Linköping University | Department of Computer and Information Science
Master’s thesis, 30 ECTS | Datateknik
21 | LIU-IDA/LITH-EX-A--21/037--SE

Clustering and Summarization of Chat Dialogues

– To understand a company’s customer base

Klustring och Summering av Chatt-Dialoger

David Björelind, davbj395
Oskar Hidén, oskhi827

Supervisor : Ehsan Doostmohammadi
Examiner : Marco Kuhlmann

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.
Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.
Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.
För ytterligare information om Linköping University Electronic Press se förlagets hemsida `http://www.ep.liu.se/`.

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years starting from the date of publication barring exceptional circumstances.
The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.
According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.
For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: `http://www.ep.liu.se/`.

© David Björelind, davbj395
Oskar Hidén, oskhi827

Abstract

The Customer Success department at Visma handles about 200 000 customer chats each year, the chat dialogues are stored and contain both questions and answers. In order to get an idea of what customers ask about, the Customer Success department has to read a random sample of the chat dialogues manually. This thesis develops and investigates an analysis tool for the chat data, using the approach of clustering and summarization. The approach aims to decrease the time spent and increase the quality of the analysis. Models for clustering (K-means, DBSCAN and HDBSCAN) and extractive summarization (K-means, LSA and TextRank) are compared. Each algorithm is combined with three different text representations (TFIDF, S-BERT and FastText) to create models for evaluation. These models are evaluated against a test set, created for the purpose of this thesis. Silhouette Index and Adjusted Rand Index are used to evaluate the clustering models. ROUGE measure together with a qualitative evaluation are used to evaluate the extractive summarization models. In addition to this, the best clustering model is further evaluated to understand how different data sizes impact performance. TFIDF Unigram together with HDBSCAN or K-means obtained the best results for clustering, whereas FastText together with TextRank obtained the best results for extractive summarization. This thesis applies known models on a textual domain of customer chat dialogues, something that, to our knowledge, has previously not been done in literature.

Keywords
Machine Learning, NLP, Text Representations, Clustering, Extractive summarization, TFIDF, S-BERT, FastText, K-means, DBSCAN, HDBSCAN, LSA, TextRank, Word Mover's Distance (WMD)

Topic areas
Machine learning, Data mining, Text mining, Unsupervised learning

Acknowledgments

First off, we would like to thank our supervisor Ehsan Doostmohammadi for the valuable support and assistance throughout the thesis work. Further, we would like to thank our examiner Marco Kuhlmann for the guidance and helpful insights. We would also like to thank the NLP master thesis group, consisting of thesis students in the NLP division, for input and developing questions during seminars.

A large thanks goes to our supervisor at Visma, Kejsi Gjordeni, for your engagement in our project, and for supplying us with resources and setting us up with people in the organization.

The Visma SPCS team in Växjö deserves a special thanks for your interest and time spent helping us. This thesis would not have been possible without your help.

Contents

Abstract iii

Acknowledgments iv

Contents v

List of Figures vii

List of Tables viii

List of Abbreviations ix

1 Introduction 1
- 1.1 Motivation 1
- 1.2 Aim 2
- 1.3 Research Questions 2
- 1.4 Related Work 3
  - 1.4.1 Clustering of Dialogue Acts 3
  - 1.4.2 Clustering of Short Text Streams 3
  - 1.4.3 Email Summary 4
- 1.5 Delimitations 4

2 Theory 5
- 2.1 Text Representation 5
  - 2.1.1 TFIDF 5
  - 2.1.2 FastText 6
  - 2.1.3 S-BERT 9
- 2.2 Clustering 9
  - 2.2.1 Distance Measures 10
  - 2.2.2 K-means 10
  - 2.2.3 DBSCAN 11
  - 2.2.4 HDBSCAN 12
  - 2.2.5 Cluster Evaluation Metrics 13
- 2.3 Extractive Summarization 15
  - 2.3.1 K-means (Centroid Based Approach) 16
  - 2.3.2 Latent Semantic Analysis 16
  - 2.3.3 TextRank (Graph-Based Approach) 17
  - 2.3.4 Supervised Approaches 18
  - 2.3.5 ROUGE Measure 18

3 Method 20
- 3.1 Pre-study of Lemmatizer 21
- 3.2 Data Set and Cleaning 22

3.3 Test Data Creation 22
3.4 Pre-processing 23
3.5 Text Representation 24
3.6 Clustering 25
3.6.1 Implementations 25
3.6.2 Evaluation of Clustering Models 26
3.6.3 Evaluation of Data Increase 26
3.7 Extractive Summarization 27
3.7.1 Implementations 27
3.7.2 Evaluation of Summarization Models 28
3.7.3 Qualitative Evaluation 29
3.8 Noise Reduction Experiment 29
3.9 Lab Environment 30

4 Results **31**
4.1 Test Data Creation 31
4.2 Clustering 33
4.2.1 Evaluation of Clustering Models 33
4.2.2 Noise Reduction Experiment 34
4.2.3 Evaluation of Data Increase 34
4.3 Extractive Summarization 35
4.3.1 Evaluation of Summarization Models 35
4.3.2 Noise Reduction Experiment 36
4.3.3 Qualitative Evaluation 37

5 Discussion **38**
5.1 Results 38
5.1.1 RQ1: Evaluation of Clustering Models 38
5.1.2 RQ2: Evaluation of Data Increase 40
5.1.3 RQ3: Evaluation of Summarization Models 41
5.1.4 RQ4: Qualitative Evaluation (of Summarization Models) 43
5.2 Method 43
5.2.1 Test Data Creation 44
5.2.2 Pre-processing 44
5.2.3 FastText Limitations 45
5.2.4 Parameter Turning 46
5.2.5 Distance Measures & Clustering 47
5.2.6 Dimensionality reduction 47
5.2.7 Alternative Approaches 48
5.2.8 Source Criticism 49
5.3 The Work in a Wider Context 49
5.3.1 Product Improvements 49
5.3.2 Bias 50
5.3.3 User Privacy 50

6 Conclusion **51**

Bibliography **53**

A Stop Words **57**

B Pre-study of Lemmatizer **58**

C Data Set Example **60**

List of Figures

2.1 Displaying the concept of n-grams 6
2.2 Skip-gram prediction task 7
2.3 CBOW prediciton task 7

3.1 Intended pipeline for the implemented solution 20
3.2 Schema over how the testing was performed and evaluated 21
3.3 Received data + generated chat array 22
3.4 Test data creation sheet 23
3.5 Applying all pre-processing steps on Swedish text 24
3.6 Timeline for data used for evaluation of data increase 27

4.1 Category distribution of the manually labeled test data 32
4.2 Distribution of number of words in important messages 32

5.1 Kdist plots for the representations used in the thesis 47

B.1 Result comparing UDpipe and Stanza 58
B.2 Result comparing without and with Kungbib spaCy lemmatization 59

List of Tables

2.1 Contingency table for two clusterings X and Y 15

4.1 Proportion of important messages containing few words 33
4.2 Result from clustering tests 33
4.3 Result from clustering tests using noise reduction 34
4.4 Result from data increase: TFIDF (uni) with K-means 35
4.5 Result from data increase: TFIDF (uni) with HDBSCAN 35
4.6 Result from summarization model 36
4.7 Result from noise reduction experiment 36
4.8 Result from qualitative evaluation 37

5.1 Words with missing FastText representation 45

List of Abbreviations

Adj. Rand Index	Adjusted Rand Index
BERT	Bidirectional Encoder Representations from Transformers
Bi	Bigram
BSTM	Bayesian Sentence-based Topic Models
CBOW	Continuous Bag Of Words
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DUC	Document Understanding Conference
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
Kungbib	The Royal Library of Sweden
LDA	Latent Dirichlet allocation
LSA	Latent Semantic Analysis
MLM	Masked Language Model
NLP	Natural Language Processing
NSP	Next Sentence Prediction
ROUGE	Recall-Oriented Understudy of Gisting Evaluation
S-BERT	Sentence-BERT
TFIDF	Term Frequency - Inverse Document Frequency
Uni	Unigram
WMD	Word Mover's Distance

1 Introduction

This thesis aims to develop and evaluate a tool using Natural Language Processing to analyze large quantities of customer dialogues for a Nordic software company, Visma. The purpose is to implement an approach that allows Visma to gain insight and knowledge about their customer base efficiently. This will help the company to understand the problems and questions that the customers are expressing in the dialogues.

1.1 Motivation

Today it is common for companies to communicate with their customers using a chat on their website, making it easier and more natural for both parties to connect. These chats are stored and contain valuable information about the problems that customers are facing. Because of the enormous amount of textual data, many companies have a hard time making use of this information.

Visma, further also referred to as "the company", is a Nordic computer software company that delivers systems for businesses. Sweden is one of Visma's markets with customers varying from small businesses to large corporations. Today at Visma, the customers can contact the support department with questions and problems through a chat where customers get automated responses from Visma's chatbot. When the chatbot cannot solve a customer problem, a human customer support takes over the conversation and handles the problem. The dialogue data is stored and in retrospect, the Customer Success department examines these chat conversations to gain insight and improve their products. Visma's Customer Success department handles about 200 000 customer dialogues (chat conversations) each year. Examining these dialogues is a very important but time consuming task. An approach to better understand this dialogue data would be to investigate it by clustering similar problems and questions together and then representing each cluster with a summary. This will guide the company in the search for problems since each cluster and correlated summary, hopefully, describes a specific problem. One such approach would also take all dialogues into consideration, compared to the current brute force approach where only a sample of all available chats can be read.

One challenge in this thesis is the absence of correctly labeled dialogue data from Visma. Therefore, within the frame of this thesis, a test set was created to evaluate the models for clustering and summarization. Creating the test set will be done manually with help from a company expert with knowledge about the chat data. One crucial aspect of any NLP system is the choice of the text representation. In this thesis TFIDF, S-BERT and FastText will be evaluated in combination with the algorithms for both clustering (K-means, DBSCAN and HDBSCAN) and summarization (K-means, LSA and TextRank).

NLP models typically need to train on a large amount of textual data in order to be successful, but the approaches used in this thesis are unsupervised and do not require any training. Clustering on shorter texts increases the difficulty due to the sparsity of words, which has been researched in previous work by Yin et al. [52] and Tang et al. [48]. This thesis investigates dialogue data, which are short text documents. Each dialogue is produced by two authors, a customer and the Visma customer support, which further increases complexity of the problem and can be taken into account by the model. However, previous work has proven NLP models to be successful on dialogue data as well [15].

1.2 Aim

This thesis develops an analysis tool for chat dialogues at the software company Visma. The analysis tool consists of clustering and extractive summarization. The purpose of the analysis tool is to improve work at Visma, and therefore, they were involved in both design and evaluation. The aim is to find the most suitable model for this task. For this reason, several models will be investigated in this thesis.

1.3 Research Questions

From the motivation and aim, the following research questions are addressed in this thesis:

1. Which of the clustering algorithms: K-means, DBSCAN and HDBSCAN in combination with TFIDF, S-BERT or FastText, performs best on chat data in terms of Silhouette Index and Adjusted Rand Index?

   This research question aims to answer which combination of algorithms that performs best as a clustering model. It is answered by implementing, testing and comparing the models. The models is evaluated intrinsically with Silhouette Index and extrinsically with Adjusted Rand Index using a manually created data set, annotated by a company expert at Visma.

2. Using the best clustering model from RQ1, how does changing the number of clustered chats impact performance, measured with Silhouette Index and Adjusted Rand Index?

   This research question aims to answer how clustering performance is affected by the number of chats included in the analysis. The goal is to simulate the chat analysis being performed on differently-sized time periods, not containing the same amount of chats. It is done by using the best-performing model from RQ1 and changing the number of chats included in the analysis, experimenting with a number of chats corresponding to 2 days up to 3 months. Silhouette Index is measured on all the clustered data and Adjusted Rand Index is only calculated using the labeled test data.

