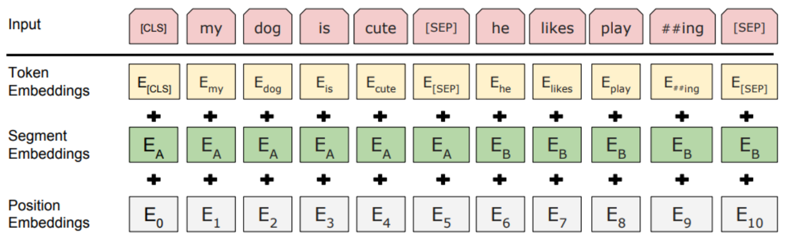

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8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

Figure 3.4: BERT input representation. After Devlin et al. (2018)

BERT is trained in two different ways, *pre-training* and *fine-tuning*. Pre-training refers to the process where the model is trained on a large corpus of unlabeled data to get a general language understanding. In the fine-tuning phase, the model is initialized with the pre-trained parameters (weights) and fine-tuned on task-specific labeled data. Essentially, the model uses transfer learning by training on a large corpus of unlabeled data to improve its performance on a specific task (Devlin et al., 2018).

The pre-training phase consists of two mechanisms: masked language modeling (MLM) and next sentence prediction (NSP). BERT is a deep bidirectional model in contrast to sequentially processing a sentence from left-to-right. Training a model to predict the next word in a sentence is not possible if the entire sentence is available to the model [5]. Instead, some words are randomly masked with a specific token, [MASK], and the training goal is to predict the original token using the contextual information provided by the transformer. Masking is done in a procedure where 15% of the token positions of the input sequence is selected as candidates. Each candidate is then either masked (80% of the cases), unchanged (10% of the cases) or exchanged with a random token (10% of the cases) (Devlin et al., 2018).

The second mechanism is next sentence prediction. Each sequence input pair of the training data consists of two sentences. In half of the pairs, the second sentence is the actual sentence that follows the first sentence. In the other half, the two sentences are unrelated. The training goal is to predict correctly whether the second sentence follows the first sentence. According to the paper, this has helped the performance of the model at question answering and sentence similarity tasks. These two pre-training mechanisms are then trained together with the goal of minimizing the combined loss (Devlin et al., 2018).

Fine-tuning in BERT refers to the adaptation of the model to specific tasks. Such tasks may be next word prediction, next sentence predictions, question answering and classification, amongst others. The pre-trained parameters are loaded and for some tasks, such as sentence paraphrasing and question answering, the inputs are already accommodated by BERT in the form of sentence pairs. In binary classification, we want to classify one text as, for example, either having a positive or negative sentiment. The output from the BERT model is fed through an added single linear layer along with a sigmoid output node to get the probabilities of the text belonging to either class. Since the parameters in the BERT model are pre-trained, we already have good word representations. These representations are generalized so the output can be fitted to many downstream tasks (Devlin et al., 2018).

[5]Refer to section 3.3.2. However, in the transformer, during training, the decoder only has access to the words in the sequence up until the word it is currently predicting. In a bi-directional transformer the model has access to the entire sequence.

RoBERTa

The Robustly optimized BERT approach (RoBERTa), is a derivative of the BERT model which builds on the original model with an extensive amount of hyperparameter tuning and optimization. Specifically, the modifications include training the model longer, over more data, removing next sentence prediction, training on longer sequences and changing the masking pattern. Otherwise, this model is the same as BERT, but achieves above human baseline performance. In addition to the 16GB of data BERT was trained on, RoBERTa has been pre-trained on 76GB of additional data from CC-news, 38 GB of additional data from OpenWebText, and 31GB of additional data from Stories (Liu et al., 2019).

XLNet

XLNet improves on BERT by using a pre-trained generalized autoregressive model. The authors of XLNet notes two issues with BERT. Firstly, if multiple tokens are masked during pre-training, BERT assumes that they are independent. This has consequences when trying to predict multiple masked tokens which are actually dependent. Secondly, the [MASK] used by BERT during pre-training is absent from real data during fine-tuning which results in a pretrain-finetune discrepancy. BERT originally tried to deal with this issue by only masking candidate tokens in 80% of the cases. To alleviate these problems in XLNet, the model utilizes something called *permutation language modeling*. The objective is to train the model on all permutations of words in a sentence (Yang et al., 2019).

So if the model at a specific timestep is trying to predict a word with positional index n in a sentence, it is only allowed to see the input up until index $n - 1$ (very similar to the original transformer). To capture bi-directional dependencies, the input sentences are permuted so that the word occurs at every position. When every permutation has been run, the model will have captured every possible dependency for this word in its sentence context. Since we no longer have any masks, we alleviate the problems identified in BERT. Note that the prediction target does not need to be a single word, in BERT multiple masks may be applied to a sentence (Yang et al., 2019).

The difference between BERT and XLNet may be easier to understand by observing the following example. Assume the input sequence X is "New York is a city" with the sequence order $X_1 = \text{New}, X_2 = \text{York}, X_3 = \text{is}, X_4 = \text{a}, X_5 = \text{city}$. If the prediction targets are the two tokens "New" and "York" BERT would assume that they are independent with the following objective

$$\log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \text{is a city})$$

Respectively, if the current permutation order is $(X_3, X_4, X_5, X_1, X_2) = (\text{is, a, city, New, York})$, XLNet would have the following objective

$$\log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \text{New, is a city})$$

What this means is that XLNet would be able to capture the dependency between "New" and "York" in a better way than BERT. Overall, this has indeed resulted in better results compared to the original BERT base (Yang et al., 2019).

Chapter 4
Models

4.1 English text classification models

4.1.1 LR

LR is a logistic regression model, and functions as our first baseline model. The model has three pre-processing steps:

- text pre-processing
- tokenization, features and padding
- construct embedding weight matrix

and the model itself then consists of:

- an embedding layer
- a single sigmoid output node

During the text pre-processing step we remove stopwords, punctuation, and non-alphabetic characters from the text. We then fit a `tensorflow` tokenizer model to the pre-processed data, and convert the texts to feature sequences using the fitted tokenizer model. These feature sequences are then padded to the same size.

The next step is to construct the embedding weight matrix, which will serve as a non-trainable weight matrix for the model's embedding layer. Instead of training the weights in the embedding layer, we construct this embedding weight matrix by mapping words in the tokenizer model (i.e. words from the pre-processed texts) to pre-trained 100-dimension English GloVe word embedding vectors.

The final model then consists of the embedding layer followed by a single sigmoid output node. The final model was then constructed using `tensorflow`. Note that the embedding layer has been flattened out to a vector before being fed into the output node.

4.1.2 LSTM

LSTM is a recurrent network with LSTM nodes, and is our second baseline model, but more complex and most likely better performing than the **LR** model. This model also has the same pre-processing steps

- text pre-processing
- tokenization, features and padding
- construct embedding weight matrix

and the model itself consists of

- an embedding layer
- two hidden layers, each with 50 LSTM units
- a single sigmoid output node

The pre-processing steps are the same as for the **LR** model, including the use of 100-dimension English GloVe word embedding vectors to construct the non-trainable embedding layer. Here for the **LSTM** model, we have however added two hidden layers, each with 50 LSTM units, between the embedding layer and the single sigmoid output node.

4.1.3 Transformer models

We have used three different transformer-based models. Each model is fine-tuned on every English dataset:

BERT is an English BERT-base model[1], with 12 hidden layers and 768 nodes in each layer along with a binary classification layer on top. The model we used is pre-trained on BookCorpus and the English Wikipedia (approximately 13GB of text).

RoBERTa is an English RoBERTa-base model, with 12 hidden layers and 768 nodes in each layer along with a binary classification layer on top. The model we used is pre-trained on the same data as BERT along with additional data from CC-news, OpenWebText and Stories (approximately 160GB of text data).

XLNet is an English XLNet-base model, with 12 hidden layers and 768 nodes in each layer along with a binary classification layer on top. The model we used is pre-trained on English Wikipedia, BooksCorpus, Giga5, ClueWeb, and Common Craw (approximately 160GB of text data).

For every transformer model we used Google Colab to fine-tune the model on the binary classification task. The code for the training pipeline was simplified with the use of the library `simpletransformers`[2]. Besides increasing the batch size to 32 and tuning the number of

[1]For specifics on model architecture and inner workings of BERT, RoBERTa and XLnet, please see section 3.3.3

[2]`https://github.com/ThilinaRajapakse/simpletransformers`

epochs with the validation dataset, all the models and training parameters used for all these transformer models are default values in `simpletransformers`. These default parameters are very similar to the parameters used in the original models.

4.2 Swedish text classification models

4.2.1 LR-Swe

LR-Swe is a logistic regression model, and our first baseline model for text classification on Swedish data. The model works just as the corresponding English model **LR** with the pre-processing steps:

- text pre-processing
- tokenization, features and padding
- construct embedding weight matrix

and the model structure:

- an embedding layer
- a single sigmoid output node

The only difference to the **LR** model is that we instead use 100-dimension Swedish word2vec word embedding vectors to construct the embedding weight matrix for the embedding layer.

4.2.2 LSTM-Swe

LSTM-Swe is a recurrent network using LSTM nodes, and is our second baseline model for Swedish text data. Just as its English equivalent **LSTM**, it is more complex and most likely better performing than the first baseline model **LR-Swe**. It also works just as the **LSTM** model, with the following pre-processing steps:

- text pre-processing
- tokenization, features and padding
- construct embedding weight matrix

and model structure:

- an embedding layer
- two hidden layers, each with 50 LSTM units
- a single sigmoid output node

As for the **LR-Swe** and **LR** models, the only difference with **LSTM-Swe** compared to the **LSTM** model, is that we instead use 100-dimension Swedish word2vec word embedding vectors to create the embedding weight matrix for the embedding layer.

4.2.3 BERT-Swe

BERT-Swe is a Swedish BERT-base model, i.e. 12 hidden layers with 768 nodes in each layer, with a binary classification layer on top. It was pre-trained by the National Library of Sweden on data sourced from Swedish books, news, government publications, Swedish Wikipedia and internet forums, approximately 20GB of data.[3] We again used Google Colab to fine-tune this model for the classification task, with the same pipeline used for the **BERT**, **XLNet** and **RoBERTa** models.

4.2.4 BERT-ML

BERT-ML is a Multilingual (a total of 102 languages, including Swedish) BERT-base model, i.e. 12 hidden layers with 768 nodes in each layer, with a binary classification layer on top. This model is trained on the whole Wikipedia corpus for every language. As some languages have significantly more data than others, the data is under or oversampled respectively. The same pipeline is used as for every previous transformer model.

4.3 English aspect-based opinion mining models

4.3.1 ABOM

ABOM is our first English aspect-based opinion mining model. To have a model applicable to text reviews (or similar) for any kind of company without the need of annotated data for training – which for the task of aspect-based opinion mining is very limited – we have chosen an unsupervised approach. We also want a model that can give us the opinion word used to describe an aspect rather than just a sentiment polarity, i.e. for example (Service, terrible, negative) rather than just (Service, negative). To accommodate these features, the model is based upon dependency parse trees and rules based on the relations from these dependency parse trees, to extract opinion pairs.

The dependency parser that we used for this model is one of the English models available in the `python` NLP library `spacy`[4].

The **ABOM** model can be summarized in three steps:

1. Pre-processing step to merge nouns that belong together;
2. Given the relations in a dependency parse tree and set of grammatical rules, extract opinion pairs for the text;
3. Classify the sentiment polarity for the extracted opinion pairs.

We illustrate using the example *The customer service was great*. The `spacy` model yields the following dependency parse tree

[3]Instructions to find the pre-trained model can be found here `https://github.com/Kungbib/swedish-bert-models`

[4]`https://spacy.io/`

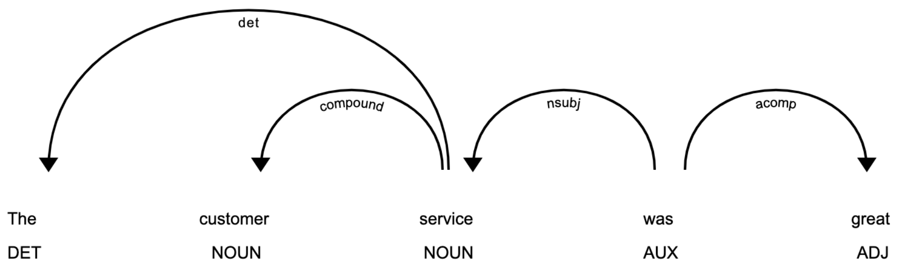


Figure 4.1: Dependency parse tree

with relations

$$
\begin{aligned}
&(\text{det, service, NOUN, The, DET})\\
&(\text{compound, service, NOUN, customer, NOUN})\\
&(\text{nsubj, was, AUX, service, NOUN})\\
&(\text{ROOT, was, AUX, was, AUX})\\
&(\text{acomp, was, AUX, great, ADJ})
\end{aligned}
$$

where the relations are on the form

$$(\text{Relation type, Governor, Gov. POS-tag, Dependent, Dep. POS-tag})$$

In the first pre-processing step, we look for compound-relations, where both words are nouns and then merge these together with a hyphen. In this example, this means that we merge *customer* and *service* into *customer-service*. Then we apply the dependency parser model again, which now treats this as one word. Instead, we get

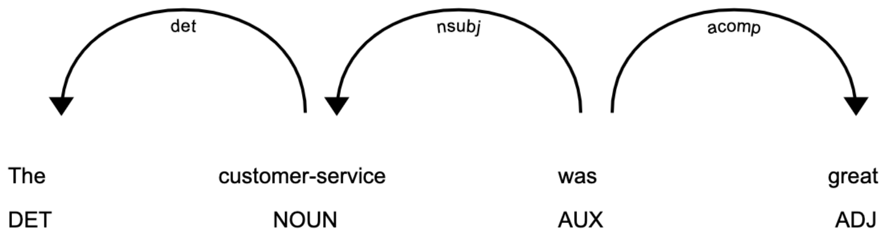


Figure 4.2: Dependency parse tree

with relations

$$
\begin{aligned}
&(\text{det, customer-service, NOUN, The, DET})\\
&(\text{nsubj, was, AUX, customer-service, NOUN})\\
&(\text{ROOT, was, AUX, was, AUX})\\
&(\text{acomp, was, AUX, great, ADJ})
\end{aligned}
$$

The next step is the opinion pair extractions, which is the main part of the model. Our algorithm for this consists of different rules based on the relations given by the `spacy` dependency parser. One example is the rule that extracts (customer-service, great). Essentially this rule says that if we have one nsubj-relation and one acomp-relation with the same governor, then the dependents from the two relations form an opinion pair. We also restrict the dependent in the nsubj-relation to nouns and proper nouns, and the dependent in the acomp-relation to adjectives.

Worth noting is that the model also takes negations into account. From sentences like *The food was not good* or *The food wasn't good*, our model finds the opinion pair (food, not good).

The final step, after all opinion pairs have been extracted, is to determine the sentiment polarity of the opinion pairs. For the example sentence it means that we extend the (only) opinion pair (customer-service, great) to (customer-service, great, positive).

In order to accomplish this, the **ABOM** model uses a combination of the Vader Sentiment Analysis lexicon from the `nltk`[5] library and our own text classifier model **RoBERTa**[6] trained on the **Yelp1** dataset. All opinion words (like *great*) are first classified with Vader. If Vader is able to classify the words as either positive or negative, the model uses this sentiment classification. This is typically the case for words that are positive or negative regardless of context, like *good*, *amazing*, *bad* and *terrible*.

For words that Vader classifies as neutral, we instead make use of the **RoBERTa** text classification model. More specifically, if we for example have the opinion pair (food, cold) and Vader classifies *cold* as neutral, we let **RoBERTa** classify *cold food*, and this prediction is hopefully the negative class in this case.

4.3.2 ABOM-SGC

Our second English aspect-based opinion mining model, denoted **ABOM-SGC**, works in exactly the same way as the **ABOM** model, but has a another pre-processing step in the beginning, namely spell and grammar correction. Dependency parser models rely on that the text it works on is correct, which is often not the case for user reviews and similar text data sources. Therefore, the first step of the model is to try and correct spelling mistakes and grammatical errors.

The **ABOM-SGC** model can thus be summarized in four steps:

1. Pre-processing step to automatically correct spelling mistakes and grammatical errors;
2. Pre-processing step to merge nouns that belong together;
3. Given the relations in a dependency parse tree and set of grammatical rules, extract opinion pairs for the text;
4. Classify the sentiment polarity for the extracted opinion pairs.

To do this first pre-processing step, we use `LanguageTool`[7], an Open Source program for spell and grammar checking. It tries to detect spelling mistakes and grammatical error and gives suggestions for correction. By replacing the marked errors with the suggested replacements, the model can automatically correct the texts.

The other parts of the model are identical to the **ABOM** model.

[5]`https://www.nltk.org/`

[6]As can be seen in the Results section, this is our best performing text classification model

[7]`http://wiki.languagetool.org/`

4.3.3 ABSA-E2E

To complement our unsupervised approaches, we also tested an already existing supervised model. This model was a modified version of an existing end-to-end BERT-based ABSA model, referred to in this paper as **ABSA-E2E**. End-to-end in this context refers to the fact that the model will take any input string of words and outputs both the aspect term and its polarity. It utilizes BERT[8] sentence level context representation and places an addition layer on top of BERT embedding layer (Li et al., 2019).

This model is trained on the 2014, 2015 and 2016 SemEval (ABSA task) restaurant train-datasets. In contrast to our unsupervised model described in Section 4.3.1, it will not give us the opinion words used to describe the aspect (like *great*). However, we do find relevant aspects and their polarity (negative, neutral or positive).

4.4 Swedish aspect-based opinion mining models

4.4.1 ABOM-Swe

Our first Swedish aspect-based opinion mining model is denoted by **ABOM-Swe**, and is practically equivalent to the **ABOM** model, but working on Swedish text data instead. With the same motivation as for the English model, this is an unsupervised model based on dependency parse trees and rules to extract opinion pairs.

This model also uses the `spacy` library for dependency parsing. However, there is no Swedish model implemented in the library, so we trained our own Swedish `spacy` model. In order to do this, we used annotated CoNLL-U text data from Swedish Talbanken[9], and followed the instructions[10] from `spacy` on how to train your own model.

Since nouns belonging together in English, e.g. "customer service", are naturally just one word in Swedish "kundtjänst", the first pre-processing step that merge nouns together in the **ABOM** model is not necessary for our Swedish equivalent.

The **ABOM-Swe** model can therefore be summarized in just two steps

1. Given the relations in a dependency parse tree and set of grammatical rules, extract opinion pairs for the text;
2. Classify the sentiment polarity for the extracted opinion pairs.

The main step of the model, i.e. the opinion pair extractions, works in a similar fashion as for the English models. Given the relations from the dependency parser, we extract opinion pairs according to some rules like the one exemplified in the section for the **ABOM** model. The rules are not identical to the ones for the English models, but to a large extent have the same logic.

To do the final step, which is classifying the sentiment polarity (positive or negative) for the opinion pairs that the model has found, we use the same approach as we did for the

[8] Regarding the inner workings of BERT, see section 4.1.3
[9] `https://github.com/UniversalDependencies/UD_Swedish-Talbanken`
[10] `https://spacy.io/usage/training`

English aspect-based opinion mining models. Here we instead use a Swedish Vader Sentiment Analysis lexicon[11] and our Swedish BERT model **BERT-Swe**[12] trained on the (Swedish) **Trustpilot** dataset.

4.4.2 ABOM-SGC-Swe

ABOM-SGC-Swe is our second Swedish aspect-based opinion mining model. Just as **ABOM-SGC** extends the **ABOM** model with an extra spell and grammar correction pre-processing step, this model adds the same grammar and spell correction step to the **ABOM-Swe** model. The idea is still to simplify the task for the dependency parsing model.

This model also use `LanguageTool`, but the Swedish variant, to correct eventual spelling mistakes and grammatical errors.

The **ABOM-SGC-Swe** model can thus be summarized in three steps:

1. Pre-processing step to automatically correct spelling mistakes and grammatical errors;
2. Given the relations in a dependency parse tree and set of grammatical rules, extract opinion pairs for the text;
3. Classify the sentiment polarity for the extracted opinion pairs.

The other parts of the model are identical to the **ABOM-Swe** model.

[11]`https://pypi.org/project/vaderSentiment-swedish/`

[12]As can be seen in the Results section, this is our best performing text classification model

Chapter 5

Datasets

5.1 Text classification datasets

5.1.1 Amazon1

The dataset which we denote by **Amazon1** is a subset of the Amazon product dataset (McAuley, 2020), a dataset consisting of user reviews (including star ratings) from Amazon. In total, **Amazon1** consist of 46,442 product reviews, and the distribution of star ratings are shown in Figure 5.1.

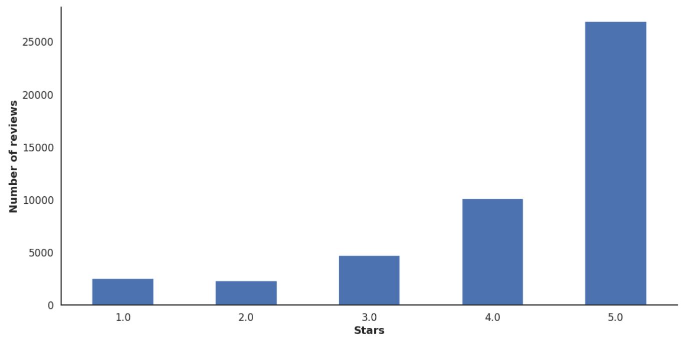


Figure 5.1: Distribution of star ratings for the **Amazon1** dataset

After a smaller qualitative analysis of the review texts, we found it reasonable to consider the reviews with 1-3 stars as negative and the ones with 4-5 stars as positive. This results in the following number of negative respective positive reviews, shown in Table 5.1 below. We can see that the dataset is very skewed with around 80 percent positives.

Table 5.1: Number of positive and negative reviews in the **Amazon1** dataset

Negative reviews	Positive reviews	Total number of reviews
9,416 (20.3%)	37,026 (79.7%)	46,442

The Amazon product dataset consists of reviews from several different product categories with different amount of reviews (McAuley, 2020).

Our **Amazon1** dataset is a stratified subsample with respect to the categories, meaning that the proportion of reviews from the different categories is the same as in the original full dataset. In Figure 5.2, we can see the distribution of categories, as well as the proportion of negative and positive reviews within each category.

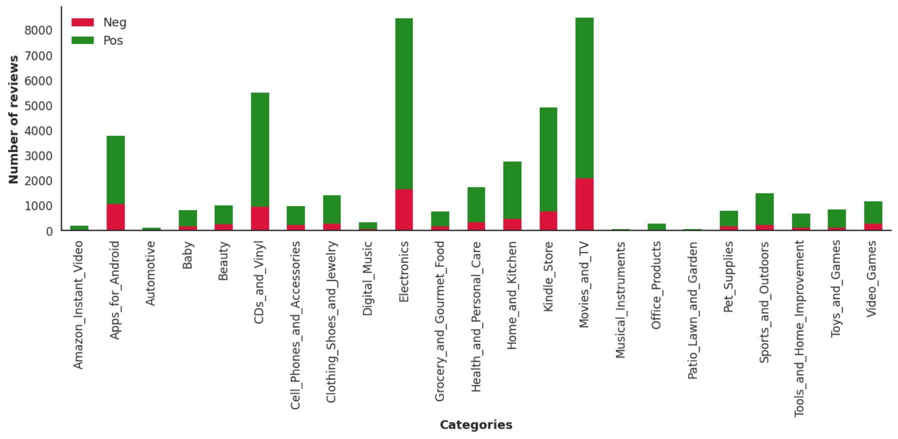


Figure 5.2: Distribution of categories and positive/negative reviews for the **Amazon1** dataset

Worth noting is that the category *Books* has been excluded since it is so much bigger than the rest of the categories.

Next, we have a look at the length of the reviews in terms of number of words. Some summarizing statistics are shown in Table 5.2.

Table 5.2: Statistics for the number of words in the reviews for the **Amazon1** dataset. All values are in number of words

Average length	Standard deviation	Shortest	Longest
118	159	1	5367

Even if the longest review is 5,367 words long, an average length of 118 words indicates that most of the review texts are not that long. That this is the case can be seen clearly in Figure 5.3 displaying the distribution of the number of words in the review texts.

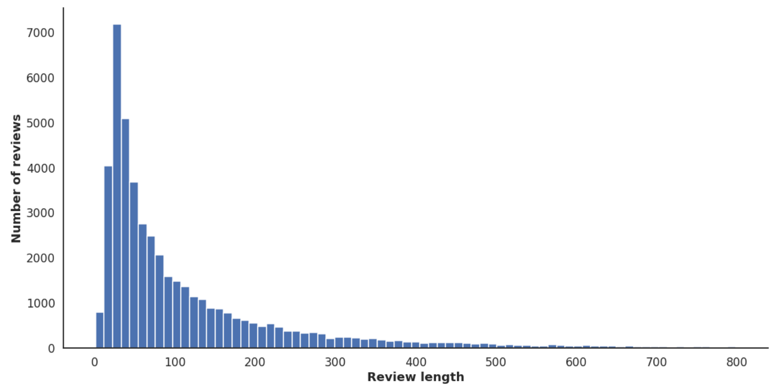


Figure 5.3: Distribution of number of words in the review texts in the **Amazon1** dataset

Finally, we list the top ten most common words in the dataset and their respective occurrences. This is listed in Table 5.3. Note that stopwords, i.e. very common words like *the*, *a* and *for* have been excluded from this list.

Table 5.3: The ten most common words (excluding stop-words) in the **Amazon1** dataset

Word	Occurrences
One	21,398
Like	20,471
Great	14,528
Good	13,814
Would	13,229
Get	12,697
Really	11,909
Love	9,706
Much	9,215
Even	8,775

Here we can see many words that express positive sentiment, such as *like*, *great* and *love*, which is consistent with the fact that roughly 80 percent of the reviews in the dataset are positive.

5.1.2 Amazon2

Our second dataset, denoted by **Amazon2**, is just another stratified subsample (w.r.t. the categories) of the Amazon product dataset (McAuley, 2020). This dataset has no overlap with the previous **Amazon1** dataset, so these are just two datasets coming from the exact same data distribution. We clearly see when comparing the same figures and statistics here for this dataset that they are very similar.

The reason for using two datasets from the same data distribution is to ensure the stability of the models to different subsets of the data, as well as making sure that the randomly sampled dataset has captured the overall characteristics of the full original dataset.

The **Amazon2** dataset has a total of 46,422 reviews, with the following distribution of star ratings, shown in Figure 5.4.

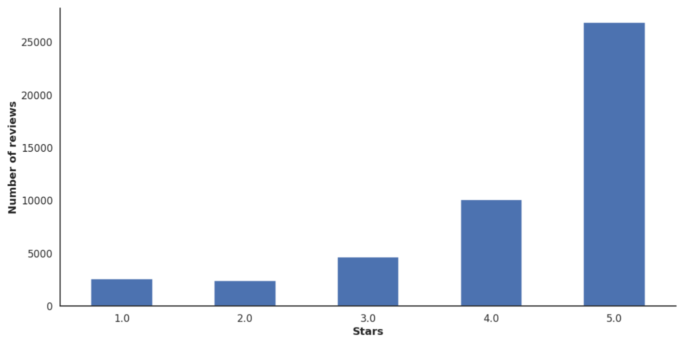


Figure 5.4: Distribution of star ratings for the **Amazon2** dataset

Again letting the reviews with 1-3 stars be considered as negative and the reviews with 4-5 stars as positive, we get the following amount of negative and positive reviews, shown in Table 5.4. We have the same skewedness with around 80 percent positives in this second Amazon dataset as well.

Table 5.4: Number of positive and negative reviews in the **Amazon2** dataset

Negative reviews	Positive reviews	Total number of reviews
9,521 (20.5%)	36,901 (79.5%)	46,422

We also look at the distribution over the categories, shown in Figure 5.5 below, which due to the stratified sampling is more or less identical to the other Amazon dataset. The proportion of positive and negative reviews within each category also seems to be very similar.

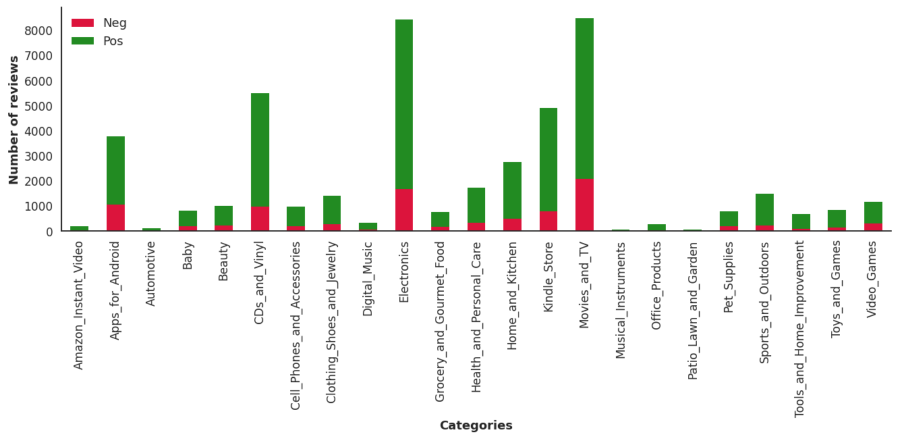


Figure 5.5: Distribution of categories and positive/negative reviews for the **Amazon2** dataset

We compute the same statistics for the lengths of the review texts, as well as display the distribution of the number of words in terms of a histogram. This can be seen in Table 5.5 and Figure 5.6 respectively.

Table 5.5: Statistics for the number of words in the reviews for the **Amazon2** dataset. All values are in number of words

Average length	Standard deviation	Shortest	Longest
118	158	1	5771

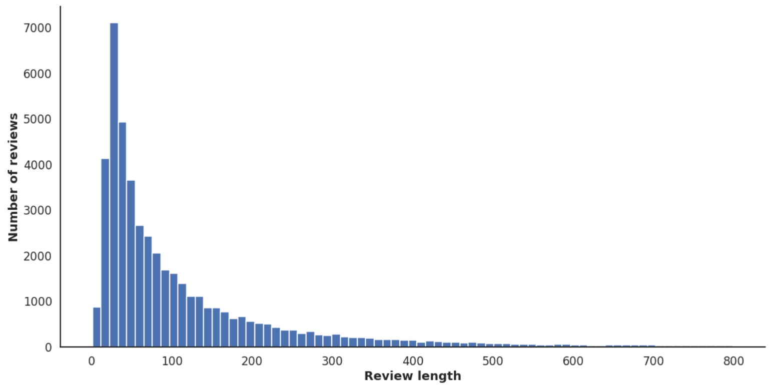


Figure 5.6: Distribution of number of words in the review texts in the **Amazon2** dataset

Also in terms of the number of words in the review texts, we can see that both Amazon datasets are very similar. We end with a list of the ten most common words (excluding stop-words) in this dataset. This can be seen in Table 5.6 below. We also note that the list is actually identical (apart from the exact values) to the one for the first Amazon dataset.