3. Which of the Extractive Summarization methods: K-means, Latent Semantic Analysis (LSA) and TextRank perform best on textual data, in terms of ROUGE measure?

This research question aims to answer which combination of algorithms that performs best as a summarization model. It is answered by implementing the proposed models and comparing them using ROUGE measure. In this research question, the models are evaluated using the labeled test data from the company.

4. How does the summaries produced by the two best summarization models from RQ3 perform compared to each other, evaluated qualitatively by a company expert at Visma?

   This research question aims to answer which of the two best summarization models that is preferred by a company expert at Visma, since preference is an important factor. A good summary is important in order to quickly get a picture of what problem a cluster is trying to capture. Evaluation of this research question is performed qualitatively by comparing the summaries performed by a company expert in the Customer Success team.

1.4 Related Work

This section will describe related work for similar clustering problems. The problems faced are similar the to challenges in this thesis.

1.4.1 Clustering of Dialogue Acts

Much research has been done to find the intention of an utterance as part of a conversation. This is done by classifying the emotion and dialogue act of the utterance. A dialogue act describes the function of the utterance. Some examples are greetings, questions or statements. Unsupervised classification or clustering of dialogue acts is a less studied area. It is a more complex problem since the interpretations of the clusters need to be made from the data itself.

Ezen-Can and Boyer [15] clustered dialogue acts using *query-likelihood clustering*. In their article, they perform clustering of individual utterances, while this thesis performs clustering on entire chat dialogues. Their produced clusters were evaluated using a gold standard of manually labeled dialogue acts. Accuracy was calculated by producing a matrix showing the occurrences of each dialogue act in each cluster. This is an evaluation method of the clusters using predefined labels, while not using the labels to train the model. Another approach for unsupervised classification of dialogue acts was performed by Jang et al. using density-based clustering [22]. To overcome the data sparsity problem in short texts, their article uses a word2vec representation of the words. To train word2vec and learn the embeddings they used a separate corpus including news articles, community texts and drama transcripts.

1.4.2 Clustering of Short Text Streams

Short text streams in the form of social media posts are common on the internet today. Clustering of text streams is a well-studied topic, which also faces the problem of word sparsity. Research by Yin et al. [51] proposes an unsupervised algorithm (MStream) that continuously can cluster text streams based on a Dirichlet process multinomial mixture model (DPMM model). The text streams (documents) are represented by their words and corresponding frequency. Each cluster is represented by a cluster feature vector (CF), which is a large document containing all clustered documents. Then the algorithm clusters the documents, one by one, by calculating the probability for a document belonging to a new or an existing cluster, assigning the document to the cluster with the highest probability. The probability has two characteristics. Firstly, a document has a higher probability for clusters that contain many documents (Chinese restaurant process [16]). Secondly, a document has a higher probability for a cluster if it shares many words with the cluster. The article also shows that by iterating several times over the documents (text streams) can improve the performance of the clustering. Sparsity is handled by representing the clusters as a CF instead of the commonly used mean vector. The conceptual drift problem is handled by the use of the Chinese restaurant process, since it produces a finite number of clusters without a limit on the total number of clusters.

Another approach for clustering short text streams was performed by Mustakim et al. [21] using DBSCAN on Twitter data. The results of the study show that DBSCAN can successfully be used in a text mining context in order to filter out noise. As an input to the algorithm, the article used a TFIDF representation with some pre-proccessing steps, such as stopword removal and lemmatization. Cosine distance was used as a distance measure. The optimal parameters for DBSCAN were found using a grid search and the set of parameters with the highest average Silhouette Index was picked.

1.4.3 Email Summary

Blitzer and Newman [34] propose a method to process a large email inbox, to help the reader digest several long threads in a managable way. The method is a combination of clustering the messages into topics and then summarizing the clusters. As a pre-processing step the model uses stemming and stop word removal. Afterwards, each message is represented as a vector of word probabilities, which is extracted using inverse document frequency (IDF) and probabilistic latent sentiment analysis (PLSA). The result allows some words, that are not part of the message, to still get a high probability if they appear together with the message-words in other messages. Some additional pre-processing of messages was carried out based on an initial observation, e.g. longer messages often describe the issue of concern.

Messages were clustered by single-link agglomerative clustering. Then clusters, larger than a given threshold, were used to extract summaries. Two termination criteria were used, size of cluster and the single-link distance upon combining two clusters. The report shows that cluster size was more commonly used as a termination criterion, and the authors state that cluster size is more effective as a halting criterion than any of the other distance-based thresholds that they explored. The extracted summary for a given cluster is based on the messages closest to the cluster centroid, with some criteria to also include relevant responses in the same email thread. One or several sentences can then be extracted from the chosen messages using a feature-based approach [34].

1.5 Delimitations

Visma's chat conversations contain dialogues in both Swedish and English. Ideally, the model would be able to handle both languages. However, that would involve other method aspects and will not be investigated in this thesis. The proposed model will only be able to use chat dialogues in Swedish. The work of this thesis is divided into two areas: text representation together with (1) clustering and (2) summarization, where David focused on the former and Oskar on the latter.

2 Theory

This chapter will introduce relevant concepts and background for the study. It follows the same order as the intended pipeline described in 3.1: text representations, clustering algorithms and extractive summarization algorithms. For each area, relevant metrics and evaluation methods are introduced.

2.1 Text Representation

One aspect that differentiates NLP from other machine learning tasks is the need to represent the text as numbers before applying machine learning algorithms. There are many different approaches for text representation, and it has a significant impact on the performance on a specific task.

2.1.1 TFIDF

Term Frequency - Inverse Document Frequency (TFIDF) [44] is a text representation technique that consists of two multiplied factors. Each text document has a bag-of-words count, or term frequency (TF), that counts the occurrence of each word in the document. The inverse document frequency (IDF) part takes all documents in the corpus into account and measures how much information each word provides. This is done by decreasing the weight for words that are common throughout all documents, in order to correctly emphasize words that are informative. A word that is common across the documents gets a lower IDF score, while unique words gets a higher IDF score. The two factors are calculated as follows:

$$TF(t,d) = \frac{f_{t,d}}{\sum_{i \in d} f_{i,d}} \tag{2.1}$$

where $f_{t,d}$ is the count for term t in document d.

$$IDF(t,C) = \log \frac{N}{|\{d \in C : t \in d\}|} \tag{2.2}$$

where N = $|C|$ is the total number of documents in the corpus and $|\{d \in C : t \in d\}|$ is the number of document where term t appears in the corpus C.

The term frequency and inverse document frequency are multiplied together to produce a TFIDF value for the document. The dimensionality of the TFIDF representation will be the total number of unique words in the corpus and can be enormous.

TFIDF can be calculated in multiple ways. What is earlier mentioned as "word count" can be different things. The baseline approach is to use each word by itself, which is also known as unigram (uni) or 1-gram. This can also be referred to a window that has the size of 1 for unigram. The window can be extended to cover multiple words, up to n-gram. Using two words is known as bigram (bi) or 2-gram. The point of doing this is to capture context within the text. For example, in the following sentences: "The book is exciting" and "It is time to book a ticket", the word "book" has a different meaning, and the surrounding words can help explain the context. Figure 2.1, shows how a bigram representation differentiates between the two different meanings.

	Sentence 1: "The book is exciting"	Sentence 2: "It is time to book a ticket"
Unigram:	["The", "book", "is", "exciting"]	["It", "is", "time", "to", "book", "a", "ticket"]
Bigram:	["The book", "book is", "is exciting"]	["It is", "is time", "time to", "to book", "book a", "a ticket"]

Figure 2.1: Displaying the concept of n-grams

2.1.2 FastText

Instead of only representing each text as a word frequency count, Mikolov et al. [31] propose the method word2vec. Word2vec represents words as fixed-length vectors and tries to capture a contextualized embedding of the words, putting words of similar meaning closer to each other in the vector space. For example, "good" and "great" would be represented by vectors that are closer to each other in the vector space compared to "good" and "bad". Word2vec uses a neural network to train embeddings on a corpus. One embedding is learned for each word in the corpus. Ideally, one would train the model on a corpus within the domain where the NLP task is carried out, but if the training data is general, the model can be used in many different domains. It allows for transfer learning and is especially powerful if not enough training data is available in the specific domain.

FastText was proposed by Mikolov et al. [4, 23] and is an extension of the word2vec model. One difference between word2vec and FastText is that the FastText model learns embeddings for all n-gram pieces, meaning that each word is divided into smaller parts and an embedding is learned for each sub-part. The vector representation of a word is then the sum of all the n-gram representations that make up the word. This allows FastText to calculate more similar representations for different word inflections, and it also allows the model to represent unknown words, by using representations of each word piece.

FastText is trained on a task called Skip-gram, which is the task of using the word representation to predict words in their context. The goal of Skip-gram is to predict, given a word, if another word appears in the context or not. Given a word, it predicts on a set of other words, containing both true and false words. True words are the actual words in the context, and the false words are randomly sampled from the dictionary. Skip-gram was proposed by Mikolov et al. [32]. For a visual figure of Skip-gram, see Figure 2.2.

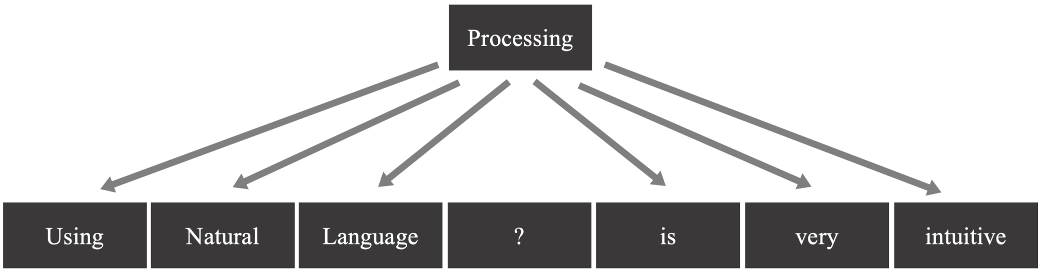


Figure 2.2: Skip-gram prediction task

Word2vec can be trained using another training task, which is called CBOW (continuous bag of words). CBOW is the task of predicting a word in the middle of a sentence from its context, see Figure 2.3. Grave et al. [18] found that training using CBOW improved performance when learning FastText representation for 157 languages.

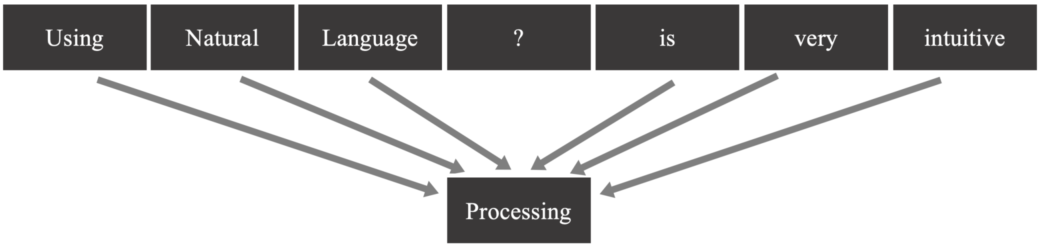


Figure 2.3: CBOW prediciton task

When using a pre-trained FastText model, the representations are stored in a hash table [32]. That is also the case for the pre-trained model used in this thesis [1].

2.1.2.1 Average Word Embedding

To obtain a document representation from FastText, a common approach is to use Average Word Embedding. This is calculated by averaging all word vectors, element-wise, which gives one vector for the entire document. Average Word Embedding is used when performing CBOW prediction in training, for example, by Grave et al. [23, 18].