Table 5.6: The ten most common words (excluding stop-words) in the **Amazon2** dataset

Word	Occurrences
One	21,409
Like	20,369
Great	14,247
Good	14,027
Would	13,445
Get	12,877
Really	11,904
Love	9,674
Much	9,046
Even	8,947

5.1.3 Yelp1

The dataset we denote by **Yelp1** is a subset of the Yelp Kaggle dataset.[1] The dataset consists of reviews of services, such as restaurants, shopping, entertainment, dentists etc. In contrast to the Amazon datasets, this dataset is not categorized by area.

This subset of the whole Yelp Kaggle dataset has a total of 52,615 reviews, and just as for the Amazon datasets, the reviews are rated on a scale from 1 to 5 stars. The distribution of star ratings in the **Yelp1** dataset are shown in Figure 5.7.

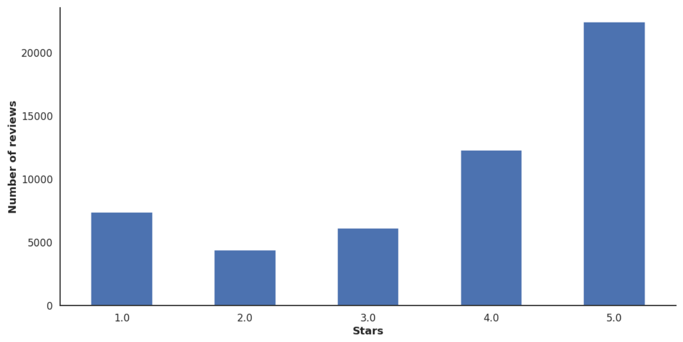


Figure 5.7: Distribution of star ratings for the **Yelp1** dataset

We also consider reviews with 1-3 stars to be negative and reviews with 4-5 stars as positive. The number of negative and positive reviews in the dataset can then be seen in Table 5.7. This dataset is a bit more balanced than the Amazon datasets, however it is still skewed towards the positive class.

Table 5.7: Number of positive and negative reviews in the **Yelp1** dataset

Negative reviews	Positive reviews	Total number of reviews
17,889 (34.0%)	34,726 (66.0%)	52,615

We also have a look at the lengths of the reviews. The same summarizing statistics as before can be seen in Table 5.8. The average review length is bit shorter compared to the Amazon datasets and follow a similar distribution but with a lower standard deviation, which we also see in Figure 5.8.

Table 5.8: Statistics for the number of words in the reviews for the **Yelp1** dataset. All values are in number of words

Average length	Standard deviation	Shortest	Longest
113	106	1	1013

[1]`https://www.kaggle.com/yelp-dataset/yelp-dataset`

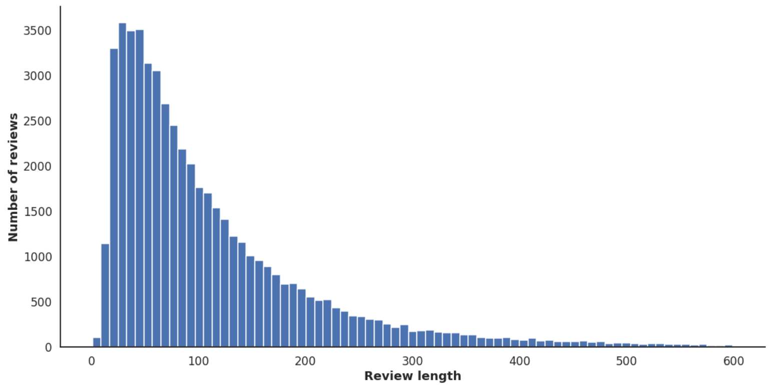


Figure 5.8: Distribution of number of words in the review texts in the **Yelp1** dataset

Looking at the **Yelp1** dataset's ten most common words, excluding stop-words, in Table 5.9, we can see that we have many words expressing positive sentiments. However, there are also words that we consider to be aspect words, such as *food*, *place* and *service*. This is probably due to the fact that the Yelp reviews specifically targets restaurants and local businesses in contrast to specific products.

Table 5.9: The ten most common words (excluding stop-words) in the **Yelp1** dataset

Word	Occurrences
Place	20,416
Food	19,389
Great	17,936
Like	17,849
Good	17,784
Get	17,106
One	16,059
Would	15,750
Time	13,302
Service	13,265

5.1.4 Yelp2

The **Yelp2** dataset is another subset of the Yelp Kaggle dataset[2], and the reason for using two datasets from the same data distribution is the same as for the Amazon datasets.

As shown by the figures and tables in this section, this dataset has almost the same characteristics as the **Yelp1** dataset. In total, the **Yelp2** dataset consist of 52,617 reviews and has the following star rating distribution, shown in Figure 5.9.

[2]`https://www.kaggle.com/yelp-dataset/yelp-dataset`

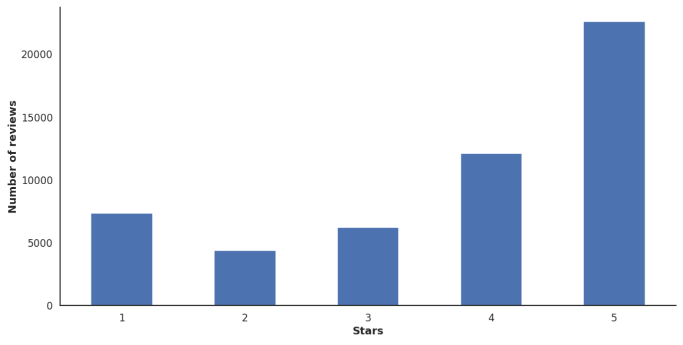


Figure 5.9: Distribution of star ratings for the **Yelp2** dataset

As before, letting reviews with 1-3 stars be negative and the ones with 4-5 stars be positive, we get the following number of negative and positive reviews in the dataset, shown below in Table 5.10. Again we have almost the same skewedness towards positive reviews.

Table 5.10: Number of positive and negative reviews in the **Yelp2** dataset

Negative reviews	Positive reviews	Total number of reviews
17,925 (34.1%)	34,692 (65.9%)	52,617

Table 5.11 and Figure 5.10 below, show us some summarizing statistics of the review lengths respective the distribution of number of words in the reviews.

Table 5.11: Statistics for the number of words in the reviews for the **Yelp2** dataset. All values are in number of words

Average length	Standard deviation	Shortest	Longest
112	106	1	1028

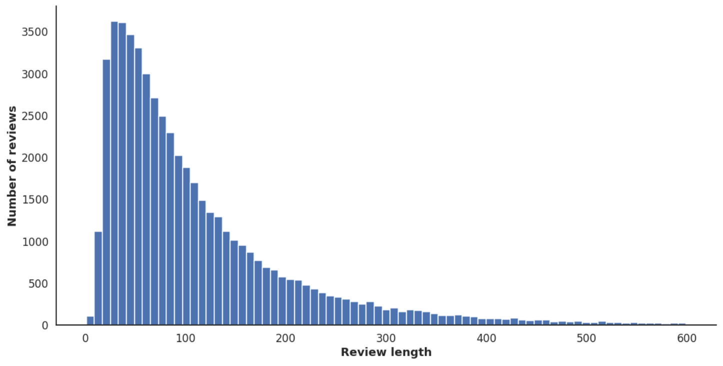


Figure 5.10: Distribution of number of words in the review texts in the **Yelp2** dataset

Finally, we list the ten most common (non-stopword) words and their corresponding occurrences below in Table 5.12. This list is almost identical to the one for the **Yelp1** dataset in Table 5.9.

Table 5.12: The ten most common words (excluding stop-words) in the **Yelp2** dataset

Word	Occurrences
Place	20,217
Food	19,663
Great	17,935
Like	17,854
Good	17,513
Get	17,019
One	16,138
Would	15,535
Really	13,336
Service	13,189

5.1.5 Trustpilot

This dataset, which we denote by **Trustpilot**, also consists of user reviews from different companies. In contrast to the Amazon and Yelp datasets that all have reviews in English, this dataset has Swedish review texts instead. We have constructed the dataset ourselves and all the reviews come from the consumer review website Trustpilot (Trustpilot, 2020b).

In total, there are 21,851 user reviews (including star ratings) in the **Trustpilot** dataset, and we can see the distribution of star ratings below in Figure 5.11.

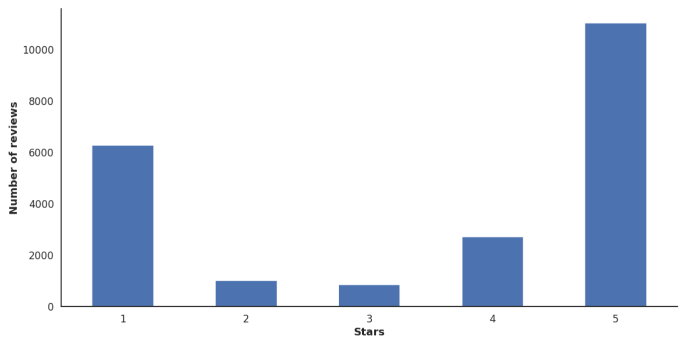


Figure 5.11: Distribution of star ratings for the **Trustpilot** dataset

We see that a clear majority of the reviews have either 1 or 5 stars. As we have done for the other datasets, we consider reviews with 1-3 stars as negative and reviews with 4-5 stars as positive. This yields the following amount of negatives and positives, shown in Table 5.13.

Table 5.13: Number of positive and negative reviews in the **Trustpilot** dataset

Negative reviews	Positive reviews	Total number of reviews
8,112 (37.1%)	13,739 (62.9%)	21,851

This dataset has some skewedness as well (around 63 percent positives), but is the most balanced of our five text classification datasets.

The **Trustpilot** dataset reviews comes from 73 different companies, divided into 14 categories. The distribution of the categories in the dataset can be seen in Figure 5.12. In this figure we can also see the distribution of negatives and positives within each category.

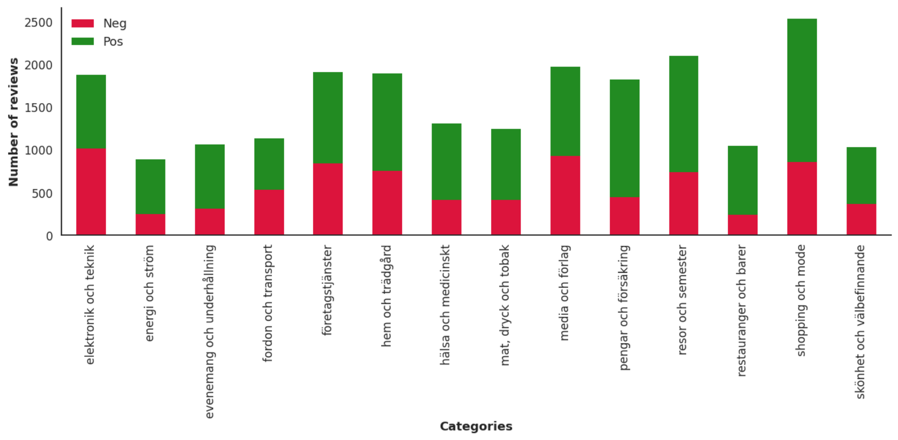


Figure 5.12: Distribution of categories and positive/negative reviews for the **Trustpilot** dataset

A complete composition of the dataset in terms of both categories and companies, can be found in Appendix A.

We end by looking at the length of the reviews in the dataset. We list a few summarizing statistics in Table 5.14 below.

Table 5.14: Statistics for the number of words in the reviews for the **Trustpilot** dataset. All values are in number of words

Average length	Standard deviation	Shortest	Longest
37	59	1	1549

These Swedish review texts, with an average length of 37 words, seem to be rather short. Looking at the distribution of the number of words in Figure 5.13 below, we can also see that almost all reviews are shorter than 200 words.

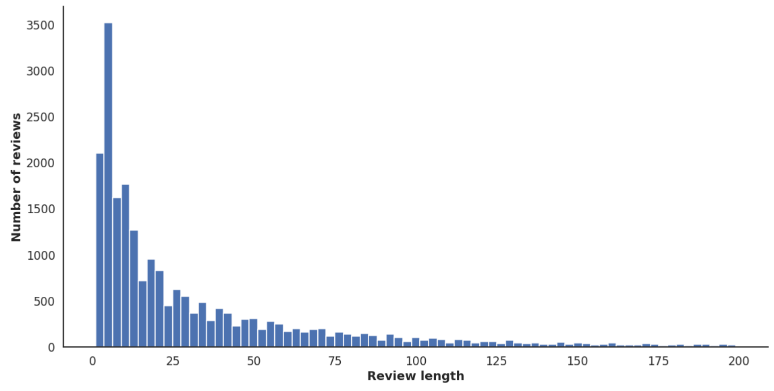


Figure 5.13: Distribution of number of words in the review texts in the **Trustpilot** dataset

At last, we also list the top ten words in the **Trustpilot** dataset.

Table 5.15: The ten most common words (excluding stop-words) in the **Trustpilot** dataset. English translation in parenthesis

Word	Translation	Occurrences
Bra	(Good)	6912
Fick	(Got)	4448
Snabb	(Quick)	2868
Få	(Get)	2510
Kommer	(Come)	2482
Aldrig	(Never)	2168
Leverans	(Delivery)	1899
Bara	(Only)	1855
Väldigt	(Very)	1627
Alltid	(Always)	1441

5.2 Aspect-based opinion mining datasets

5.2.1 SemEvalABSA-En

The first aspect-based opinion mining dataset, denoted by **SemEvalABSA-En**, comes from the competition SemEval-2016 Task 5 - Aspect-Based Sentiment Analysis. More specifically, it is the Restaurants test set (with Gold annotations) for Subtask 1 (SemEval-2016 Task 5, 2020).

This dataset consist of English sentences from reviews on restaurants, that have been annotated with mentioned aspects (targets) and the polarity (positive, negative or neutral) of the expressed sentiment towards these aspects. An example sentence with these annotations are displayed (in XML-code) below.

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<sentence id="en_BlueRibbonSushi_478218345:2">
    <text>It has great sushi and even better service.</text>
    <Opinions>
        <Opinion target="sushi" category="FOOD#QUALITY"
            polarity="positive" from="13" to="18"/>
        <Opinion target="service" category="SERVICE#GENERAL"
            polarity="positive" from="35" to="42"/>
    </Opinions>
</sentence>
```

In total, this dataset has 676 different sentences and together they have a total of 650 annotated aspect mentions with corresponding sentiment polarities. In Table 5.16 below, we can see the exact distribution of positive, negative and neutral aspect mentions.

Table 5.16: Number of positive, negative and neutral aspect mentions in the **SemEvalABSA-En** dataset

Positive mentions	Negative mentions	Neutral mentions	Total mentions	Sentences
483 (74.3%)	135 (20.8%)	32 (4.9%)	650	676

The number of mentioned aspects in each sentence vary between sentences, and Figure 5.14 shows the distribution of number of aspect mentions in the **SemEvalABSA-En** dataset. We see that zero, one or two aspect mentions per sentence is the most common.

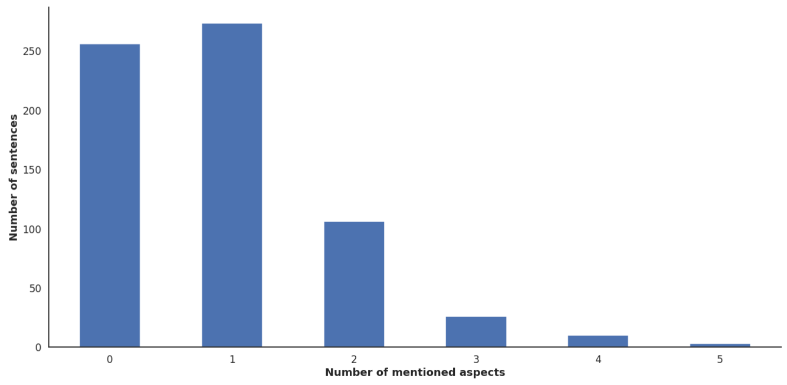


Figure 5.14: Distribution of number of aspect mentions in the sentences in the **SemEvalABSA-En** dataset

We also display the ten most common aspect words in the **SemEvalABSA-En** dataset, and their distribution of positive, negative and neutral mentions. This is shown in Figure 5.15 below.

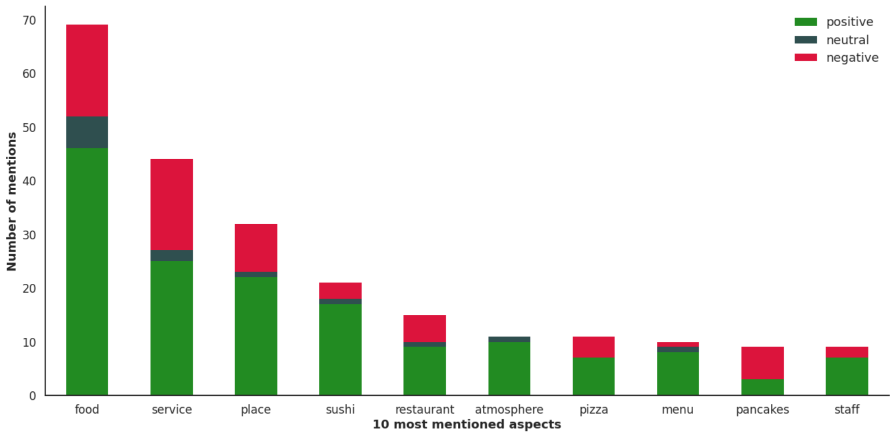


Figure 5.15: Distribution of positive, negative and neutral aspect mentions for the 10 most mentioned aspects in the **SemEvalABSA-En** dataset

The most common aspect words are *food*, *service* and *place*. All of these words are typically used to describe any kind of restaurant, so it makes sense that they are the most common ones.

As we did for the text classification datasets, we end by looking at the lengths of the sentences in the dataset. In this case, we look at sentence level instead of the entire review. Some summarizing statistics are shown in Table 5.17, while in Figure 5.16 below, the distribution of the sentence lengths is displayed.

Table 5.17: Statistics for the number of words in the sentences for the **SemEvalABSA-En** dataset. All values are in number of words

Average length	Standard deviation	Shortest	Longest
12	8	1	69

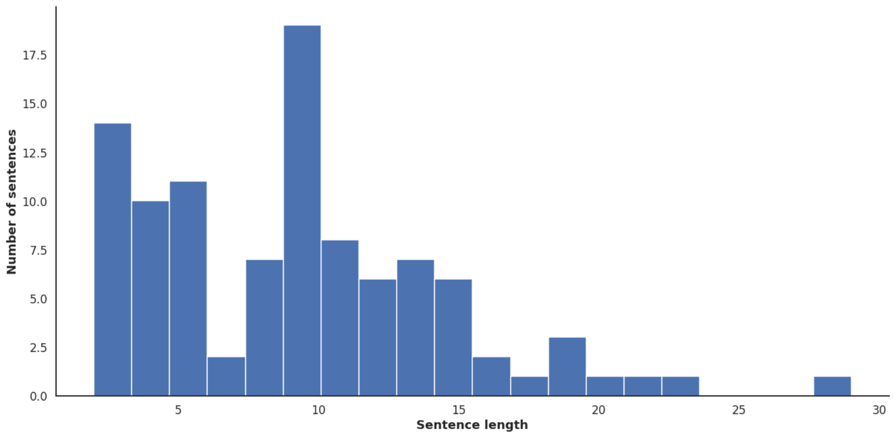


Figure 5.16: Distribution of number of words in the sentences in the **SemEvalABSA-En** dataset

5.2.2 SemEvalABSA-Swe

Our second aspect-based opinion mining dataset, which we denote by **SemEvalABSA-Swe**, comes from the competition SemEval-2014 Task 4 - Aspect-Based Sentiment Analysis. Here we have used the Restaurants trial set (SemEval-2014 Task 4, 2020).

This dataset also consist of English sentences from restaurant reviews annotated with mentioned aspects and corresponding sentiment polarities, but with a slightly different annotation scheme compared to the SemEval-2016, as shown in this example below

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<sentence id="3440">
    <text>Even though its good seafood, the prices are too high.</text>
    <aspectTerms>
        <aspectTerm term="seafood" polarity="positive" from="21" to="28"/>
        <aspectTerm term="prices" polarity="negative" from="34" to="40"/>
    </aspectTerms>
    <aspectCategories>
        <aspectCategory category="food" polarity="positive"/>
        <aspectCategory category="price" polarity="negative"/>
    </aspectCategories>
</sentence>
```

However, in this case we have translated the dataset into Swedish in order to get a Swedish aspect-based opinion mining dataset. For the example above we get