2.1.2.2 Word Mover's Distance

Another way of obtaining a document representation from FastText is using Word Mover's Distance (WMD). WMD calculates distances between documents by using the difference between words of the documents, it was introduced by Kusner et al. [25]. WMD builds on Earth Mover's Distance (EMD) [43], which can be interpreted as a distance between two distributions. Two distributions can be seen as two different piles of dirt over an area. Then EMD is the "cost" of turning pile 1 into pile 2. The "cost" is the amount of dirt moved times the distance it is moved. In the same way, WMD is the cost of turning document d1 into d2 by "moving" the words. The words can be moved in total or in parts.

---

[1] https://fasttext.cc/docs/en/faqs.html

To explain WMD, some formulas are needed. WMD builds on normalized BOW, calculating a normalized word count $d1_i$, where word i appears c_i times in d1:

$$d1_i = \frac{c_i}{\sum_{j=1}^{n} c_j} \tag{2.3}$$

$c(i, j)$ is the "travel cost" or distance from word i to j.

$T_{i,j} \geqslant 0$ is how much of word i in d1 that is moved to j in d2 (a word can be moved in parts).

$\sum_j T_{i,j} = d1_i$ controls the outgoing words, words to be moved in d1.

$\sum_i T_{i,j} = d2_j$ controls the ingoing words, words to be resembled in d2.

The cost to perform all movements from d1 to d2 are then: $\sum_{i,j} T_{i,j} c(i, j)$

WMD is obtained by solving the following optimization problem:

$$\begin{aligned} \min_{T \geqslant 0} \quad & \sum_{i,j} T_{i,j} c(i, j) \\ \text{s.t.} \quad & \sum_{j=1}^{n} T_{i,j} = d1_i \qquad \forall_i \in \{1, ..., n\} \\ & \sum_{i=1}^{n} T_{i,j} = d2_j \qquad \forall_j \in \{1, ..., n\} \end{aligned} \tag{2.4}$$

The objective function aims to minimize the cost of moving while still using all words in d1 to resemble all words in d2.
Original WMD is a quite heavy problem computation-wise. An alternative is Relaxed WMD, which also is presented by Kusner et al. [25]. It relaxes the constraints for the optimization problem to find an approximate solution. First, the second constraint is removed and the following problem is solved:

$$\begin{aligned} \min_{T \geqslant 0} \quad & \sum_{i,j} T_{i,j} c(i, j) \\ \text{s.t.} \quad & \sum_{j=1}^{n} T_{i,j} = d1_i \qquad \forall_i \in \{1, ..., n\} \end{aligned} \tag{2.5}$$

Because the constraint is relaxed, a lower bound for the optimization problem is achieved. The solution for the relaxed problem is optimal when each word in d1 is moved entirely to the closest word in d2.

Second, the first constraint is removed, and the optimal solution is to move each word in d2 entirely to d1. Doing this also gives a lower bound to the original problem. The relaxed WMD is the maximum of the two lower bounds calculated. The relaxed WMD decreases computation complexity from $O(p^3 \log p)$ to $O(p^2)$, where p is the number of unique words. Reducing complexity is the main reason to use relaxed WMD, since it produces an approximate WMD as a lower bound to the true WMD. Relaxed WMD has been found to be a tight lower bound to WMD and is considered to be a distance comparable to the exact WMD [25].

WMD was evaluated on 8 data sets for document classification, from shortest Twitter data to longest BBC sports data. The Twitter data contained an average of 9 unique words per document, and the BBC sport contained an average of 117 unique words per topic. WMD obtained a relatively low error rate across all data sets and performed better than other distances measures: TFIDF, BM25, LSA (LSI) and LSD.

The cost $c(i, j)$ is calculated using Euclidean distance in the original document [25] and the wmd-relax package [2] aims to replicate the original document, which is why one can assume that this package also is implemented with Euclidian distance.

2.1.3 S-BERT

Another deep learning approach to learning word representations is BERT (Bidirectional Encoder Representations from Transformers), and it was proposed by Devlin et al. [12]. BERT makes use of transformers, which are deep learning components. BERT uses two self-supervised pre-training tasks, masked language model (MLM) and next sentence prediction (NSP). MLM randomly masks some of the input tokens, and the objective is to predict the missing token based on surrounding tokens. Here, bidirectionality is important since context can be understood for multiple directions of the sequence. NSP constructs pairs of sentences from the corpus that are either in following order or not. The task is to predict whether the first sentence of each pair is followed by the second sentence or not.

Sentence-BERT (S-BERT) was proposed by Reimers and Gurevych [38] and uses BERT as base architectures. S-BERT aims to improve on BERT to find semantically meaningful sentence embeddings. S-BERT uses embeddings from BERT for further tuning of the transformer architecture. Three objective functions or tasks are used to train the network:

- Classification Objective Function

  Two sentence embeddings from BERT, u and v, are concatenated with the difference $|u - v|$ multiplied with their respective weights and run through a softmax function.

- Regression Objective Function

  The cosine similarity is calculated between the sentence embeddings u and v.

- Triplet Objective Function

  A triplet loss function is used to make distances between an anchor sentence a and a positive sentence p smaller and the distance between a and a negative sentence n larger.

The three tasks above are used in a 3-way softmax-classifier objective function to tune the network to produce meaningful embeddings. S-BERT is also faster for inference than normal BERT. For regression tasks, instead of calculating all pair combinations in a data set to find the pair with the highest similarity, S-BERT uses its fixed size embeddings which can use be used to measure cosine similarity. S-BERT negates the need to run two sentences at the same time to measure similarity.

2.2 Clustering

Clustering is the task of grouping a limited set of objects into clusters, such that objects in a cluster are similar to each other and dissimilar to objects in other clusters. The goal is to assign data points close in distance (or have a high similarity) to the same cluster. Clustering algorithms have several different approaches, which makes them suitable for different kinds of data. Density-based, hierarchical, grid-based and partitioning are the most common approaches for clustering algorithms. Text clustering is a widely investigated area, Aggarwal et al. has done previously compared different clustering algorithms on textual data [1].

---

[2] https://github.com/src-d/wmd-relax

2.2.1 Distance Measures

In clustering, there are different ways to measure the distance between data points. The goal of measuring distances is to quantify the similarity or dissimilarity between two objects. In most cases, the Euclidean distance is used. However, depending on what the data is modeling, other distance measures might be more appropriate.

1. Euclidean Distance

   Euclidean distance between two n-dimensional data points is defined as:

   $$d_{Euclidean}(p,q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{2.6}$$

   The Euclidean distance measure can be any value larger than or equal to 0, $d_{Euclidean}(p,q) \geqslant 0$. A visual representation of the reachable area around a data point using a Euclidean distance with some radius measure is a sphere in a 3-dimensional space.

2. Cosine Distance

   The Cosine distance between two n-dimensional data points is defined as:

   $$d_{cosine}(p,q) = 1 - cos(\theta) = 1 - \frac{\sum_{i=1}^{n} p_i * q_i}{\sqrt{\sum_{i=1}^{n} p_i^2 * \sum_{i=1}^{n} q_i^2}} \tag{2.7}$$

   The definition of the cosine similarity is the normalized dot product between two vectors, making the similarity between two equal vectors to `1`. The cosine distance is $1 - (cosine\ similarity)$, making the distance between two equal vectors to be `0`. The cosine distance measure can be any value between 0 and 2, $0 < d_{cosine}(p,q) < 2$. A visual representation of the reachable area around a data point using cosine distance with some radius measure would be a cone in a 3-dimensional space. According to Singhal, cosine similarity is a better similarity measure to use for information retrieval tasks, such as text mining, compared to Euclidean similarity [46]. The main reason is that text representations can be seen as vectors. For example, when comparing two TFIDF representations of two texts where the first text is a copy of the second one but repeated twice. The vectors would have the same angle but different magnitudes in the vector space. A cosine distance measure would judge the texts to be very similar, whereas a Euclidean measure would judge the text to be very different. When investigating related work, the use of cosine distance for NLP applications is common practice, see Section 1.4.

2.2.2 K-means

The original K-means algorithm was proposed by MacQueen [28]. It was one of the first clustering algorithms and it is still widely used today. Typically, clustering algorithms require the number of clusters, *k*, as an input parameter and K-means is no exception to this. K-means aims to assign data points to clusters such that variance is minimized based on a distance measure. Because of this, K-means tries to create spherical clusters and cannot create clusters if the data has complex geometrical shapes.

The algorithm performs the following steps to produce clusters:

- *k* data points are randomly chosen as initial cluster centers, called centroids

- Each data point is assigned to the centroid that is the closest, forming clusters
- Until there is no change in clusters, repeat:
  1. Centroids are updated to be the mean of the data points within each cluster.
  2. Data points are assigned to the closest centroid

The parameter k can be intuitively chosen if domain knowledge exists, for example, clustering flower data where the number of different flower types is known. If the desired amount of clusters is unknown, a grid search for different k-values can be utilized where manual inspection can find the most meaningful clusters. Another method of finding the optimal value for k is using the elbow method [49]. It consists of running K-means for different values of k and plotting k-values against the mean of squared distances for each run. The variance will decrease when k is increasing, but at a certain point, there should be an "elbow" in the curve which is the point of diminishing returns for getting a decreased variation. The elbow method is a heuristic that gives a k-value which produces "good" clusters, according to the authors.

2.2.3 DBSCAN

DBSCAN was initially proposed by Ester et al. [14] and introduces the concept of density within a data set. To define density in this context, some concepts need to be introduced.

ϵ-neighborhood
Using any distance measure, a neighborhood spans around each point. The radius of the neighborhood, ϵ, functions as a threshold value and is a parameter for the DBSCAN algorithm. Points within ϵ distance from another point is parts of its ϵ-neighborhood.

Directly density-reachable
A point p is directly density-reachable from a point q if q's ϵ-neighborhood contains q and at least *minPts*. *minPts* is an input to the algorithm in the form of an integer.

Density-reachable
A point p is density reachable, with regards to ϵ and *minPts*, from q if there is a chain of points connecting p and q that are directly density-reachable to each other. This concept is an extension of directly density-reachable and is an asymmetric relationship. Meaning that if p is density-reachable from q, then q is not necessarily density-reachable from p.

Core points
A point p is classified as a core point if it has at least *minPts* within its ϵ-neighborhood. This introduces the second input parameter for DBSCAN, *minPts*. Connecting to earlier definitions, core points within a cluster are all density-reachable to each other.

Border point
A point p is classified as a border point if it is density-reachable from a core point q, but p itself does not have at least *minPts* within its ϵ-neighborhood. Border points are typically located on the outer edge of a cluster.

Density-connected
Two points, p and q, are density-connected with regards to ϵ and *minPts* if there exists a point w such that p and q are density-reachable from w. It is an asymmetric relationship, meaning that if p is density-connected to q, then q is density-connected top. The concept of clusters can now be defined.

Cluster
All points that belong to a cluster are density-connected. If a point p is density-reachable from a point q that belongs to a cluster, p is also part of that cluster. Consequently, clusters consist of two types of points only, core points and border points.

Noise
Points are classified as noise (or outliers) if they are not considered core points or border points.

The algorithm starts from an arbitrary point p from the database. All density-reachable points are retrieved. If p is a core point, a cluster is formed, and if it is not, the algorithm will try another point in the database. All points in the database are systematically checked whether they belong to a cluster, should create a new cluster, or are considered as noise. Clusters may merge along the course of the algorithm if two clusters contain points that are density-connected to each other. In this way, DBSCAN can cluster data according to its density and leave some points out of the clusters to be considered noise.