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<sentence id="3440">
    <text>Trots dess goda skaldjur, är priserna för höga.</text>
    <aspectTerms>
        <aspectTerm term="skaldjur" polarity="positive" from="21" to="28"/>
        <aspectTerm term="pris" polarity="negative" from="34" to="40"/>
    </aspectTerms>
    <aspectCategories>
        <aspectCategory category="mat" polarity="positive"/>
        <aspectCategory category="pris" polarity="negative"/>
    </aspectCategories>
</sentence>
```

This dataset is smaller than **SemEvalABSA-En**, with a total of only 100 different sentences. Together, these sentences have 96 aspect mentions with expressed sentiment polarities. The exact distribution of positive, negative and neutral aspect mentioned can be seen below in Table 5.18.

Table 5.18: Number of positive, negative and neutral aspect mentions in the **SemEvalABSA-Swe** dataset

Positive mentions	Negative mentions	Neutral mentions	Total mentions	Sentences
68 (70.8%)	18 (18.8%)	10 (10.4%)	96	100

Also, Figure 5.17 shows us the distribution of number of aspect mentions in the sentences in the **SemEvalABSA-Swe** dataset.

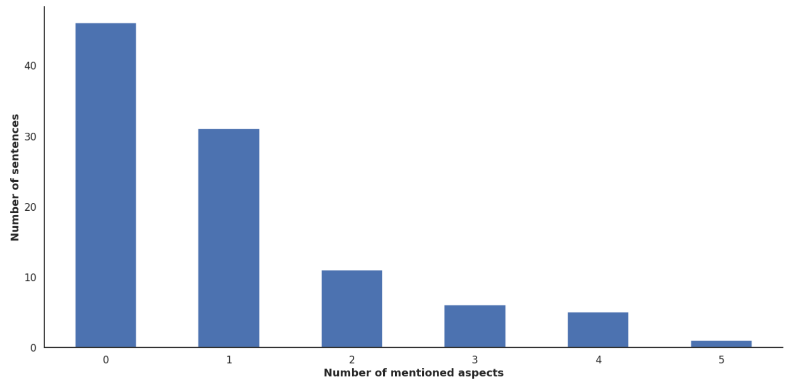


Figure 5.17: Distribution of number of aspect mentions in the sentences in the **SemEvalABSA-Swe** dataset

In Figure 5.18, we see the ten most mentioned aspect words and their corresponding distribution of positive, negative and neutral mentions. Also in this case, the most common words are *mat (food)*, *pris (price)* and *service*. Which are typical words used to describe any kind of restaurant.

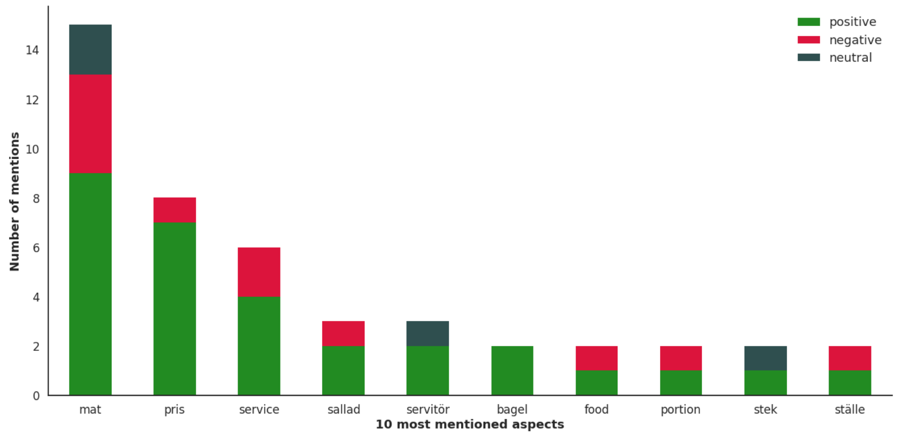


Figure 5.18: Distribution of positive, negative and neutral aspect mentions for the 10 most mentioned aspects in the **SemEvalABSA-Swe** dataset

Finally, a few statistics summarizing the length of the sentences in the **SemEvalABSA-Swe** dataset can be seen in Table 5.19, while Figure 5.19 shows the distribution of number of words in the sentences.

Table 5.19: Statistics for the number of words in the sentences for the **SemEvalABSA-Swe** dataset. All values are in number of words

Average length	Standard deviation	Shortest	Longest
9	5	2	29

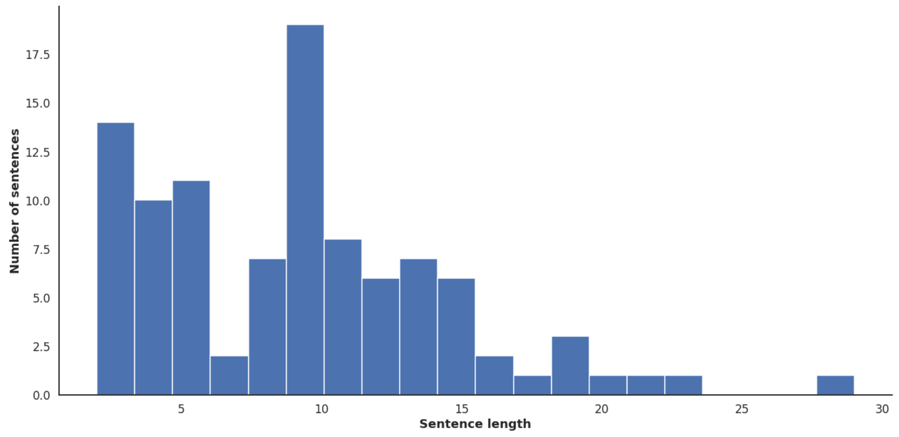


Figure 5.19: Distribution of number of words in the sentences in the **SemEvalABSA-Swe** dataset

5.3 Demo datasets

The other datasets before are used for training and/or evaluation of text classification and aspect-based opinion mining models. The demo datasets will instead be used as practical use cases for our models and for creating the plots and tables in our user interface.

5.3.1 McDonalds

The first one, denoted by **McDonalds**, is a dataset consisting of 1,555 reviews in English for the fast food restaurant McDonald's, all coming from the user review website Trustpilot (Trustpilot, 2020a).

Below in Figure 5.20, we can see the distribution of star ratings. There are a majority of reviews with bad ratings, so we can expect our models to predict many of these as negative.

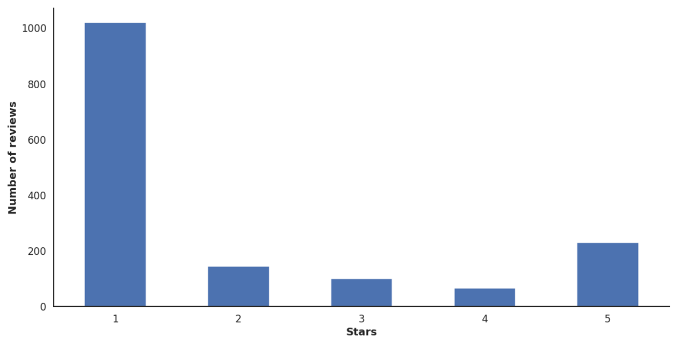


Figure 5.20: Distribution of star ratings for the **McDonalds** dataset

Table 5.20 and Figure 5.21 below, show us some summarizing statistics of the review lengths respective the distribution of number of words in the reviews in the **McDonalds** dataset.

Table 5.20: Statistics for the number of words in the reviews for the **McDonalds** dataset. All values are in number of words

Average length	Standard deviation	Shortest	Longest
71	74	1	844

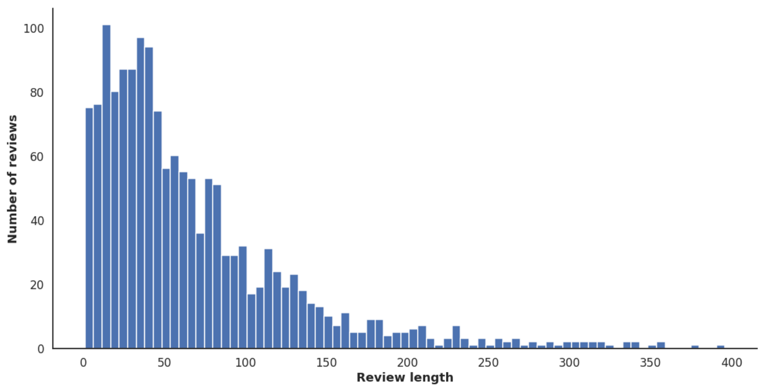


Figure 5.21: Distribution of number of words in the reviews in the **McDonalds** dataset

Finally, we look at the most common (non-stopword) words in this dataset. This can be seen in Table 5.21.

Table 5.21: The ten most common words (excluding stop-words) in the **McDonalds** dataset

Word	Occurrences
Food	759
Order	509
McDonald's	456
Get	418
Staff	388
Like	340
Service	309
Time	307
One	304
Went	284

Looking at these words, we might expect words like *food*, *order*, *McDonald's*, *staff*, *service* and *time* among the most commonly mentioned aspect words extracted by our models. Given the clear majority of negative reviews, it is likely that our models will find the overall sentiment of many of these aspect words to be negative as well.

5.3.2 Qliro

The second demo dataset, denoted by **Qliro**, consists of 1,471 reviews in Swedish for the payment service Qliro, all coming from the user review website Trustpilot (Trustpilot, 2020b).

We look at the distribution of star ratings for the reviews in the dataset. This is shown in Figure 5.22 below. Here we instead have a fairly even split between very bad and great reviews, so we can expect the same split between negative and positive predictions from our models.

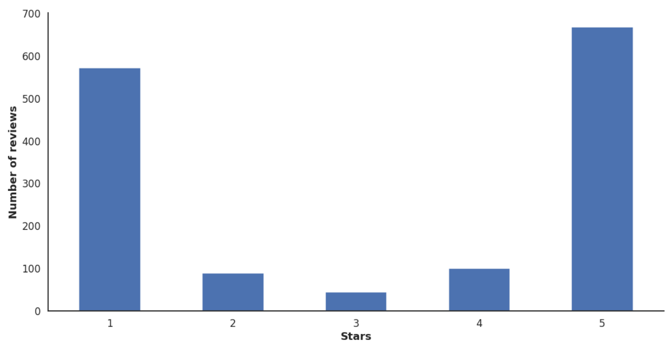


Figure 5.22: Distribution of star ratings for the **Qliro** dataset

Next we have a look at the length of the reviews. Table 5.22 shows us some summarizing statistics while Figure 5.23 displays the distribution of number of words in the reviews.

Table 5.22: Statistics for the number of words in the reviews for the **Qliro** dataset. All values are in number of words

Average length	Standard deviation	Shortest	Longest
26	36	1	311

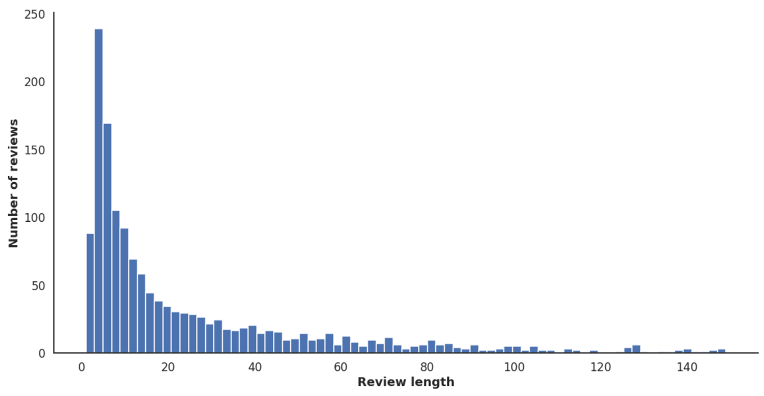


Figure 5.23: Distribution of number of words in the reviews in the **Qliro** dataset

We end with a list of the most common words (excluding stopwords) in the **Qliro** dataset, and this is shown in Table 5.23 below.

Table 5.23: The ten most common words (excluding stop-words) in the **Trustpilot** dataset. English translation in parenthesis

Word	Translation	Occurrences
Qliro		344
Faktura	(Invoice)	297
Betala	(Pay)	296
Bra	(Good)	288
Fick	(Got)	215
Aldrig	(Never)	178
Mer	(More)	98
Bara	(Only)	95
Enkelt	(Simple)	95
Klarna[3]		89

Words like *Qliro*, *faktura* and *Klarna* are likely to be among the most commonly mentioned aspect words extracted by our models.

[3]Klarna is the name of Qliro's biggest competitor

Chapter 6

Evaluation

6.1 Experimental setup

6.1.1 Text classification

Our text classification experiments consist of training and evaluating all our text classification models. We have trained and evaluated all of the English models **LR**, **LSTM**, **BERT**, **RoBERTa** and **XLNet**, on every English text classification dataset, i.e. on **Amazon1**, **Amazon2**, **Yelp1** and **Yelp2**. Similarly, we have trained and evaluated all the Swedish text classification models **LR-Swe**, **LSTM-Swe**, **BERT-Swe** and **BERT-ML** on the Swedish text classification dataset **Trustpilot**.

The review texts in these datasets have been used as the inputs to the models, while the 1/0 (positive/negative) labels constructed from the star ratings (see the Dataset section), are used as the true target values. In all cases, we have used a dataset split of 60/20/20 for training/validation/testing during these experiments, and all models have used the same split.

We did not put much focus on the tuning of various model and training parameters. The validation dataset was just used to monitor the number of epochs when training, while the standard parameter settings in `tensorflow` and `simpletransformers` was used for all other parameters.

After training, we evaluated all models on the test set and it is these results that we have presented in the Results section. Macro F1-score has been our main metric for evaluating the performance of the models. However, we have also included the precision, recall (both macro averages) and accuracy in the result tables. Also, we looked at confusion matrices for the models, and included the confusion matrix for the best model for every dataset in the Results section.

6.1.2 Aspect-based opinion mining

Besides the **ABSA-E2E** model, which is an already complete model (with ready-to-use scripts for training and evaluation) made by Li et al. (2019), all our aspect-based opinion mining models are unsupervised models. Our experiments therefore consist of just evaluating all our models, presented in the Model section, using all the aspect-based opinion mining datasets from the Dataset section.

All the English models **ABOM**, **ABOM-SGC** and **ABSA-E2E** have been evaluated on the English dataset **SemEvalABSA-En**, and the Swedish models **ABOM-Swe** and **ABOM-SGC-Swe** have both been evaluated on the Swedish dataset **SemEvalABSA-Swe**. To compare the performance of our English and Swedish models, our English models **ABOM** and **ABOM-SGC** have also been evaluated on the original English (non-translated) version of the Swedish **SemEvalABSA-Swe** dataset.

When evaluating the models, the different tasks aspect extraction and aspect sentiment classification have been evaluated separately for all models and datasets. Starting with aspect extraction, i.e. the task of finding the aspect words that are being addressed in the text, we have evaluated this using F1-score. However, as can be seen in the Results section, we have also included the precision and recall measures for our own models. For clarity, we have the following interpretation of true positives (TP), false positives (FP) and false negatives (FN) in the case of aspect extraction

- **TP** – correctly extracted aspect words
- **FP** – extracted aspect words that has not been annotated
- **FN** – annotated aspect words that has not been extracted

The English dataset **SemEvalABSA-En** is a test set for the SemEval-2016 Task 5 competition. We can thus compare the F1-scores for our English models with the ones from the participating systems. All these results have been presented by Pontiki et al. (2016), and are included in our Results section as well.

We also want to evaluate the performance of our models on the task of aspect sentiment classification, i.e. classifying the extracted aspect words as positive or negative. When doing this, we have only considered the aspect words correctly extracted (i.e. true positives). The datasets also contain aspects with neutral sentiment, but since our model only classify aspects as positive or negative these neutral mentions have been disregarded. Due to different ways of evaluating, the results from systems competing in the SemEval-2016 competition have not been included for comparison.

For aspect sentiment classification, we present the precision, recall, F1-score (all macro averages) and accuracy, just as we did for the text classification experiments.

6.1.3 Demo cases

Besides the experiments presented above, we will also demonstrate our user application. For this purpose, we will use the demo datasets **McDonalds** and **Qliro**, both presented in the Dataset section.

Given a dataset with review texts (and date of the reviews), the user application uses our models for predictions. These results are then aggregated in various ways and presented in different graphs and exploratory tables. The idea is that this should provide insights that can be used for a company analysis.

Screenshots of these graphs and tables are presented in the Results section.

6.2 Results

6.2.1 Text classification

In this section, we report the results of our text classification experiments. For every dataset, we present the precision, recall, F1-score and accuracy for all our text classification models. Precision, recall and F1-score are all macro averages (arithmetic average of the two classes' individual scores). To complement the result tables, a confusion matrix for the best model according to macro F1-score has been included for every dataset.