A critical aspect of DBSCAN is to pick parameters correctly. *minPts* can be intuitively chosen based on domain knowledge. ϵ, on the other hand, is more tricky to choose since its interpretation is not as intuitive. For some applications, it can be intuitive, for example, using GPS location data. Adjusting ϵ and manually inspecting the produced clusters is one way of finding meaningful clusters. One more structured method of choosing ϵ is using grid search for a wide variety of values and picking the value that meets some decision criteria, such as the number of clusters or proportion of outliers in the produced clusters. The authors of DBSCAN suggest a heuristic for picking suitable parameters[14]: for each data point, the distance to the k-th closest neighbor is calculated. This list is then sorted in ascending order and plotted. In theory, there should be a "valley" in the graph, which is where the ϵ-value would be obtained. All points to the left of the threshold would be considered outliers. This heuristic is useful since it does not require any domain knowledge.

DBSCAN* is an extension of DBSCAN that makes some changes to the definitions. It was proposed by the same researchers that proposed DBSCAN [6]. One main difference is that the concept of border points is no longer present. It eliminates the problems that might arise when border points are assigned to different clusters depending on what points are visited first in the algorithm, making the algorithm deterministic which meaning that it produces the exact same clusters for every run.

2.2.4 HDBSCAN

The authors of DBSCAN* propose an improved version of DBSCAN* called Hierarchical DBSCAN* (Hierarchical Density-Based Spatial Clustering of Applications with Noise) [6]. The goal was to eliminate the need for the user to set the ϵ-parameter, since it is a problematic task. Instead, the user only needs to specify the *minPts*-parameter, the minimal amount of points that should be allowed in a cluster.

HDBSCAN is a hybrid algorithm that is both hierarchical and density-based. A clustering algorithm following a hierarchical approach orders the data points according to some metric or rule. Then, the data points are either aggregated one by one (Agglomerative) or divided into groups (Divisive) until a threshold is met. A feature of a hierarchical approach is that the geometric shape of the clusters does not matter.

The density-based approach aims to find parts of the data set that are dense enough, according to some metric, and divides the dense parts into clusters. The data points that are not dense enough are considered outliers. The density-based approach can also create clusters of arbitrary geometric shapes.

For HDBSCAN, a tree structure (dendrogram) is built, based on a distance measure, connecting the data points. With a notion of density, the algorithm decides where the graph should be split, creating clusters. In this way, the algorithm can separate groups of points with higher density from groups of lower density. The hierarchical approach brings a desirable property compared to a partitioning approach, like K-means, which is that knowledge of the domain is not required. This includes knowing beforehand the number of clusters that would be suitable for the data. For many clustering algorithms, knowledge of the data is crucial for setting correct input parameters. Hence, HDBSCAN is suitable when domain knowledge is limited. In order to define the HDBSCAN algorithm, some new concepts need to be introduced:

Core distance
The core distance of a point p, with regards to *minPts*, is the minimal distance required for the neighborhood to contain *minPts* number of points.

Mutual reachability distance
The mutual reachability distance is defined as the minimal distance ϵ for which two points p and q are density reachable. If one of the points is located in a less dense area, the mutual reachability distance is set to the largest core distance of p or q.

The algorithm starts by calculating the core distance for each point, with regards to *minPts*, using some distance measure. A minimal spanning tree is calculated using the mutual reachability distance. The edges of the graph are then sorted in increasing order. The edges are removed one after one (starting with the largest) while checking if the two resulting clusters contain at least *minPts* points. If they do, they are considered true clusters. If one does not have at least *minPts*, the cluster will stay intact, saving information at what core distance the split would have occurred. After this procedure, a much smaller tree is left that can be presented as a dendrogram, where the leaf nodes are individual data points and the top corresponds to all points in the data set. It contains information about at what distance the data is being split into clusters. To determine where in the dendrogram the clusters should split and where they should stay intact, a number for the cluster stability is calculated. For points p in a cluster C, stability is defined as:

$$stability(C) = \sum_{p \in C} (\lambda_p - \lambda_{birth}), \tag{2.8}$$

$$\text{where } \lambda_p = \frac{1}{distance_p}$$

λ_{birth} is the lambda value where the cluster becomes its own cluster. It is desirable to create clusters that have a long "life time". In this context it means that the cluster is persistent for a large range of distances in the dendrogram. Walking up the tree, if the childrens' stability is higher than that of their parent, the cluster is split. Otherwise, the clusters are merged and kept together when continuing up the tree. At a certain distance level, if the cluster contains less than *minPts*, the data points are considered as noise. In this way, HDBSCAN can capture clusters of varying density, effectively capturing clusters of different values of ϵ in a single run of the algorithm, compared to DBSCAN where this would not be possible.

2.2.5 Cluster Evaluation Metrics

There are two main approaches to evaluate clusters, intrinsic and extrinsic evaluation methods. The foundation of intrinsic evaluation of cluster algorithms are the concepts of separation and cohesion. The measures are unsupervised, meaning that the data is not labeled. Instead of class labels, the intrinsic measures use distances within and between clusters to evaluate them. The goal is to provide a value for the performance of clustering algorithms. The idea was initially proposed by Bouldin and Davies [9]. A general definition was provided, but specific details were not given. It made the usage very adaptable and has been extended multiple times. The concept of separation generally refers to how distinguishable different clusters are from each other, whereas cohesion refers to similarities within each cluster. In this setting, high separation and high cohesion mean that clusters are far apart and that data points within each cluster are close together.

Extrinsic evaluation metrics are based on data that is not used for the clustering itself. The most commonly used data are predefined labels. These metrics can also be used to solve other machine learning problems, such as eliminating outliers using clustering before training a classifier. The goal of extrinsic evaluation metrics is to investigate if the clusters are meaningful for the task at hand.

2.2.5.1 Silhouette Index

An intrinsic evaluation measure called silhouette Index and was proposed by Rousseeuw [42] and extends the concept of separation and cohesion. The measure is based on creating silhouettes for each cluster based on their tightness. It combines comparing values within the produced clusters with each other and the separation of the clusters as a whole. The Silhouette Index for each point in the data set is defined as:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{2.9}$$

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i,j) \tag{2.10}$$

$$b(i) = \min_k \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j) \tag{2.11}$$

$s(i)$ = Silhouette Index for point i
$a(i)$ = Average dissimilarity of i to objects of its own cluster C
$b(i)$ = Minimum dissimilarity of i to cluster k, where k is all other cluster than that of i
$d(i,j)$ = Distance or dissimilarity measure between i and j

To calculate a value for Silhouette Index on the entire data set, the average of each point's Silhouette Index is calculated. A Silhouette Index for each cluster can also be calculated by averaging the values for points that belong to a specific cluster. A high Silhouette Index indicates that the clusters are dense and far apart and therefore desirable. When calculating Silhouette Index, any distance measure can be used. One negative property of Silhouette Index is that it is expensive to calculate since pairwise distances between all point in the data set has to be computed. However, this could be done beforehand, reducing calculation time.

2.2.5.2 Adjusted Rand Index

The extrinsic evaluation measure Rand Index was proposed by William M. Rand [37]. It measures the similarity between two data clusterings on a scale from 0 to 1, where 1 means that the clusterings are the same and 0 means that the clusterings are entirely different. Rand Index compares all pairs in the data set and their clusterings to see if they are grouped in the same or different categories.

In this thesis, the produced clusters from the clustering models will be measured against a manually labeled data set. A manually labeled data set can in of itself be seen as a clustering. If the clustering model captures the exact labeling of the data set, a perfect score for Rand Index is achieved. Rand Index for a data set D and its two clusterings, X and Y, is calculated as follows:

$$R = \frac{A+B}{A+B+C+D} \tag{2.12}$$

A = Pairs of data points in D that are in the **same** group in X and in the **same** group in Y
B = Pairs of data points in D that are in **different** groups in X and in **different** groups in Y
C = Pairs of data points in D that are in the **same** group in X and in **different** groups in Y
D = Pairs of data points in D that are in **different** groups in X and in the **same** group in Y

An extension of the Rand Index called Adjusted Rand Index was proposed by Hubert et al. [20]. One problem with Rand Index is that bad clusterings still can produce fairly good values due to chance. Hence, Adjusted Rand Index corrects the chance of randomly producing a good clustering. Two assumptions are that the number of clusters is the same and that each cluster has equal probability. This means that randomly clustered data should result in an Adjusted Rand Index value of 0. It also means that the Adjusted Rand Index can produce negative values if the clustering is worse than randomly picked, see equation 2.13. In order to define Adjusted Rand Index, a contingency table for two clusterings, X and Y on a data set D, needs to be introduced, see Table 2.1.

Table 2.1: Contingency table for two clusterings X and Y

	Y_1	Y_2	$\cdots$	Y_k	sum
X_1	n_{11}	n_{12}	$\cdots$	n_{1k}	a_1
X_2	n_{21}	n_{22}	$\cdots$	n_{2k}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	$\cdots$	n_{rk}	a_r
sum	b_1	b_2	$\cdots$	b_k	

X and Y is divided into $\{X_1, X_2, ..., X_r\}$ and $\{Y_1, Y_2, ..., Y_k\}$ respectively, where each element is a set of data points in a cluster. n_{ij} is then the number of common data points between cluster i of X and cluster j of Y, $|X_i \cap Y_j|$. a_i tell us how many data points cluster i of X has in common with all clusters in Y. r is the number of clusters in X and k is the number of clusters in Y. Adjusted Rand Index is defined as:

$$ARI = \frac{\sum_{i,j \in C}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] \frac{1}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}} \tag{2.13}$$

where n_{ij}, a_i and b_i are used from Table 2.1

2.3 Extractive Summarization

Extractive summarization is the task of creating a summary by extracting sentences from the input. The input being one text or a set of documents to summarize, called single document summarization and multiple document summarization. All sentences in the summary do exist in the input. There are also other summary creation methods that try to build new sentences by creating a summary containing completely new sentences that do not necessarily exist in the input. This is called abstractive summarization. In this thesis, extractive summarization is performed by extracting entire chat messages instead of sentences. That means that below extractive summarization is described as it is performed in literature (extracting sentences) and later in method it is described as it is used in this thesis (extracting chat messages).

2.3.1 K-means (Centroid Based Approach)

K-means summarization is a centroid based approach. Centroid based approaches typically try to find and capture different topics in different clusters and then extract the summary based on these, hoping to gain full coverage of the input. The approach can vary a bit but typically follows the structure of:

1. Split the document into sentences
2. Use a textual representation technique for the sentences
3. Cluster the sentences
4. Rank the clusters. Clusters can, for example, be ranked by size
5. Extract the sentences closest to the centroid of the top-ranked clusters

K-means is a commonly used model for this approach. But it performs worse than both Bayesian and Graphical approaches on DUC 2002 and DUC 2004 data according to ROUGE measure [50].

2.3.2 Latent Semantic Analysis

Latent Semantic Analysis (LSA) was proposed by Deerwester et al. [10], and is an approach that tries to estimate an underlying latent semantic structure in the data by the use of statistical techniques. LSA is based on the notion that a document is based on a distribution of words and topics. LSA tries to find topics in the corpus, with the assumption that similar words occur in similar contexts. LSA can also be named Latent Semantic Indexing (LSI) in the literature.

Liu and Gong [17] then used LSA to find the most important sentences to extract as a summary. The idea is that a summary is created by trying to explain the topics that occur in the input. Their LSA summarization model works according to the following method:

1. The text is split into sentences.
2. A word-by-sentence matrix **A** is created, where a row corresponds to a word and a column corresponds to a sentence. An element a_{ij} holds the TFIDF score for word i in sentence j.
3. Then Singular Value Decomposition (SVD) is used to obtain:

   $$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T \tag{2.14}$$

   Where $\mathbf{V}^T$ is the right singular matrix, its interpretation is explained below.
4. The following steps are repeated until the set number of sentences is obtained. Set k=1
   a) Select row k in $\mathbf{V}^T$
   b) From the selected row, select the sentence that corresponds to the highest weight and include it in the summary

c) increment k by one.