We start with the results for the two Amazon datasets, **Amazon1** and **Amazon2**. These results are shown in Tables 6.1 and 6.2, and the corresponding confusion matrices for the best models are displayed in Figures 6.1 and 6.2 respectively.

Table 6.1: Results on the **Amazon1** dataset

Model	Precision	Recall	F1-score	Accuracy
LR	0.63	0.60	0.61	0.78
LSTM	0.76	0.68	0.70	0.84
BERT	0.82	0.80	0.81	0.88
RoBERTa	**0.87**	**0.83**	**0.85**	**0.91**
XLNet	0.84	**0.83**	0.84	0.90

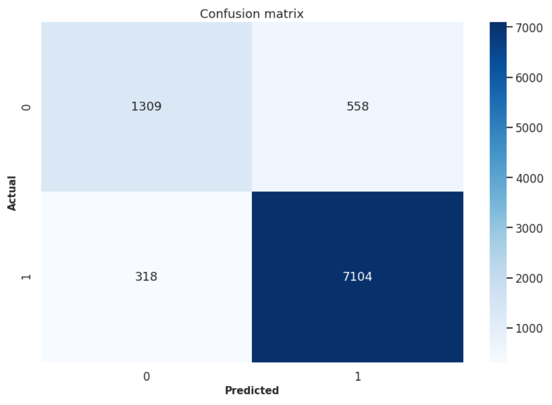


Figure 6.1: Confusion matrix for the RoBERTa model on the **Amazon1** dataset

Table 6.2: Results on the **Amazon2** dataset

Model	Precision	Recall	F1-score	Accuracy
LR	0.62	0.59	0.59	0.77
LSTM	0.78	0.64	0.67	0.83
BERT	0.83	0.80	0.81	0.88
RoBERTa	**0.87**	**0.84**	**0.85**	**0.91**
XLNet	0.86	0.82	0.84	0.90

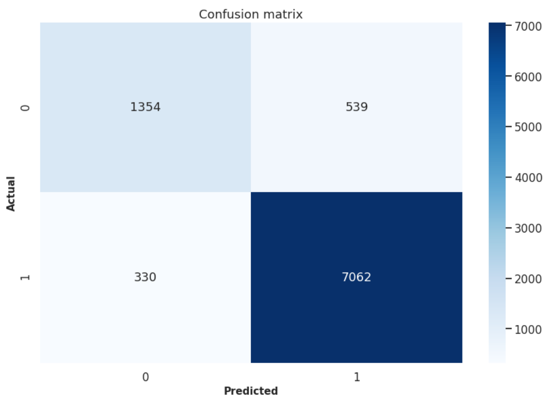


Figure 6.2: Confusion matrix for the RoBERTa model on the **Amazon2** dataset

We can see that the results on both Amazon datasets are very similar, which is fortunate since the two datasets come from the same data distribution. The **Amazon2** dataset appears to be a bit more difficult for the baseline models **LR** and **LSTM**, while the transformer models produce the same F1-scores on both datasets. We can note that **RoBERTa** was the best model, just slightly better than **XLNet**.

Next, we have the results for the both Yelp datasets, **Yelp1** and **Yelp2**. Tables 6.3 and 6.4 present these results, and Figures 6.3 and 6.4 show us the confusion matrices for the best models.

Table 6.3: Results on the **Yelp1** dataset

Model	Precision	Recall	F1-score	Accuracy
LR	0.75	0.75	0.75	0.77
LSTM	0.87	0.85	0.86	0.87
BERT	0.89	0.89	0.89	0.90
RoBERTa	**0.92**	**0.91**	**0.91**	**0.92**
XLNet	**0.92**	**0.91**	**0.91**	**0.92**

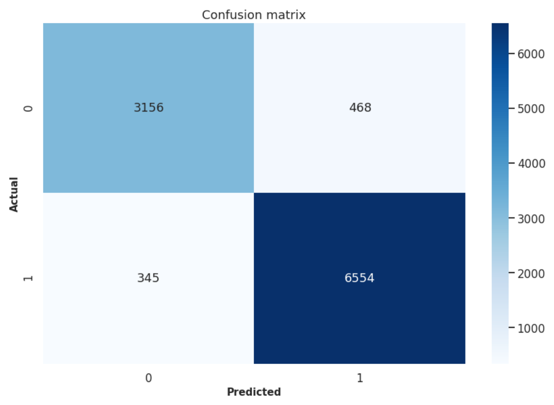


Figure 6.3: Confusion matrix for the XLNet model on the **Yelp1** dataset

Table 6.4: Results on the **Yelp2** dataset

Model	Precision	Recall	F1-score	Accuracy
LR	0.75	0.75	0.75	0.77
LSTM	0.86	0.85	0.86	0.87
BERT	**0.91**	0.90	0.90	0.91
RoBERTa	**0.91**	**0.91**	**0.91**	**0.92**
XLNet	**0.91**	**0.91**	**0.91**	**0.92**

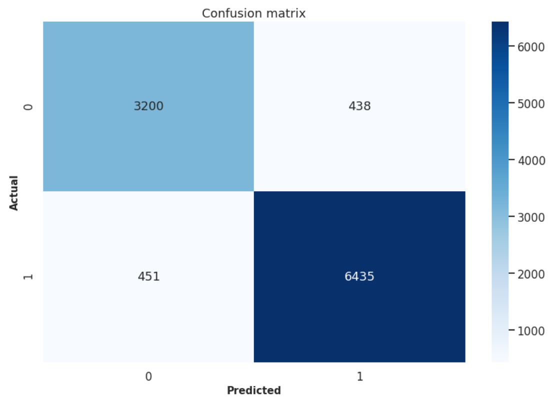


Figure 6.4: Confusion matrix for the XLNet model on the **Yelp2** dataset

Also for the two Yelp datasets, we have very stable results. Apart from the **BERT** model, which is slightly better on the **Yelp2** dataset, all models have the same F1-score on both datasets. All the models also perform better on these datasets compared to **Amazon1** and

Amazon2. Note that the confusion matrices display the results of the **XLNet** model on both datasets, but we could just as well have chosen the **RoBERTa** model since these two models perform equally well.

Finally, we have the results for the Swedish dataset, **Trustpilot**. Table 6.5 presents these results, while the confusion matrix for the best model is shown in Figure 6.5 below.

Table 6.5: Results on the **Trustpilot** dataset

Model	Precision	Recall	F1-score	Accuracy
LR-Swe	0.87	0.87	0.87	0.88
LSTM-Swe	0.92	0.92	0.92	0.93
BERT-Swe	**0.96**	**0.96**	**0.96**	**0.96**
BERT-ML	0.90	0.90	0.90	0.91

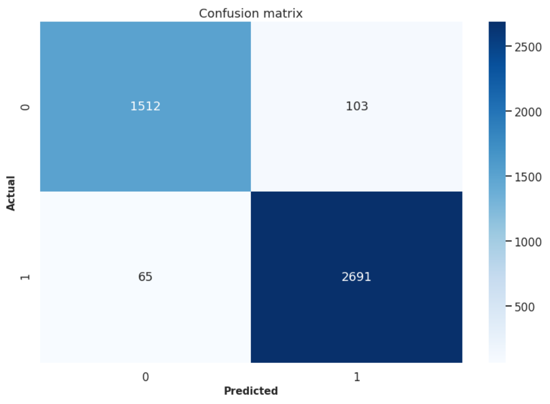


Figure 6.5: Confusion matrix for the BERT-Swe model on the **Trustpilot** dataset

The **Trustpilot** dataset appears to be easier than the Amazon and Yelp datasets, and as we can see, the results are extremely good. The best model **BERT-Swe** has an almost perfect score, and even the baseline models show great results.

6.2.2 Aspect-based opinion mining

Now we present the results for our aspect-based opinion mining experiments. The results are divided into different tables for aspect extraction and aspect sentiment classification, as described under Experimental setup. For our systems, we present precision, recall and F1-score for aspect extraction. For aspect sentiment classification, we include precision, recall and F1-score, all macro averages, as well as accuracy in the results.

We start with the English dataset **SemEvalABSA-En**, and the results for aspect extraction. Since this is the test set for the SemEval-2016 Task 5, we have included the F1-scores for all participating systems, as well as the competion's own baseline model denoted **Baseline/C** (Pontiki et al., 2016). All these aspect extraction results can be seen in Table 6.6 below.

Table 6.6: Aspect extraction results for the **SemEvalABSA-En** dataset

Model	Precision	Recall	F1-score
NLANG./U			**0.723**
AUEB-./U			0.704
UWB/U			0.671
UWB/C			0.669
GTI/U			0.666
Senti./C			0.665
ABSA-E2E	0.614	0.689	**0.649**
bunji/U			0.649
NLANG./C			0.639
DMIS/C			0.635
XRCE/C			0.620
AUEB-./C			0.616
ABOM	0.536	0.615	**0.573**
UWate./U			0.571
KnowC./U			0.568
ABOM-SGC	0.529	0.611	**0.567**
TGB/C			0.551
BUAP/U			0.503
Baseline/C			**0.441**
IHS-R./U			0.438
IIT-T./U			0.462
SeemGo/U			0.343

As we can see, our models outperforms the **Baseline/C** model with a large margin. They are also better than several of the participating systems. As expected, our unsupervised approaches are beaten by many of the participating implementations. This shows that a supervised approaches typically are better performing than unsupervised ones, when annotated training data is available. We can also note that our model without spell and grammar correction , **ABOM**, is slightly better than our model with spell and grammar correction, **ABOM-SGC**.

The performance of the **ABSA-E2E** model is good but not quite among the best ones. However, the model has been trained on SemEval competition data from not only 2016 but 2014 and 2015 as well. Training the model on only the training data from 2016 would probably improve the performance on this dataset (which is the test set for the 2016 competition).

We also look at the aspect sentiment classification results for our models, shown in Table 6.7 below. As explained in the Experimental setup section, this is only evaluated on correctly extracted aspects, and due to different setups the results from the participating SemEval-2016 systems are not included.

Table 6.7: Aspect sentiment classification results for the **SemEvalABSA-En** dataset

Model	Precision	Recall	F1-score	Accuracy
ABOM	0.784	0.781	0.782	0.865
ABOM-SGC	0.791	0.797	0.794	0.872

At least on the correctly extracted aspects, our models are quite good at classifying the correct sentiment polarities.

We move on with our Swedish dataset **SemEvalABSA-Swe**. To compare the performance of our Swedish and English aspect-based opinion mining models, the results of the English models **ABOM** and **ABOM-SGC** on the original (non-translated) dataset have been included. First we present the aspect extraction results. This is shown in Table 6.8 below.

Table 6.8: Aspect extraction results for the **SemEvalABSA-Swe** dataset. *The English models **ABOM** and **ABOM-SCG** have been evaluated on the original (non-translated) English dataset

Model	Precision	Recall	F1-score
ABOM-Swe	0.607	0.531	0.567
ABOM-SGC-Swe	0.588	0.521	0.553
ABOM*	0.660	0.708	0.683
ABOM-SGC*	0.657	0.698	0.677

For the Swedish models, spelling and grammar correction does not seem to improve the results of the **ABOM-Swe** model. In fact, **ABOM-SGC-Swe** performs worse than **ABOM-Swe**. We can also see that our English models are clearly better than their Swedish equivalents.

Finally, we present the aspect sentiment classification results in Table 6.9, which have been evaluated on the correctly extracted aspects only. Again we include the results of our English models evaluated on the original (non-translated) dataset. Our Swedish models are also quite good at classifying the correct sentiment polarities for extracted opinion pairs.

Table 6.9: Aspect sentiment classification results for the **SemEvalABSA-Swe** dataset. *The English models **ABOM** and **ABOM-SCG** have been evaluated on the original (non-translated) English dataset

Model	Precision	Recall	F1-score	Accuracy
ABOM-Swe	0.697	0.775	0.725	0.857
ABOM-SGC-Swe	0.709	0.842	0.749	0.875
ABOM*	0.812	0.906	0.848	0.910
ABOM-SGC*	0.802	0.864	0.827	0.894

6.2.3 Demo cases

We end the Result section by presenting screenshots of graphs and tables from our user application. All graphs and tables show various aggregated results based on predictions using our text classification and aspect-based opinion mining models on the two demo datasets **McDonalds** and **Qliro**.

McDonald's

We start with the English demo dataset **McDonalds**. These results are based on the predictions from the text classification model **RoBERTa** (trained on **Yelp1**) and the aspect-based opinion mining model **ABOM**.

The first graph, shown in Figure 6.6, is a stacked bar plot over the number of negative respective positive reviews that our text classification model has predicted, grouped by year. The graph also has a trend line of the proportion of positive reviews. Disregarding the large fluctuations in the first years due to very little data, it seems that the distribution between positive and negative reviews is stable at around 15-20 percent positive reviews every year.

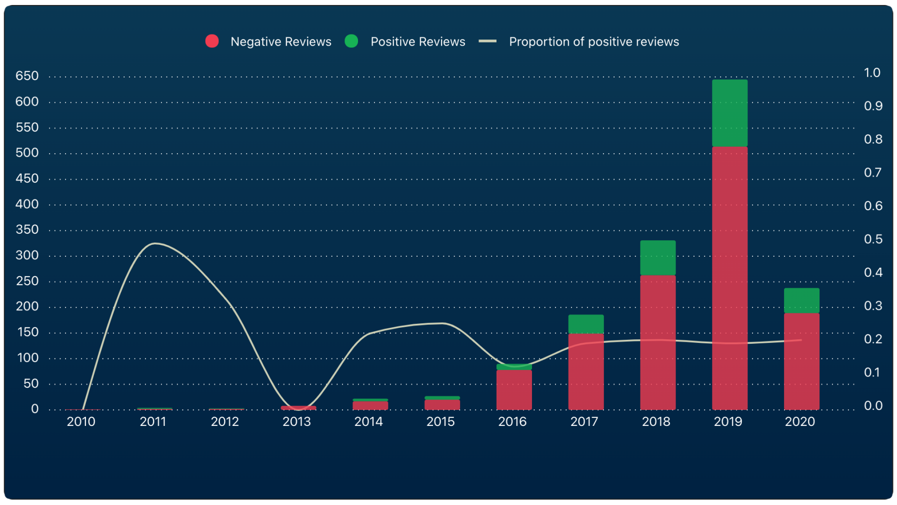


Figure 6.6: Positive and negative reviews over time

Figure 6.7 shows our next graph. This graph is a bar plot over the positive and negative mentions for the ten most mentioned aspects, all based on the extracted and classified opinion pairs from the **ABOM** model on this dataset. We can see that most of these top aspects have an overall negative sentiment.

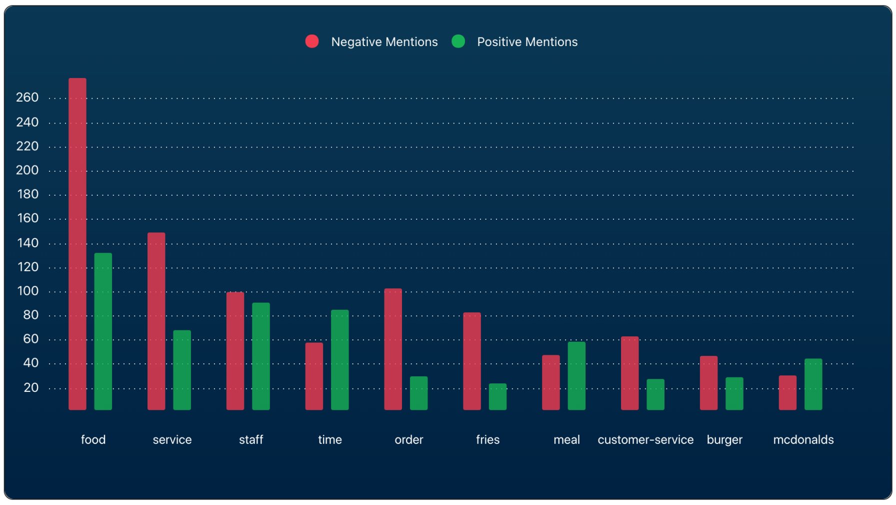


Figure 6.7: Positive and negative mentions for the ten most mentioned aspects

For comparison we have also included the same graph as in Figure 6.7, but based on the predictions from the **ABSA-E2E** model. This is shown in Figure 6.8. Note that this supervised model has not been trained on this dataset, but on review sentences for various different restaurants.

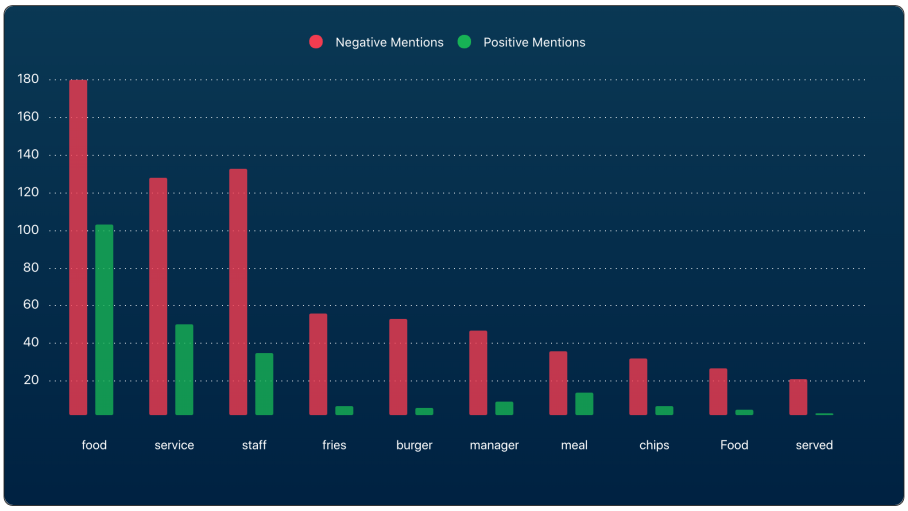
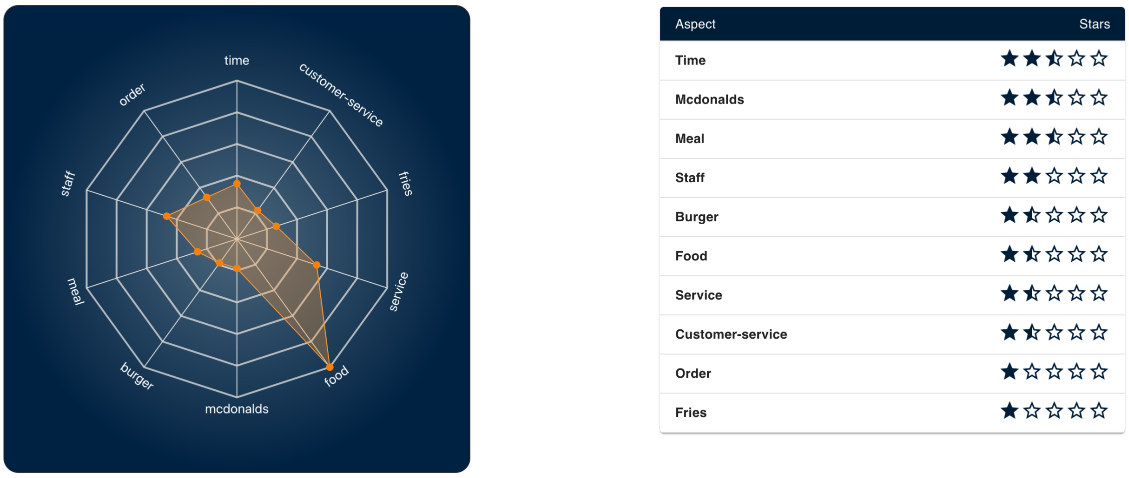


Figure 6.8: Positive and negative mentions for the ten most mentioned aspects. This graph is based on the predictions from the **ABSA-E2E** model

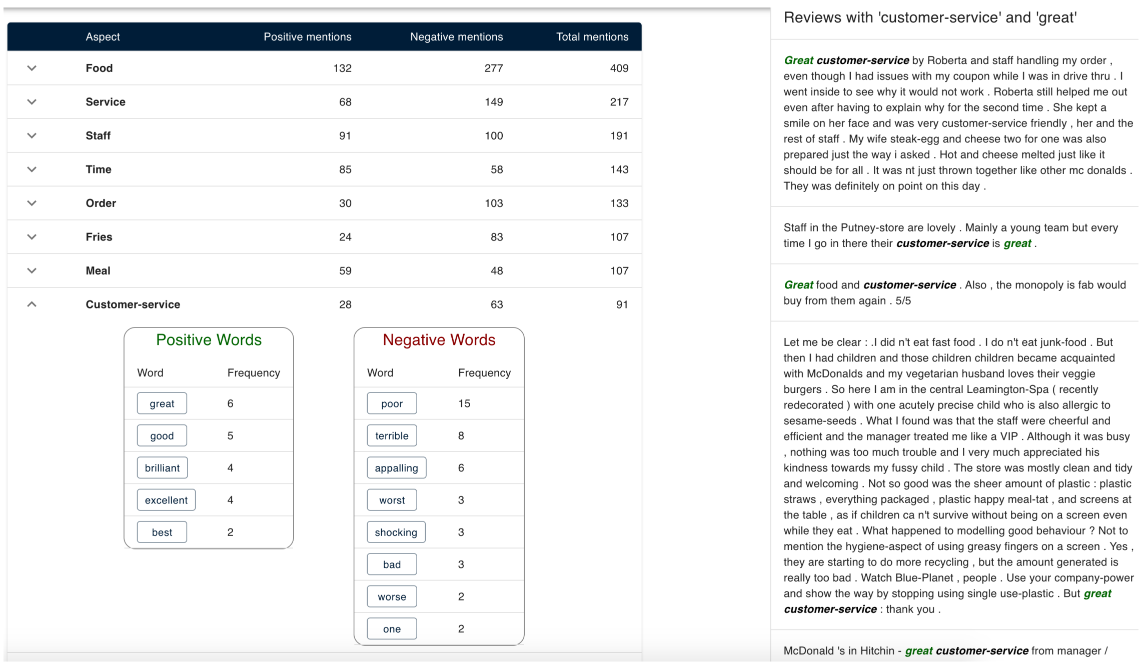
The next screenshot from the user application is displayed in Figure 6.9 below. The radar graph to the left is over the total number of aspect mentions extracted for these ten most mentioned aspects. The table to the right show us star ratings for these aspects, summarizing the distribution of positive and negative mentions for the aspect words.



Aspect	Stars
Time	★★⯪☆☆
Mcdonalds	★★⯪☆☆
Meal	★★⯪☆☆
Staff	★★☆☆☆
Burger	★⯪☆☆☆
Food	★⯪☆☆☆
Service	★⯪☆☆☆
Customer-service	★⯪☆☆☆
Order	★☆☆☆☆
Fries	★☆☆☆☆

Figure 6.9: Radar graph over number of mentions as well as star ratings for the ten most common aspects

Finally, we have a table where you can explore the different opinion words that were used to describe these top ten aspect words. As we can see in Figure 6.10, it is also possible to display the review texts where the chosen opinion pair, in this case (Customer-service, great), has been extracted by the **ABOM** model.



Aspect	Positive mentions	Negative mentions	Total mentions
Food	132	277	409
Service	68	149	217
Staff	91	100	191
Time	85	58	143
Order	30	103	133
Fries	24	83	107
Meal	59	48	107
Customer-service	28	63	91

Positive Words

Word	Frequency
great	6
good	5
brilliant	4
excellent	4
best	2

Negative Words

Word	Frequency
poor	15
terrible	8
appalling	6
worst	3
shocking	3
bad	3
worse	2
one	2

Reviews with 'customer-service' and 'great'

Great customer-service by Roberta and staff handling my order , even though I had issues with my coupon while I was in drive thru . I went inside to see why it would not work . Roberta still helped me out even after having to explain why for the second time . She kept a smile on her face and was very customer-service friendly , her and the rest of staff . My wife steak-egg and cheese two for one was also prepared just the way i asked . Hot and cheese melted just like it should be for all . It was nt just thrown together like other mc donalds . They was definitely on point on this day .

Staff in the Putney-store are lovely . Mainly a young team but every time I go in there their ***customer-service*** is ***great*** .

Great food and ***customer-service*** . Also , the monopoly is fab would buy from them again . 5/5

Let me be clear : .I did n't eat fast food . I do n't eat junk-food . But then I had children and those children children became acquainted with McDonalds and my vegetarian husband loves their veggie burgers . So here I am in the central Leamington-Spa (recently redecorated) with one acutely precise child who is also allergic to sesame-seeds . What I found was that the staff were cheerful and efficient and the manager treated me like a VIP . Although it was busy , nothing was too much trouble and I very much appreciated his kindness towards my fussy child . The store was mostly clean and tidy and welcoming . Not so good was the sheer amount of plastic : plastic straws , everything packaged , plastic happy meal-tat , and screens at the table , as if children ca n't survive without being on a screen even while they eat . What happened to modelling good behaviour ? Not to mention the hygiene-aspect of using greasy fingers on a screen . Yes , they are starting to do more recycling , but the amount generated is really too bad . Watch Blue-Planet , people . Use your company-power and show the way by stopping using single use-plastic . But ***great customer-service*** : thank you .

McDonald 's in Hitchin - ***great customer-service*** from manager /

Figure 6.10: Table from which you can explore the reviews with the chosen opinion pair, in this case (Customer-service, great)

Qliro

At last, we look at our Swedish demo dataset **Qliro**. These results are based on the predictions from the text classification model **BERT-Swe** (trained on **Trustpilot**) and the aspect-based opinion mining model **ABOM-Swe**.

First, in Figure 6.11, we have the stacked bar plot over the number of negative respective positive reviews, predicted by **BERT-Swe**, and grouped by year. The trend line of the proportion of positive and negative reviews, shows a steady increase of overall positive sentiment (except for 2019 to 2020 but only a couple of months worth of reviews from 2020 are present in the data).

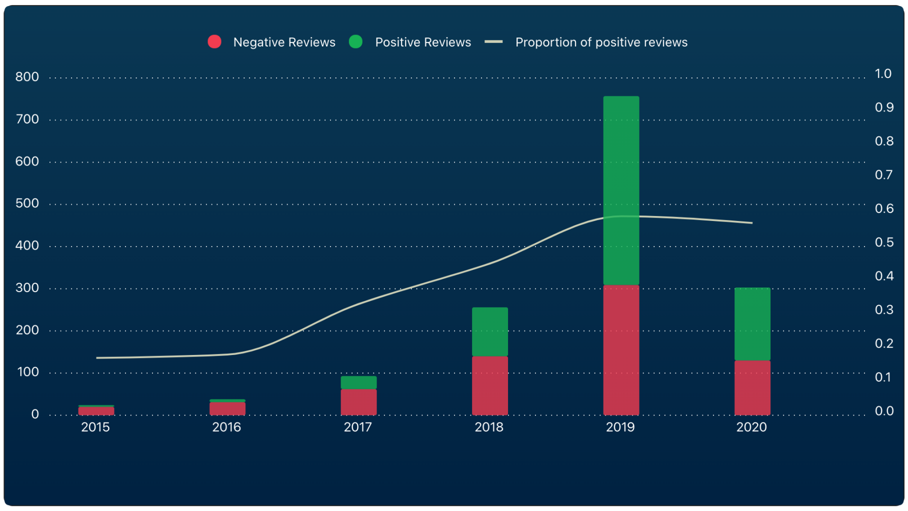


Figure 6.11: Positive and negative reviews over time

Next, Figure 6.12 displays the positive and negative mentions for the ten most commonly mentioned aspect words, based on the opinion pairs extracted and classified by the **ABOM-Swe** model. Here most aspects have an overall positive sentiment.

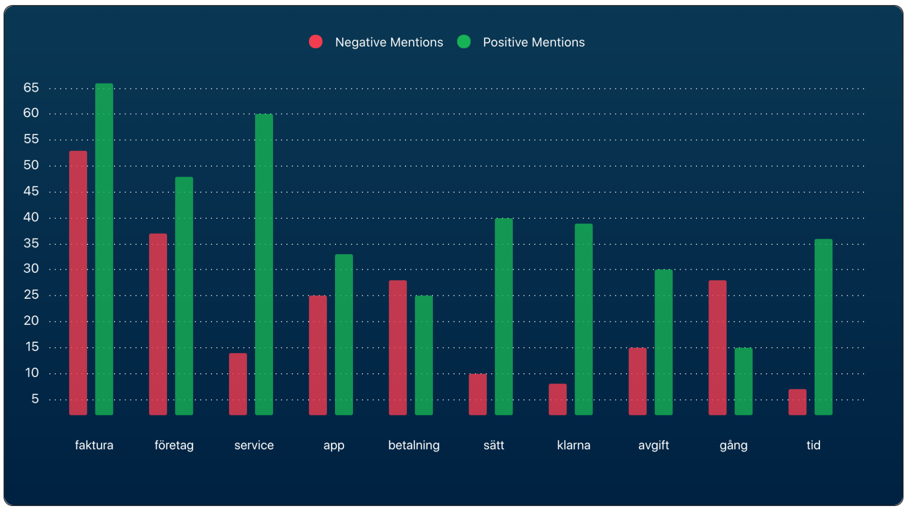
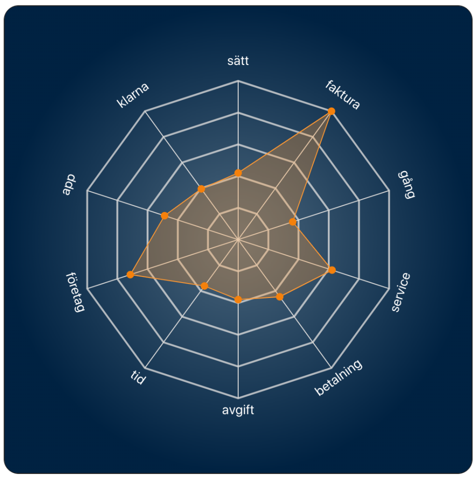
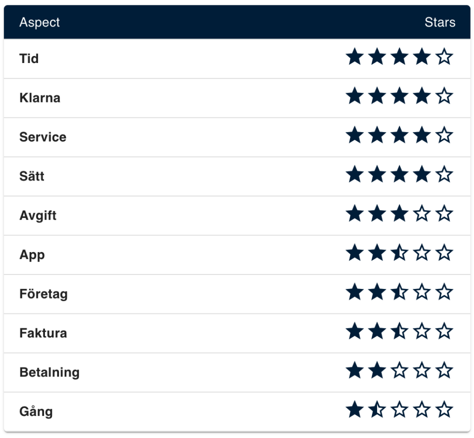


Figure 6.12: Positive and negative mentions for the ten most mentioned aspects

We also have the radar graph over number of mentions and star ratings for the ten most mentioned aspects. This can be seen to the left respectively right in Figure 6.13 below.

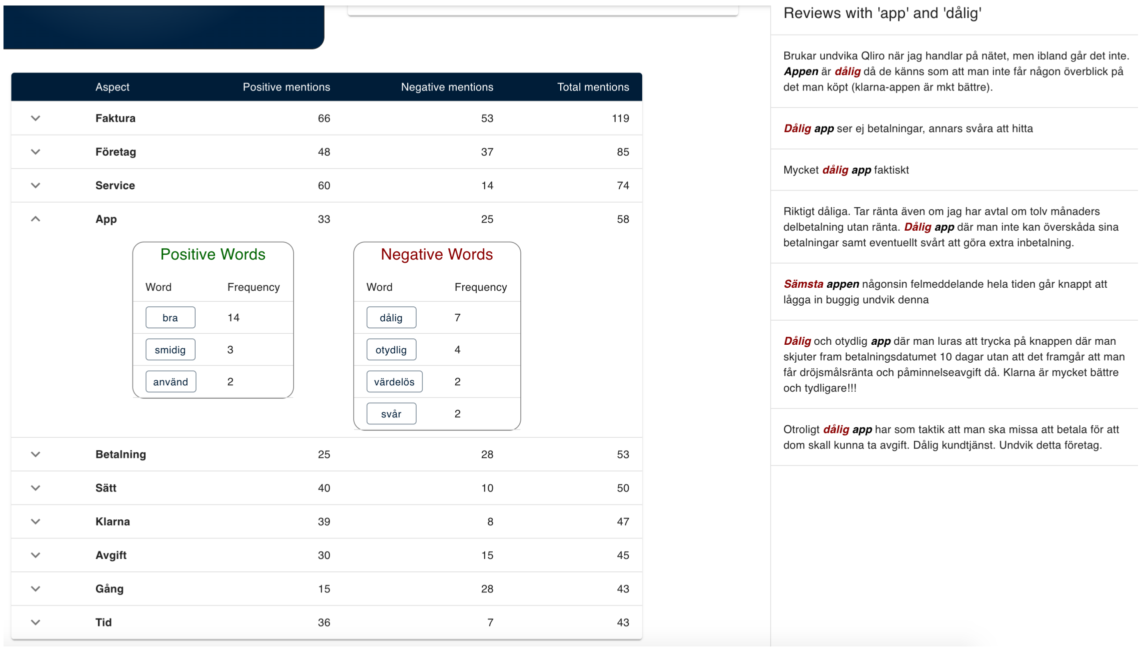




Aspect	Stars
Tid	★★★★☆
Klarna	★★★★☆
Service	★★★★☆
Sätt	★★★★☆
Avgift	★★★☆☆
App	★★⯪☆☆
Företag	★★⯪☆☆
Faktura	★★⯪☆☆
Betalning	★★☆☆☆
Gång	★⯪☆☆☆

Figure 6.13: Radar graph over number of mentions as well as star ratings for the ten most common aspects

At last, Figure 6.14 show us a screenshot from the user application, displaying the table where you can explore the different opinion words used to describe the ten most common aspects. By selecting an aspect and a corresponding opinion word, we can also see all reviews where this opinion pair has been extracted, in this case (App, dålig).



Aspect	Positive mentions	Negative mentions	Total mentions
Faktura	66	53	119
Företag	48	37	85
Service	60	14	74
App	33	25	58
Betalning	25	28	53
Sätt	40	10	50
Klarna	39	8	47
Avgift	30	15	45
Gång	15	28	43
Tid	36	7	43

Positive Words

Word	Frequency
bra	14
smidig	3
använd	2

Negative Words

Word	Frequency
dålig	7
otydlig	4
värdelös	2
svår	2

Reviews with 'app' and 'dålig'

Brukar undvika Qliro när jag handlar på nätet, men ibland går det inte. ***Appen*** är ***dålig*** då de känns som att man inte får någon överblick på det man köpt (klarna-appen är mkt bättre).

Dålig ***app*** ser ej betalningar, annars svåra att hitta

Mycket ***dålig*** ***app*** faktiskt

Riktigt dåliga. Tar ränta även om jag har avtal om tolv månaders delbetalning utan ränta. ***Dålig*** ***app*** där man inte kan överskåda sina betalningar samt eventuellt svårt att göra extra inbetalning.

Sämsta ***appen*** någonsin felmeddelande hela tiden går knappt att lågga in buggig undvik denna

Dålig och otydlig ***app*** där man luras att trycka på knappen där man skjuter fram betalningsdatumet 10 dagar utan att det framgår att man får dröjsmålsränta och påminnelseavgift då. Klarna är mycket bättre och tydligare!!!

Otroligt ***dålig*** ***app*** har som taktik att man ska missa att betala för att dom skall kunna ta avgift. Dålig kundtjänst. Undvik detta företag.

Figure 6.14: Table from which you can explore the reviews with the chosen opinion pair, in this case (App, dålig)

Chapter 7
Discussion

7.1 Text classification

From our text classification experiments, we saw that the transformer models outperformed our baseline models on all datasets. Given the complexity and sophistication of these models compared to our logistic regression and LSTM model, this was what we expected.

The only exception here was that **LSTM-Swe** beat **BERT-ML** on the Swedish dataset **Trustpilot**. This exception rather highlights the lower performance for a multilingual model on a specific language compared to a model which only handles that single language. After all, the transformer model **BERT-Swe** was the best one.

It is also interesting to see that over the different English datasets, **RoBERTa** and **XLNet** are the top performers. This aligns well with what was described in the Theory section 3.3.3 regarding the respective placements rankings of RoBERTa and XLNet on global public leaderboards (RoBERTa placed just above XLNet). We also believe that it deserves mentioning that in the paper outlining the XLNet model, they theorized that the architecture of BERT had certain issues, especially regarding pre-training and fine-tuning discrepencies and the assumption of independent masking. To our knowledge however, the implementation of RoBERTa does not differ from the original BERT model in a way that addresses any of these issues. As RoBERTa achieves betters results than XLNet, perhaps these issues are not that significant after all.

Another observation from these experiments is that the best results were achieved on the **Trustpilot** dataset. The results on the Yelp datasets in turn are better than the ones for the Amazon datasets. This is consistent for all the different models. Having a look at the data analysis presented in the Dataset section, we believe that the following four factors impact the difficulty of the classification task

- review lenghts
- class imbalance

- amount of specific domain words
- proportion of 1-star and 5-star reviews

The reviews lengths are typically much shorter for the Swedish **Trustpilot** dataset compared to the English datasets. The shortness of the text itself might be easier for the models to deal with, but the longer texts often contain relatively less relevant content and more anecdotal information that may distract the models from the useful information in the texts.

The class imbalance also seems to be an important factor, with the **Trustpilot** dataset as the most balanced and **Amazon1** and **Amazon2** as the most imbalanced. The models may become biased towards the largest class, which is evident when comparing the results on the Amazon and Yelp datasets. While the accuracy scores are very similar, the F1-scores are clearly better on the Yelp datasets. We can see this for all models.

Another thing that possibly makes the Amazon reviews more difficult to classify, is the amount of domain specific words. The Amazon reviews are product reviews while the reviews from Yelp and Trustpilot are for different companies. The amount of product specific words are likely higher than the amount of company specific words, resulting in a lower degree of homogeneity in the whole dataset vocabulary.

The proportion of 1-star and 5-star reviews, i.e. how much of the reviews that are either very negative or very positive, is likely a factor that impacts the difficulty of the classification task. The **Trustpilot** dataset has a clear majority of these kind of reviews, which makes it easier for the text classification models.

7.2 Aspect extraction

From the aspect extraction results, we observed several things worth mentioning. Somewhat surprising, we saw that our models without spell and grammar correction was slightly better than the ones with it. This was the case for both the English and Swedish models. The number of errors in the sentences in the **SemEvalABSA-En** and **SemEvalABSA-Swe** are few, so we expected the improvement of using spell and grammar correction to be small, but not that it would get worse.

A thing that we did expect, was that the English models would be better than the Swedish ones. The main reason for this, is likely that the English dependency parser model is better. It is trained on more data and a part of the `spacy` library while we had to train the Swedish dependency parser model ourselves. Perhaps it is also easier to write general grammatical rules defining opinion pairs in English compared to Swedish.

Another thing that we note when looking at the results, is that our English models clearly have a better recall score than precision score, and for the Swedish models we observe the opposite. This means that our English aspect-based opinion mining models seem to extract more non-annotated aspects than they fail to extract annotated ones. For our Swedish models, the opposite appear to be the case. Why this is the case is hard to determine, and might differ on another dataset.

Finally, it is worth mentioning that the annotation guidelines used in the **SemEvalABSA-En** dataset, are in some cases not optimal for our models. As an example of this, we can consider the following sentence and annotations (in XML code):

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<sentence id="en_BlueRibbonSushi_478218518:0">
    <text>Good Sushi, High Price </text>
    <Opinions>
        <Opinion target="Sushi" category="FOOD#QUALITY"
            polarity="positive" from="5" to="10"/>
        <Opinion target="Sushi" category="FOOD#PRICES"
            polarity="negative" from="5" to="10"/>
    </Opinions>
</sentence>
```