Each diagonal element in Σ corresponds to the "topic weight" over the entire corpus (all sentences) in decreasing order. Therefore the first row corresponds to the most important topic. Each row in $\mathbf{V}^T$ corresponds to a topic, in the same order as in Σ, and each column corresponds to a sentence. The values hold the weight which represent to what extent that a sentence match with a topic. Therefore the loop will go through one topic at a time (starting with the most important topic), and extract the sentence that best describes the topic (looking at the weights for each element on that row). Through this, LSA tries to capture a large coverage of the input, picking the most descriptive sentence for the most relevant topics in the input.

2.3.3 TextRank (Graph-Based Approach)

Inspired by PageRank [35], graphical approaches represent the input as a highly connected graph, with sentences as nodes and the edges between the nodes hold the similarity between the two sentences. To measure similarity, cosine similarity together with TFIDF is commonly used in graphical methods.

One such method is TextRank [30], which can be used for both keyword and sentence extraction, and it typically has the following approach:

1. Put text units into nodes of the graph
2. Draw edges between nodes that have a relation. Relations can be directed or not, weighted or not.
3. A scoring function for the nodes is looped until convergence.
4. Text units are extracted from the nodes based on their score from the previous function.

For sentence extraction, a weighted, undirected graph is typically used. The weight of an edge can be calculated in different ways. In the original paper [30], textual overlap was used, which simply counts tokens that appear in both sentences. The texts are typically preprocessed in some way, and one can decide which words to count, for example, only nouns and verbs. The paper also adjusted for the sentence length by dividing the overlap with the length of each sentence. The following equation is used to calculate the score of a given node.

$$WS(V_i) = (1-d) + d * \sum_{V_i \in In(V_i)} \frac{w_{ij}}{\sum_{V_k \in Out(V_j)} w_{jk}} * WS(V_j) \tag{2.15}$$

$WS(V_i)$ - Score of a given node i
V_i - A given node i
$In(V_i)$ - A subset of edges that points to node i
$Out(V_i)$ - A subset of edges that points from node i
w_{ij} - The similarity measure between two sentences, $similarity(V_i, V_j)$
d - A factor between 0 and 1, used to integrate randomness. (Based on Marcov Chanis it adds randomness to the function and allows it to iterate to find the converging scores.)

TextRank can be used with different similarity functions and in 2016, Barrios at al. [3] tested TextRank with three different similarity functions: *Length of Longest Common Substring*, *Cosine Distance with TFIDF* and *BM25*. BM25 is a version of the TFIDF model that incorporates probabilities [41]. Both BM25 and TFIDF with cosine similarity improved the results compared to the original TextRank. The result was measured with ROUGE-1, ROUGE-2 and ROUGE-SU4 (explanation of ROUGE in section 2.3.5), and which of the two improvements that performed best depended on the metric. Averaging all three metrics gave an overall performance where

BM25 performed 2.92% better than the original TextRank and TFIDF with cosine similarity performed 2.54% better. Performance was evaluated on the DUC 2002 data set.

LexRank is another version of TextRank that performed best in DUC 2004, using ROUGE score. As similarity measure it uses TFIDF together with cosine similarity [13]. In LexRank, each sentence is also represented by a node, but the edges between sentences are undirected, binary, and do only exist if the similarity is above a certain threshold. LexRank can also be calculated with weighted edges. In that case, the chosen similarity measure (TFIDF with cosine similarity) is what differentiates to the original TextRank and this model would correspond to the TFIDF with cosine similarity model above, evaluated by Barrios at al. [3]. The two versions, with and without weighted edges, showed similar performance in the evaluation. The two articles support the use of TextRank with weighted undirected edges where the weight is measured using TFIDF with cosine similarity.

2.3.4 Supervised Approaches

In recent years supervised machine learning techniques has been applied to the task of summary extraction, and they show great performance. Liu [27] adds a summarizing layer on the BERT architecture and fine-tunes the model on the extractive summarization task. There are also other neural nets that achieve state of the art performance on the extractive summaries task [53, 40]. Generating an extractive summary has also been seen as a classification task [7, 33]. However, all these machine learning approaches need to learn which sentences to extract by the use of a large training set, which probably need to be problem and topic specific. Due to the lack of training data, these approaches will not be investigated in this thesis.

2.3.5 ROUGE Measure

For summarization, ROUGE measure is a measure that aims to evaluate the created summary. ROUGE is based on textual overlap and tries to evaluate the content selection. ROUGE does not measure other aspects, such as grammar or readability. This makes it a good evaluation for extractive summarization since each sentence already exists in the corpus and hopefully already have good grammar and readability. There are different types of ROUGE [26]:

ROUGE-N - Overlapping n-gram
ROUGE-L - Longest common subsequence
ROUGE-W - Weighted longest common subsequence
ROUGE-S - Skip-bigram co-occurrence statistics.

Skip-Bigram is a bigram of two words that do not have to be next to each other. Skip-Bigram allows for skipping words between, but the two words has to be in the sentence order. ROUGE-N or ROUGE-S seem reasonable to use in this thesis since it is important to capture different aspects in the summary rather than obtaining an exact phrase. In the paper [26], they found that out of the different ROUGE-N variations, ROUGE-2 performed best on single document summarization, while ROUGE-1 performed best on producing shorter summaries. Because this thesis will handle both short documents and short summaries ROUGE-1 is most promising, ROUGE-2 will be used as well.

ROUGE-N counts the number of overlapping n-grams, that is, the n following words [26].

$$\text{ROUGE-N} = \frac{\sum_{S \in (RefSummaries)} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in (RefSummaries)} \sum_{gram_n \in S} Count(gram_n)} \tag{2.16}$$

$Count_{match}(gram_n)$ is the maximum number of n-grams that co-occur in the extracted summary and the reference summary. The denominator is the total number of n-grams in the reference summaries. One or many reference summaries can be used.

ROUGE was evaluated in the original paper [26] by looking at the correlation between ROUGE and human scores. The human assessment is the ground truth, and both reference summaries and human scores were collected from DUC 2001, 2002 and 2003. They did run several experiments and using several reference summaries also increased the correlation to human summaries. They also concluded that due to the lack of data, the human judgments was not statistically stable, which could cause instability in their correlation analysis.

BLEU [36] is another measure for evaluating texts, commonly used on translation tasks but can also be used for evaluating automatically generated summaries. BLEU is not used in this thesis, but it is still presented here in theory, because some research done on BLEU is considered relevant for ROUGE as well. BLEU is similar to ROUGE-N in that it also compares a candidate summary with a set of reference summaries and counts n-grams co-occurrence between the two. BLEU differ from ROUGE because an n-gram in BLEU does not have to be in the sentence order, nor does the containing words have to be following each other. A n-gram in BLEU is just a subset of n words from the text. BLEU is a precision measure while ROUGE is a recall measure, BLEUs denominator is the total n-grams in the candidate set while ROUGE is the total n-grams in the reference summary. Recall is more important for this thesis since it is desirable to capture as many aspects from the important messages as possible, rather than make sure that each component of the candidate summary is important.

$$\text{BLEU} = \frac{\sum_{C \in (CandSummaries)} \sum_{gram_n \in C} Count_{clip}(gram_n)}{\sum_{C' \in (CandSummaries)} \sum_{gram_n \in C'} Count(gram'_n)} \tag{2.17}$$

ROUGE and BLEU are good measures for quick evaluation. In a structured review [39] BLEU is stated to be a good metric during development of a system, but for a final evaluation, the evidence is not as strong. One can assume that ROUGE has similar properties. ROUGE and BLEU also have a limitation due to their lack of capturing complexity. They look at textual overlap which means that the meaning of the words are not captured. Two summaries could have the same meaning and still get a bad score. In this application, however, an extractive summarization in conducted, and hopefully, different important messages are captured in the test data.

3 Method

In this chapter, the technical approach is described in the same order as the intended pipeline for the implemented solution. The chats are pre-processed, represented as numbers and clustered together. Finally the clusters are described by extractive summaries. The intended pipeline can be seen in Figure 3.1 below. The development of the models and the final pipeline is done with the intent to use it on the real world problem at Visma. Therefore a lot of discussion has taken place with several stakeholders at Visma during the development of the approach.

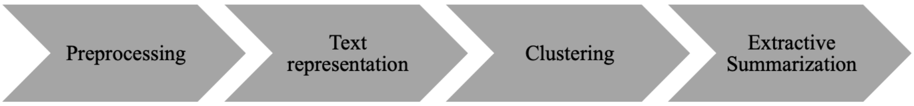


Figure 3.1: Intended pipeline for the implemented solution

The evaluation of the produced result is an important aspect in this thesis, since no gold standard exist for the data set. Therefore, within the time frame of this thesis, a test data set has been created with the help from Visma to be used for evaluation. The two parts, clustering and extractive summarization, are evaluated separately to ensure their individual quality. For summarization, evaluation with the test data is not enough to ensure quality of the final model and therefore a qualitative evaluation is also performed to decide on the final model.

Figure 3.2 below describes the developed pipelines for testing in more detail. The clustering and extractive summarization were investigated individually to ensure their individual quality. The top part of the figure explains how the clustering algorithms were tested. The bottom part explains how the summarization algorithms were tested. The figure also describes which parts of the data were used, what algorithms were used and what metrics they were evaluated with. This chapter will explain this entire process in detail.

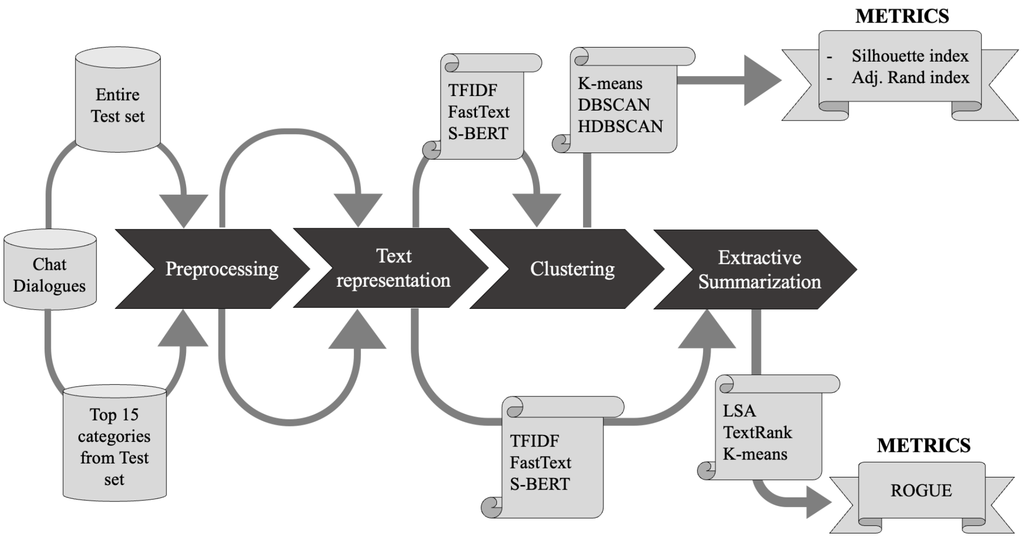


Figure 3.2: Schema over how the testing was performed and evaluated

3.1 Pre-study of Lemmatizer

A lemmatizer was used in pre-processing, and several lemmatizers for Swedish exist. Three promising ones are: Kungbib spaCy model[1], UDpipe[2] and Stanza[3]. The first two are supposed to be faster and Stanza is supposed to be more accurate since it is based on a trained neural model. To know which model to use, a small test was conducted.

The three models were compared by using Kungbib spaCy lemmatizer as a base and then comparing the top 100 words with the highest TF (Term Frequency) to see if UDpipe and Stanza could improve the result. It was done by first using Kungbib spaCy model on the data to remove stopwords, lowercase and remove numbers and then the model also performed lemmatization. Next, TF was used to obtain the top 100 words in the corpus, these words are considered to be the most important for the lemmatizer to handle. Then, UDpipe and Stanza were separately used to perform lemmatizaton on these top 100 words (note that these words already are lemmatized by Kungbib spaCy model). The result was gathered by looking at the lemmatizations that UDpipe and Stanza changed. The changed words were compared to see if the new lemmatization form was an improvement over the Kungbib spaCy lemmatization or not. The result showed that Stanza performed best. Stanza changed 39% of the top 100 words with one incorrect change, while UDpipe changed 35% with four incorrect changes. The one incorrect change in Stanza was also obtained in UDpipe. The result can be read in appendix B in Figure B.1.

In the test above the Kungbib spaCy lemmatizer was used before the Stanza lemmatizer, and in the final pre-processing model only Stanza is used. To know what happens when the Kungbib spaCy lemmatizer is removed another test was performed. The test compared the lemmatization with and without Kungbib spaCy lemmatizer. The result showed that the final lemmatization was the same with and without Kungbib spaCy lemmatizer. The result can be found in appendix B in Figure B.2

---

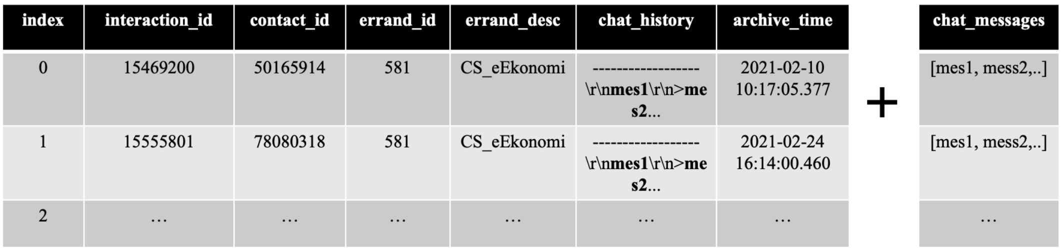
[1] https://github.com/Kungbib/swedish-spacy

[2] https://github.com/ufal/udpipe

[3] https://stanfordnlp.github.io/stanza/lemma.html

3.2 Data Set and Cleaning

The data was collected from a database using an SQL-query. It was acquired as a csv-file, from which the chats were extracted. The data set is not publicly available. The extracted data contained seven columns, see the left table in Figure 3.3, and each row correspon§ed to one chat. The column *chat_history* contained the entire chat stored as a string, with each messages separated with "$\backslash r \backslash n$". From the chat-string, all messages were extracted and stored, message by message in an array. This generated chat array was added in column eight, see Figure 3.3.



index	interaction_id	contact_id	errand_id	errand_desc	chat_history	archive_time
0	15469200	50165914	581	CS_eEkonomi	----------------- \r\n**mes1**\r\n>**me** **s2**...	2021-02-10 10:17:05.377
1	15555801	78080318	581	CS_eEkonomi	----------------- \r\n**mes1**\r\n>**me** **s2**...	2021-02-24 16:14:00.460
2	…	…	…	…	…	…

+

chat_messages
[mes1, mess2,..]
[mes1, mess2,..]
…

Figure 3.3: Received data + generated chat array

The generated column *chat_messages* separated *chat_history* into an array of messages for each chat. Each chat consists of a number of messages that are either from a client or the customer support, see Appendix C for an example of a chat. In the example, the conversation starts with the chat bot, and continues on with a human customer support. In some cases, one part sent two or more messages back to back. Because of this, it is non-trivial to automatically separate messages sent from a client or a customer support. There are messages at the start and end of each chat that are generic, which does not add any value to the chats. These are for example chat-queue updates and survey links. These generic messages are removed as a cleaning step.

The customer has the option to start the conversation with a chatbot. The bot tries to answer the question. When the bot cannot answer, a human customer support is needed. The conversation is then continued in the same chat and the new messages are stored together with those from the chatbot. The answers from the chatbot in these chats still added context about the problem and therefore these chatbot messages were not removed.

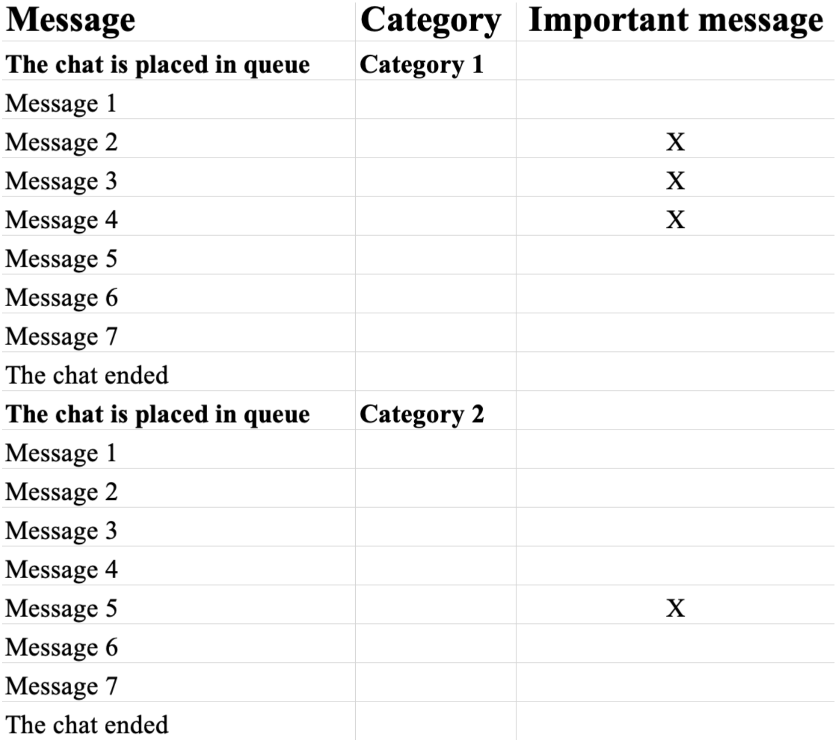
The data set contains chats that are incomplete. It happens that a customer starts a chat but does not provide any question. In that case the customer support ends the chat, but the chat is still stored. They do not contain any valuable information and is essentially noise. Since these chats will occur organically, they are kept in the data set in this thesis.

3.3 Test Data Creation

When creating the test data it was investigated how Visma works with the chats today. At Visma there is one person that every month performs at least one chat analysis. This person is considered a company expert, since the person has the best knowledge about these chats. When performing an chat analysis, a random sample of 400 chats from last month's chats are extracted and each chat is labeled with a category. A category is a problem area of interest for Visma. The labeling is done by the company expert, and then the statistics over the most common categories are reported and discussed. To create the test data this thesis aims to follow the current process at the company for chat analysis, unfortunately previously performed chat analyses cannot be used because the labeling of chats is not persistently stored.

Therefore, to create the test data, the company expert at Visma performed four new batches of chat analysis, in the usual way, but also storing the labeling of each chat. To obtain this data in an extractable format, the chat analysis was performed in a spreadsheet as in Figure 3.4. Each batch contained 400 chats and was randomly extracted from a given month, two from February 2021 and two from March 2021. Two extracted samples from one month were done without replacement. This gave a total of 1600 labeled chats used for evaluating the clustering models.

In addition to the labeling, an extra step was added to the chat analysis, namely the marking of important messages. While reading the chats, the expert marked some important messages within each chat that described the category especially well. One chat could contain one or multiple important messages. These were then used to evaluate the summarization models, by using the important messages to create reference summaries.



Message	Category	Important message
The chat is placed in queue	**Category 1**	
Message 1		
Message 2		X
Message 3		X
Message 4		X
Message 5		
Message 6		
Message 7		
The chat ended		
The chat is placed in queue	**Category 2**	
Message 1		
Message 2		
Message 3		
Message 4		
Message 5		X
Message 6		
Message 7		
The chat ended		

Figure 3.4: Test data creation sheet

3.4 Pre-processing

Pre-processing steps were used before applying TFIDF and FastText. These text representations look at each individual word in the representation and the pre-processing steps aimed to reduce the number of important words to represent. No pre-processing steps were used before applying S-BERT, because this text representation (see section 2.1.3) is trained on entire sentences and therefore the model also expects entire sentences as input.

The pre-processing included the following steps:

1. Stop word removal

2. Lowercase
3. Number removal
4. Lemmatization

spaCy[4] was used for stop word removal, lowercase and number removal. Stanza[5] was used for the lemmatization step.

This thesis used spaCy version 2.3.2 together with Kungbib Swedish spaCy model[6]. During our project Kungbib also released a new Swedish spaCy model that worked with spaCy 3.0. This new model was trained on the same text corpus but was still an updated version in terms of performance, because spaCy 3.0 is a larger model allowing Kungbib to train a the model using larger transformers. Due to the larger model size this model did appear to be a lot slower to use. The authors claim by email that the new transformer models were inherently much bigger and slower than the traditional models, and due to some development issue Kungbib also had to include one extra transformer to obtain the desired output. This makes a total of two transformers which impacts the computation speed. Therefore, this thesis will use the old model, spaCy 2.3.2. Another difference is that the new Kungbib spaCy model do use transformers to represent the words instead of the pre-trained FastText vectors that was used in the old model. To speed up the spaCy model some unnecessary parts were excluded, using the following argument: `disable=[“ner”, “tager”, “parser”]`

spaCy can also be used for performing lemmatization, but the lemmatizer in spaCy is not trainable and is just a dictionary lookup. This speeds up the computation time but impacts the performance negatively, see section 3.1. Stanza however has a trainable neural model to perform lemmatization and it was used to perform the lemmatization step in this thesis. To speed up the Stanza model some unnecessary parts were excluded using the following argument: `processors = ’tokenize, lemma’`. Since Stanza was used after spaCy, Stanza’s tokenizer was disabled by using the following argument: `tokenize_pretokenized=True`.

The entire stop word list from Kungbib spaCy model was used together with a few self added words: `[ “hej”, “hejsan”, “tja”, “tjenare”, “okej”, “ok”, “vera”, “tack”]`. The full stop word list can be found in appendix A. Figure 3.5 below displays an example of using all pre-processing steps in the actual model on a Swedish text.

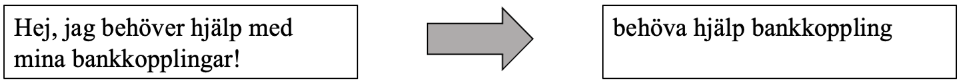

Hej, jag behöver hjälp med mina bankkopplingar!
behöva hjälp bankkoppling

Figure 3.5: Applying all pre-processing steps on Swedish text

3.5 Text Representation

In this thesis, three different text representation algorithms were implemented and tested. All of them were implemented using open-source libraries. Below follows a description of how each was implemented in this thesis.

TFIDF:
For TFIDF, the implementation from `scikit-learn` was used, for both unigram and bi-

---

[4] https://spacy.io/
[5] https://stanfordnlp.github.io/stanza/
[6] https://github.com/Kungbib/swedish-spacy

grams. The IDF scores were calculated on the entire corpus, all 1 600 chats.

S-BERT:
S-BERT was implemented using the `sentence_transformer` package together with the pre-trained model `paraphrase-xlm-r-multilingual-v1` [7], which is a multilingual model trained on millions of paraphrase sentences in multiple languages, including Swedish. A monolingual pre-trained model would have been interesting to investigate, since it would have an increased chance of performing well. However, because S-BERT is a relatively new architecture, there are no monolingual models available in Swedish. Pre-trained models optimized for semantic textual similarity would also have been interesting for this thesis, but unfortunately there are no monolingual Swedish versions or multilingual versions available.