Our models would find the opinion pairs (Sushi, good) and (Price, high). Here however, *Price* is not annotated as an aspect but instead incorporated in the category and we have two different entries for *Sushi*. This type of "annotation problem" does not occur in the **SemEvalABSA-Swe** dataset.

7.3 Aspect sentiment classification

Our models seem to handle the sentiment classification of the extracted opinion pairs rather well. The accuracy scores look very good, while the F1-scores are lower. The number of negative mentions are not that many, so every false negative or false positive punishes the F1-score for the negative class quite heavily, thus reducing the final macro F1-score.

7.4 Demo cases

When presenting the demo case datasets **McDonalds** and **Qliro** in the Dataset section, we commented on some expectations on the demo results given the data analysis we did. Based on the distribution of star ratings, we expected that our model would predict many of the **McDonalds** reviews as negative, and a fairly even split between positives and negatives for the **Qliro** dataset. With around 80-85 percent predicted negatives for the **McDonalds** dataset, and a roughly even sentiment split of the **Qliro** reviews, we got the predictions we expected.

In our data analysis, we also listed the words (excluding stopwords) with the most occurrences throughout all reviews in the dataset. From these word lists, we got several aspect words likely to be among the ten most mentioned aspects extracted by our models. For the **McDonalds** dataset we had *food*, *order*, *McDonald's*, *staff*, *service* and *time*, while we for the **Qliro** dataset had *Qliro*, *faktura* and *Klarna*. Apart from *Qliro*, all of these words were actually among the ten most commonly mentioned aspect words extracted by our models.

The majority of the extracted top **McDonalds** aspects have an overall negative sentiment, which match the many negative reviews. In the **Qliro** case, we see the opposite; most extracted top aspects have an overall positive sentiment.

7.5 Unsupervised vs supervised approaches

We end this Discussion section by addressing the advantages and disadvantages of our unsupervised approach to aspect-based opinion mining compared to a supervised approach like the **ABSA-E2E** model or the systems from the SemEval-2016 competition.

The big disadvantage of the unsupervised approach compared to (at least one of the better working) supervised approaches, is of course worse performance. Looking at the results of the aspect extraction task on the English **SemEvalABSA-En** dataset (Table 6.6), we clearly see the potential performance increase of a supervised model. Better performance would increase the credibility of all the different graphs in the user application.

However, it is not possible to evalutate our models on the demo datasets with regards to aspect extraction and aspect sentiment classification since this data is unlabeled. By utilizing multiple models and cross-referencing the different results, we can get an intuitive feel for how well the models are performing on new, unseen data. This is mainly the reason why we tried the **ABSA-E2E** model.

While one of the participating supervised systems in the SemEval-2016 competition could be used to create all the graphs in the user interface, it is not possible to make the last exploratory table. Our unsupervised dependency parser approach enables the extraction of the opinion word used to describe the aspect as well, and not just the aspect and the sentiment polarity. This is a great advantage of our models. The opinion words corresponding to extracted aspects give us more information and possibly actionable insights.

The other big advantage of our unsupervised approach is that we do not need any training data. This enables our aspect-based opinion mining models to work on data from any domain without first having to annotate training data. This is extra beneficial in this case where almost no publicly available aspect-based opinion mining data is available. This is especially true for datasets in Swedish.

Chapter 8
Conclusions

Throughout this work, we have investigated multiple modern approaches for sentiment text classification and aspect-based opinion mining in both English and Swedish. Upon comparing these different models, we have found that models based on RoBERTa and XLNet achieved the best performance on classification tasks in the case of English data. The Swedish BERT model performed best on the Swedish text classification dataset, with an almost perfect F1-score.

Regarding aspect-based opinion mining, we have found that several supervised models achieve better performance on the SemEval datasets compared to our unsupervised dependency parser approaches. This is expected, but as discussed in Section 7.5, our models gain other advantages such as extracted opinion words. There are also very few annotated datasets for aspect-based opinion mining, and we have yet to find one in Swedish. As such, unsupervised models are essential for creating a general framework that works regardless of the domain without the need of an extensive annotation process.

For our demo-datasets (McDonald's and Qliro), we have seen that a supervised model based on the SemEval restaurant dataset works fairly well on customers reviews for McDonald's. So it is possible to use transfer learning between datasets. However, this would likely not be the case when the domains are further apart, e.g. restaurants and automotive industry.

Some improvements we considered was that it might have been useful to fine-tune the hyperparameters used for the transformer models to see if it would yield any improvements on our datasets (changing batch sizes, maximum sequence length, learning rate etc.). This would have been a very time-consuming task for the multitude of configuration options with multiple models, datasets and languages. Based on our background reading, we decided this would yield little real value to the project as a whole and we were also limited by the amount of GPU hours allocated in Google Colab.

It would also be beneficial to move the backend to a platform with better hardware and support for distributed tasks. When we were running this on a Google Colab GPU, it took approximately 5 minutes to generate the analysis data (aspect-based opinion pairs) of 1,500 reviews (with an average length of 71 words). We originally wanted our application to able to generate the analysis data in real time by simply searching for a company by its name or organizational number, but this is not possible with the current setup without waiting, relatively, a considerable amount of time.

An improvement which would make the analysis of specific companies easier would be more fine-grained views for some of the plots. It would be useful if we could choose one of the aspects in the bar plot over positive and negative mentions (e.g. Figure 6.7), and investigate how this distribution of mentions has evolved over time, similar to the first stacked bar plot of positive and negative reviews (e.g. Figure 6.6).

Another useful feature would be the option to choose which aspect words we want the model to look for. We can then immediately get an intuitive understanding of what the customers are thinking on a specific subject that we are interested in. This would be very useful if a company is trying to understand the public general sentiment towards, for example, how the company is dealing with environmental issues.

Another thing we considered was to present the customer sentiment of multiple companies at the same time. This would simplify the comparison between a specific company's competitors and it would highlight the areas to capitalize on.

With regards to future work, we have found that new models are constantly being developed. What was available and practical to use when we started working on this project is no longer state-of-the-art. For example, it would be interesting to see if Google's text-to-text transfer transformer model, T5, can further improve the results from the models we have used. Hopefully, we will soon get to see releases of Swedish RoBERTa and XLNet models as well, and thus possibly be able to improve on the Swedish BERT model.

Hopefully, a public Swedish annotated dataset for aspect-based opinion mining will become available in the near future. It would be interesting to see how well transformer models like a Swedish BERT can be used to perform aspect-based sentiment analysis in a supervised setting.

In this work, we have restricted us to consumer reviews as the only data source. It would however be interesting to perform a similar analysis, but instead use other types of data sources such as Twitter posts, Facebook posts and employee reviews.

Almost no publicly available annotated data for aspect-based sentiment analysis exist today. An idea we find interesting is therefore to automatically annotate data for this purpose using extracted opinion pairs from a dependency parsing approach like our own, and then use this data in a supervised setting with transformer models like BERT. This would allow us to generalize the supervised approach without the need to manually annotate data for each domain. However, some manual overview would be necessary. Especially when considering that the opinion pairs that the dependency parse tree models do not find would greatly influence the extent to which the supervised models can be used.

A future study that we would find interesting is to use our sentiment-based company analysis as an instrument for financial analysis. Investigate if we can find some correlation between the sentiment of a company, or the sentiment of specific parts of the company, and the stock price.

References

Akan Esen, B. (2018). How to manage machine learning/deep learning project. https://medium.com/@akanesen/how-to-manage-machine-learning-deep-learning-project-51f9b0fe164f.

Alammar, J. (2018). The illustrated transformer. http://jalammar.github.io/illustrated-transformer/.

Anoop, V. and Asharaf, S. (2018). Aspect-oriented sentiment analysis: A topic modeling-powered approach. In *Journal of Intelligent Systems, Volume 29, Issue 1*, pages 1166–1178.

Britz, D. (2016). Attention and memory in deep learning and nlp. http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/.

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

Giannakopoulos, A., Musat, C., Hossmann, A., and Baeriswyl, M. (2017). Unsupervised aspect term extraction with b-lstm & crf using automatically labelled datasets. *arXiv preprint arXiv:1709.05094v1*.

Gong, X.-R., Jin, J., and Zhang, T. (2019). Sentiment analysis using autoregressive language modeling and broad learning system. In *IEEE International Conference on Bioinformatics and Biomedicine*.

IBM (2020). Crisp-dm help overview. https://www.ibm.com/support/knowledgecenter/SS3RA7_15.0.0/com.ibm.spss.crispdm.help/crisp_overview.htm.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882v2*.

Kostadinov, S. (2019). Understanding encoder-decoder sequence to sequence model. https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346/.

Li, X., Bing, L., Zhang, W., and Lam, W. (2019). Exploiting BERT for end-to-end aspect-based sentiment analysis. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 34–41.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

McAuley, J. (2020). Amazon product data. `http://jmcauley.ucsd.edu/data/amazon/`.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Science/Engineering/Math, England, first edition.

Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., Al-Smadi, M., Al-Ayyoub, M., Zhao, Y., Bing, Q., De Clercq, O., Hoste, V., Apidianaki, M., Tannier, X., Loukachevitch, N., Kotelnikov, E., Bel, N., Jiménez-Zafra, S. M., and Eryiğit, G. (2016). Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of SemEval-2016*, pages 19–30.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144.

Russell, S. J. and Norvig, P. (2016). *Artificial Intelligence: A Modern Approach*. Pearson Education, England, third edition.

SemEval-2014 Task 4 (2020). Data and tools. `http://alt.qcri.org/semeval2014/task4/index.php?id=data-and-tools`.

SemEval-2016 Task 5 (2020). Data and tools. `http://alt.qcri.org/semeval2016/task5/index.php?id=data-and-tools`.

Singh, A. (2017). Anomaly detection for temporal data using long short-term memory. Master's thesis, KTH Information and Communication Technology.

Sun, C., Huang, L., and Qiu, X. (2019). Utilizing bert for aspect-based sentiment analysis via constructing auxiliary sentence. *arXiv preprint arXiv:1903.09588v1*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1556–1566.

Trustpilot (2020a). Trustpilot home page (english). `https://www.trustpilot.com/`.

Trustpilot (2020b). Trustpilot home page (swedish). `https://se.trustpilot.com/`.

Universal Dependencies (2020a). Universal dependency relations. `https://universaldependencies.org/u/dep/`.

Universal Dependencies (2020b). Universal pos tags. `https://universaldependencies.org/u/pos/`.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Appendices

Appendix A
Composition of the Truspilot dataset

Table A.1: Composition of the **Truspilot** dataset

Category	Company	Negatives	Positives	Total
Elektronik och teknik	Comviq	259	38	297
	Dustin Home	36	36	72
	Komplett	97	82	179
	Teknikproffset	159	679	838
	Viasat	461	29	490
Energi och ström	Eon	88	4	92
	Göta Energi	125	259	384
	Sala-Heby Energi	36	375	411
Evenemang och underhållning	Biljett24	21	32	53
	Biljettnu	49	20	69
	CDON	26	274	300
	C More	46	20	66
	Discshop	26	31	57
	Homeenter	73	146	219
	Jollyroom	73	227	300

Table A.2: Composition of the **Truspilot** dataset

Category	Company	Negatives	Positives	Total
Fordon och transport	Hedin Bil	186	209	395
	Park It Smart	20	351	371
	Parkster	141	23	164
	SMS Park	183	19	202
Företagstjänster	Qliro	662	694	1356
	Staples	178	377	555
Hem och trädgård	Bagaren och Kocken	72	67	139
	Bauhaus	168	45	213
	Furniturebox	169	331	500
	HTH	60	163	223
	Jula	69	37	106
	Luxi	113	104	217
	Sängjätten	101	399	500
Hälsa och medicinskt	Apotea	146	315	461
	Beconfident	34	39	73
	Shopping4net	62	87	149
	Specsavers	75	14	89
	Synsam	99	441	540
Mat, dryck och tobak	Coop	110	26	136
	Fodi	41	542	583
	Linas matkasse	89	73	162
	Mathem	106	69	175
	Matsmart	49	28	77
	Rawfoodshop	17	99	116
Media och förlag	Adlibris	370	83	453
	Bokus	291	37	328
	Bookoutlet	102	11	113
	Campusbokhandeln	67	433	500
	Ginza	26	54	80
	Tidningskungen	76	424	500

Table A.3: Composition of the **Truspilot** dataset

Category	Company	Negatives	Positives	Total
Pengar och försäkring	Collector	85	126	211
	Ferratum	62	396	458
	If	119	89	208
	Lendo	52	634	686
	Moderna Försäkringar	95	88	183
	Safello	29	51	80
Resor och semester	Apollo	63	269	332
	Expedia	65	36	101
	Momondo	53	268	321
	Resecentrum	203	385	588
	SAS	167	47	214
	Travelstore	117	186	303
	Tui	74	170	244
Restauranger och barer	Delitea	36	493	529
	Etnomat	34	264	298
	Mat	23	16	39
	Max	78	17	95
	McDonalds	72	16	88
Shopping och mode	8848 Altitude	17	567	584
	Björn Borg	11	77	88
	Bubbleroom	176	43	219
	H&M	37	14	51
	Nelly	195	805	1000
	Zalando	425	170	595
Skönhet och välbefinnande	HARMONIQ	40	84	124
	Lyko	167	333	500
	Nordicfeel	104	228	332
	Vivamondo	56	21	77

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2020-06-15

EXAMENSARBETE Sentiment Analysis and Aspect Extraction for Business Intelligence
STUDENTER Fredrik Olsson, Gustav Handmark
HANDLEDARE Pierre Nugues (LTH)
EXAMINATOR Jacek Malec (LTH)

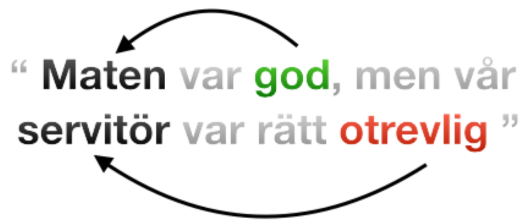
Sentimentanalys av kundåsikter för strategiskt beslutsfattande

POPULÄRVETENSKAPLIG SAMMANFATTNING **Fredrik Olsson, Gustav Handmark**

Mängden tillgänglig data i form av användarrecensioner, inlägg och kommentarer på sociala medier har ökat kraftigt på senaste tiden. För många företag är det en överväldigande uppgift att manuellt granska denna data för att få en uppfattning om vad kunder tycker. Hur kan vi automatisera denna process?

Att undersöka kunddata och kunna förutse trender är viktiga instrument vid affärsanalys och kan leda till identifiering av strategiska affärsmöjligheter. I detta arbete har vi utforskat modern språkteknologi för bearbetning av företagsriktade kundrecensioner, och på så sätt lyckats automatisera analysen och möjliggöra ett datadrivet beslutsfattande.

I vårt arbete har vi behandlat kundrecensioner på två nivåer. Först tittar vi på recensionerna som helhet och klassificerar dessa som positiva eller negativa. Detta innebär att vi kan få ett omdöme som är frikopplat från ett eventuellt stjärnbetyg, och som istället är helt baserat på innehållet i texten.



Vi arbetar därefter med aspekt-baserad sentimentanalys för att extrahera mer detaljerad information från varje recension. Detta innebär att vi på detaljerad nivå kan få ut aspekter och tillhörande beskrivande ord. Istället för att enbart se hela recensionen som positiv eller negativ, kan vi till exempel få ut att kunden tyckte maten var god, men var missnöjd med sin servitör som var rätt otrevlig.

Sedan aggregerar vi resultaten från tusentals användarrecensioner för ett visst företag och presenterar dessa i diverse grafer och interaktiva tabeller. Detta möjliggör enkla tolkningar av resultaten och observerade trender över tid.

Vi kan se hur antalet positiva och negativa recensioner har varierat över tid. Vi kan också se fördelningen mellan positiva och negativa omnämnanden för de tio vanligaste aspekterna och få ut ett stjärnbetyg för dessa. Slutligen kan vi också undersöka de ord som använts för att beskriva de vanligaste aspektorden.

För att öka användbarheten av vårt system har vi skapat modeller som klarar av att hantera såväl engelska som svenska texter. I framtiden hade vi gärna titta närmare på möjligheterna att använda våra nuvarande modeller för att skapa syntetiska svenska dataset, vilka kan användas för att träna upp mer kraftfulla modeller för det svenska språket. Dessutom hade det varit intressant att se om det finns något samband mellan kundernas åsikter om ett företag och dess aktiekurs.