FastText:
FastText embeddings were obtained from the Swedish Kungbib spaCy model [8]. The implemented FastText embeddings in the Kungbib spaCy model were pre-trained by Grave et al. [18]. When using average word embedding as explained in section 2.1.2.1, a representation for the entire chat was obtained by averaging all word vectors, element-wise. When using WMD, all FastText word embeddings were used with WMD to generate a distance matrix, containing the distance between all chats. This thesis used relaxed WMD implemented by the package `wmd-relax` [9]. This package was also implemented in spaCy 2.3.2 [10], therefore the WMD matrix was created using the Kungbib spaCy model.

3.6 Clustering

The following section covers how the clustering models were practically implemented and used, as well as how their parameters were tuned and how their performance were evaluated. It is also described how this thesis examines how data size affects the best clustering model.

3.6.1 Implementations

Three different clustering algorithms were implemented: DBSCAN, HDBSCAN and K-means. All of them were implemented using open-source libraries. All clustering algorithms used a cosine distance measure.

DBSCAN was implemented using the package `scikit-learn`. For HDBSCAN, Leland McInnes' implementation was used [29]. It is hosted on `scikit-learn-contrib` GitHub page, for high-quality scikit-learn compatible projects. It does not allow the use of a cosine distance measure. However, it allows for using a pre-computed distance matrix. For calculation of a distance matrix based on cosine distance, `scikit-learn`'s `cosine_distances` was used. K-means was implemented using the `Natural Language Tool Kit (NLTK)` [11]. The reason why `scikit-learn`'s implementation of K-means was not used in this thesis is because it does not allow for the use of a cosine distance measure, which the `NLTK` implementation did.

Some of the implemented algorithms requires the user to choose hyper-parameters. This was a crucial step since it has a huge effect on the performance of the model. In this thesis, a clustering model is a combination of text representation and clustering algorithm. For each

---

[7] https://www.sbert.net/docs/pretrained_models.html
[8] https://github.com/Kungbib/swedish-spacy
[9] https://github.com/src-d/wmd-relax
[10] https://spacy.io/universe/project/wmd-relax
[11] https://www.nltk.org/api/nltk.cluster.html?highlight=k%20means#module-nltk.cluster.kmeans

clustering model tested the parameters were picked accordingly:

K-means:
K-means requires the number of clusters to be produced, *k*. Initially, elbow method was used to find *k* [49]. Unfortunately, the produced elbow plot did not give a good indication of what *k* should be. Instead, the value of *k* was picked on intuition about the data. There are 40 categories in the test data set. It is known that some categories does not have very many chats assigned to them. Therefore, *k* was set to 30.

DBSCAN:
DBSCAN requires two parameters, *minPts* and ϵ. *minPts* was chosen intuitively by picking a number of chats that is a reasonable cluster size to analyze. In this case, it was set to 3. For each test, a baseline for the parameter ϵ was chosen using a heuristic developed by the authors of DBSCAN[45], that uses k-dist plots. From this baseline the parameter ϵ was adjusted in order to produce a reasonable number of clusters.

HDBSCAN:
minPts was chosen using the same reasoning as for DBSCAN, to 3. One of the advantages of this algorithm is that no more parameters are needed.

3.6.2 Evaluation of Clustering Models

In order to answer RQ1, the clusters produced were evaluated using Silhouette Index and Adjusted Rand Index. Both were implemented using the `scikit-learn` library [12] [13].

14 clustering models were created by combining all text representations with all clustering algorithms, except for the combination of WMD and K-means. This combination is not possible since K-means does not support a pre-computed distance matrix. In order to assess the results a model that is randomly generating equally weighted clusters was used as a baseline approach.

Each clustering model was evaluated using the following steps:

1. All chats in the test data were clustered using the model.
2. Silhouette Index was calculated.
3. Adjusted Rand Index was calculated using only the top 15 largest categories from the test data.
4. Outlier rate was calculated

3.6.3 Evaluation of Data Increase

In order to answer RQ2, the best model from previous results was used. In this evaluation the amount of data was increased to understand how clustering performance was affected. The goal was to simulate the chat analysis being performed on differently sized time periods. The time periods tested were between 2 days and 3 months. These time periods corresponds to chat sizes ranging from 250 to 11 000 chats being analyzed at one time. This amount of data is larger than our test data, therefore an additional set of chats were extracted from the period Apr 2020 to Jan 2020, which together with the test set made up chats from an entire year. When testing, all chats from the test data were used together with random sampling

---

[12] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

[13] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html

from the additional set. This was done to reduce any potential seasonal biases. For data sizes smaller than the test set, random samples from only the test set were used. To evaluate, all clusters were used to calculate Silhouette Index, but only data from the test data could be used to calculate Rand Index. Therefore, the clusters are evaluated on how it handles the test data, see Figure 3.6.

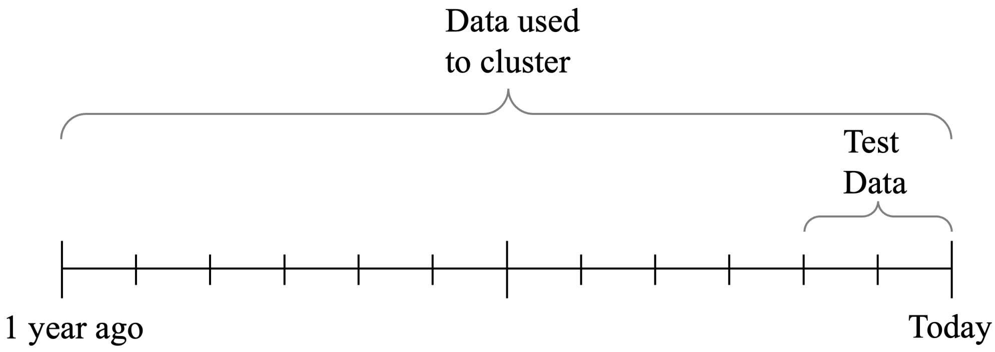


Figure 3.6: Timeline for data used for evaluation of data increase

Each size of the data was evaluated using the following steps:

1. All chats were clustered and each cluster was represented with a number, which was stored for each chat.
2. Silhouette Index was calculated using the text representations and produced clusters.
3. The top 15 largest categories from the test data were extracted from the clustering results.
4. Adjusted Rand Index was calculated using only the extracted chats together with their labels from the Test Data.
5. Outlier rate was calculated by dividing the number of non-outlier data points with the total number of data points used for clustering.

3.7 Extractive Summarization

The following section covers how the summarization models were practically implemented, used and how their performance was evaluated.

3.7.1 Implementations

The Summarizaton algorithms creates summaries by extracting messages from the chats. All implementations used open-source libraries and below is an explanation of the implementation for each of the three summarization algorithms.

3.7.1.1 K-Means Summarization

K-Means was implemented with `NLTK`'s `KMeansClusterer`. Distance was measured with cosine distance. K-Means was implemented to produce 10 clusters and the message closest to the centroid in each cluster was extracted to the summary. The centroid was obtained by the use of `.means()` and then the distance to the centroid for each sentence was calculated using cosine distance.

3.7.1.2 TextRank

TextRank was implemented by representing each message using any of the text representations. The edges between each message were undirected and weighted by cosine similarity, which was calculated using `scikit-learn`'s `cosine_similarity`, or by WMD, as described in section 3.5. Since the edges were undirected, function 2.15 is adjusted to:

$$WS(V_i) = (1-d) + d * \sum_{V_i \in In(V_i)} w_{ij} * WS(V_j) \tag{3.1}$$

Only edges going in to a node will impact the rank. The score $WS(V_i)$ is obtained by iterating a matrix multiplication according equation 3.1 until a convergence threshold was reached. The convergence threshold was set by comparing the sum of all scores after each iteration. When the difference in score between two iteration was less then $1e^{-5}$, the algorithm had converged. A max number of iteration was also set to 100. A damping factor of d = 0.85 was used since this is the number usually picked [5].

The ten messages with the highest score according to $WS(V_i)$ were extracted as a summary. To make sure that each extracted message was unique an extra condition was added, if one identical message already was extracted, the algorithm just moved on and tried to extract the next sentence with the highest score.

3.7.1.3 LSA Summarization

LSA summarization was implemented by using `scikit-learn`'s `TfidfVectorizer` to generate the **A** matrix followed by `scikit-learn`'s `TruncatedSVD` to obtain $\mathbf{V}^T$. From $\mathbf{V}^T$ ten messages, from different topics, were extracted according to theory in section 2.3.2.

3.7.1.4 Baseline Summarizations

Random sumarization was implemented by using python's `random.sample`. Largest sumarization was implemented by ranking the messages according to the largest number of characters.

3.7.1.5 ROUGE Measure

In this thesis ROUGE 1.5.5 was used, implementation made by google-research[14]. It is a native approach to replicate the original ROUGE measure introduced by Chin-Yew Lin [26], with the exception of stop word removal. The package is called `rouge-score` and it gives the user the option to use an English stemmer or not. In this thesis, `use_stemmer` is set to False because our Swedish pre-processing steps are used.

3.7.2 Evaluation of Summarization Models

In order to answer RQ3, a Quantitative evaluation was performed to compare the models for Extractive Summarization. This evaluation was used to obtain the two best models to use for a final qualitative evaluation. The Quantitative evaluation aimed to test the summarization algorithms independently from the clustering algorithm, by summarizing the top 15 categories, and evaluate them by using the important messages from the created test data set.

Every batch of 400 chats from the test data was all treated together. With all test data, the following steps were done:

---

[14] https://pypi.org/project/rouge-score/

1. The top 15 most common categories were extracted from the data and were summarized by category. One summary for each category was obtained, a total of 15 summaries.
2. From the test data, all important messages from all chats within a category were extracted and stored as a reference summary for that category, a total of 15 summaries.
3. For the model tested, both ROUGE-1 and ROUGE-2 measures were obtained for each created summary, then average ROUGE-1 and average ROUGE-2 were calculated and reported.

3.7.3 Qualitative Evaluation

In order to answer RQ4, a qualitative evaluation was performed. It uses the two best models from the quantitative evaluation and was performed as follows:

1. For all test data, the top 15 largest categories were individually summarized using the two models.
2. An expert at Visma got to know the category and was then shown two summaries, one from each model. The expert then pointed out the best summary by scoring them. The potential scores were:
    a) 1-0 when one summary can explain the category and one can not.
    b) 2-1 when one summary obviously was better than the other.
    c) 1-1 when the summaries were equally good.
    d) 0-0 when neither of the summaries did explain the category.
3. The result was collected as the total points for each model, together with the total number of 0's. The 0's explain the number of times where the model did not manage to explain the category.

3.8 Noise Reduction Experiment

During this thesis project it seemed like the longer messages tended to contain more information. In the chats, the customer often used more text to explain the problem and when answering, the customer support often used longer messages when explaining how the problem should be solved. In addition to this, chatbot messages are prewritten and, depending on the smartness of the chatbot, could potentially be noisy. These messages could appear in several categories and therefore decrease distance between messages that belong to different categories.

These thoughts resulted in an experiment, trying to reduce noise in the chat data. To run this experiment an extra step was added at the end of pre-processing. The extra step was called noise reduction and it removed short messages and messages sent by the chatbot. The short messages were removed by counting the tokens generated from pre-processing, messages containing less than five (<5) tokens were removed. Chatbot messages start with the string "Vera:" and these messages were also removed, using regular expressions. If a chat was found empty, not containing any messages after noise reduction, it was removed from the data.

3.9 Lab Environment

All code for this thesis was written in the Python programming language using Jupyter notebooks and made frequent use of packages like `pandas` and `numpy`. All packages referred to in this thesis are Python packages. The code and all tests were executed using Google Colab, which runs on Google Cloud [15]. It provides access to free computing resources and besides being convenient to run code remotely, it allows for consistency in tests and relatively powerful hardware. However, the resources cannot be guaranteed from session to session. The workload for this thesis does not require fast calculations, but it does require the system to have enough RAM to store the loaded models. Specially when loading pre-trained language models and storing distance matrices on large amounts of data.

---

[15] https://research.google.com/colaboratory

4 Results

This chapter will cover four parts. First, the distribution and some properties of the created test data will be provided. Secondly, the result from the clustering models are presented. Thirdly, the result from the effect of different data sizes used for clustering is presented. Lastly, the result comparing the summarization models is presented.

4.1 Test Data Creation

The test data consists of 1 600 chats and follow the distribution according to Figure 4.1. There were 40 categories over the time period in total. 7% of the data has been categorized as incomplete, meaning that the chat conversation did not lead to anything meaningful. The top 15 categories make up 81% of the data. There are 12 categories with less than 10 chats and they make up 4% of the data.

The test data also contains the marking of important messages. The 1 600 chats had a total of $\sim$ 35 000 messages and out of these $\sim$ 5 000 were marked as important, which correspond to 14%. The important messages are different in size and contain different number of words, from 1 up to 152 words. The distribution over number of words in the important messages are presented below in Figure 4.2.

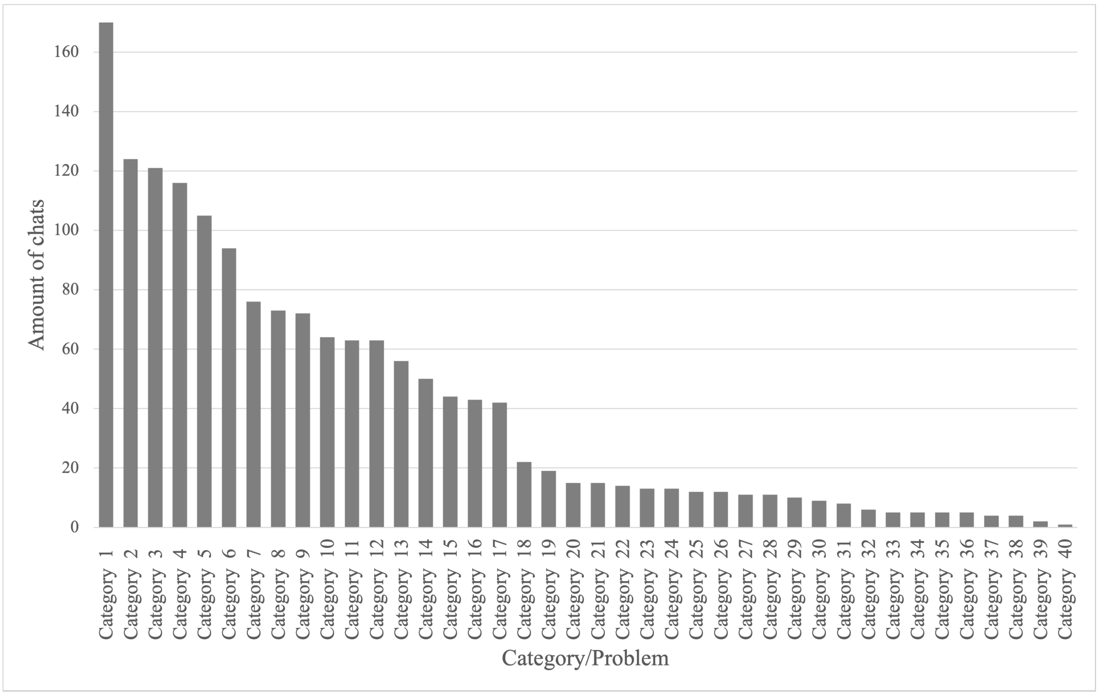


Figure 4.1: Category distribution of the manually labeled test data

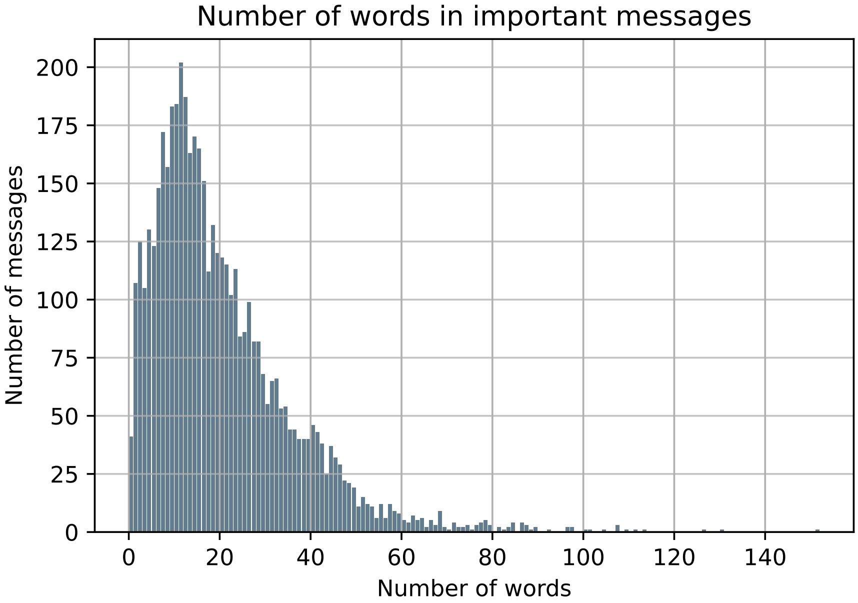


Figure 4.2: Distribution of number of words in important messages

Figure 4.2 above, shows that it is common for important messages to contain around 7–13 words. But there are also a lot of longer messages, containing over 20 words, which gives a total average of 20 words. When running the noise reduction experiment, as explained in section 3.8, messages containing less than 5 words were removed which corresponds to 10.5% of the important messages, as can be seen in Table 4.1 below.

Table 4.1: Proportion of important messages containing few words

Number of words	**% of important messages**
<2	3.0
<3	5.6
<4	7.8
<5	10.5
<6	13.0
<7	16.1
<8	19.6
<9	22.8
<10	26.6

4.2 Clustering

In this section, the results from evaluating the clustering models are presented.

4.2.1 Evaluation of Clustering Models

The results from the testing of clustering models can be seen in Table 4.2. A total of 14 models were tested. The result for WMD combined with K-means is not available since K-means does not support a pre-calculated distance matrix. The K-means algorithm does not produce any outliers, which is why all results from K-means produce 0.00% outliers. Two versions of TFIDF were tested, unigram representation and bigram representation. It is notable that both models using WMD produced invalid or unreasonable results. In the case of DBSCAN, all of the data was put into one single cluster and for HDBSCAN all the data was considered as outliers by the model. This means that no chats were put in a cluster, which also made it impossible to calculate Silhouette Index since it requires at least two groupings. The most promising results are marked in bold. TFIDF with DBSCAN, which achieved the highest Adj. Rand Index, was not chosen to be the most promising since the parameter tuning was time consuming and the results were extremely sensitive to changes of the parameters.

Table 4.2: Result from clustering tests

		Silhouette Index	**Adj. Rand Index**	**Outlier %**
TESTS				
TFIDF (uni)	**K-means**	**0.04**	**0.15**	**0.00**
	DBSCAN	0.35	0.51	89.42
	HDBSCAN	**0.38**	**0.43**	**90.21**
TFIDF (bi)	K-means	0.01	0.03	0.00
	DBSCAN	0.54	0.40	97.37
	HDBSCAN	0.01	0.00	0.06
S-BERT	K-means	0.09	0.07	0.00
	DBSCAN	0.05	0.02	88.10
	HDBSCAN	0.61	0.17	83.72
FastText	K-means	0.03	0.04	0.00
	DBSCAN	0.78	0.00	5.26
	HDBSCAN	0.78	0.00	5.20
WMD	DBSCAN	-	0.00	0.00
	HDBSCAN	-	0.00	100.00
BASELINE				
Random		$\sim$ 0.00	$\sim$ 0.00	0.00

4.2.2 Noise Reduction Experiment

Below, the results from the repeated test on noise reduced data, described in section 3.8, are presented. Just like in Table 4.2 above, WMD produces infeasible results. Notable results have been marked in bold. It can be seen that Silhouette Index increased while Adjusted Rand Index decreased. Overall, the noise reduction experiment did not lead to improvements in performance.

Table 4.3: Result from clustering tests using noise reduction

		Silhouette Index	**Adj. Rand Index**	**Outlier %**
TESTS				
TFIDF (uni)	**K-means**	**0.06**	**0.14**	**0.00**
	DBSCAN	0.43	0.31	90.74
	HDBSCAN	**0.47**	**0.33**	**89.79**
TFIDF (bi)	K-means	0.03	0.02	0.00
	DBSCAN	0.59	0.34	94.93
	HDBSCAN	0.49	0.27	93.21
S-BERT	K-means	0.09	0.13	0.00
	DBSCAN	**0.98**	**0.33**	**97.15**
	HDBSCAN	0.58	0.33	89.85
FastText	K-means	0.07	0.04	0.00
	DBSCAN	0.53	0.00	11.11
	HDBSCAN	0.60	0.12	89.92
WMD	DBSCAN	-	0.00	0.00
	HDBSCAN	-	0.00	100.00
BASELINE				
Random		~ 0.00	~ 0.00	0.00

4.2.3 Evaluation of Data Increase

When investigating the effects of data increase, two model were considered; TFIDF (uni) with HDBSCAN and TFIDF (uni) with K-means. Two models were chosen since there was no clear best model, see discussion section 5.1.1. The tests were performed on 250 chats to 11 000 chats. Below, the results from the two most promising models are presented. It can be seen that both models produce stable results at around 1 000 chats.

Model 1: TFIDF (uni) with K-means

Table 4.4: Result from data increase: TFIDF (uni) with K-means

TFIDF (uni) with K-means			
Number of chats	**Silhouette Index**	**Adj. Rand Index**	**Outlier %**
250	0.01	0.09	0.00
500	0.02	0.09	0.00
1 000	0.04	0.12	0.00
2 000	0.05	0.14	0.00
3 000	0.04	0.14	0.00
4 000	0.04	0.16	0.00
5 000	0.04	0.14	0.00
6 000	0.05	0.15	0.00
7 000	0.05	0.15	0.00
8 000	0.05	0.17	0.00
9 000	0.05	0.19	0.00
10 000	0.05	0.18	0.00
11 000	0.05	0.19	0.00

Model 2: TFIDF (uni) with HDBSCAN

Table 4.5: Result from data increase: TFIDF (uni) with HDBSCAN

TFIDF (uni) with HDBSCAN			
Number of chats	**Silhouette Index**	**Adj. Rand Index**	**Outlier %**
250	0.10	0.02	72.00
500	0.19	0.35	84.80
1 000	0.39	0.45	88.90
2 000	0.39	0.41	90.80
3 000	0.40	0.35	90.54
4 000	0.40	0.30	89.86
5 000	0.38	0.31	89.61
6 000	0.37	0.30	89.61
7 000	0.40	0.31	90.29
8 000	0.36	0.31	89.48
9 000	0.33	0.27	88.92
10 000	0.34	0.26	89.10
11 000	0.35	0.23	90.23

4.3 Extractive Summarization

In this section the result from summarization tests are presented.

4.3.1 Evaluation of Summarization Models

The results obtained from the evaluation of summarization models are presented in the table below. These results are obtained without noise reduction, explained in section 3.8.

Table 4.6: Result from summarization model

		ROUGE-1	ROUGE-2
TESTS			
TFIDF (uni)	K-means	0.029	0.009
	TextRank	0.020	0.011
	LSA	0.028	0.007
TFIDF (bi)	K-means	-	-
	TextRank	0.038	0.019
	LSA	0.038	0.015
S-BERT	K-means	0.030	0.014
	TextRank	0.015	0.006
FastText	K-means	0.056	0.022
	TextRank	**0.122**	**0.063**
	TextRank + WMD	0.122	0.063
BASELINE			
	Random	0.029	0.014
	Largest	**0.158**	**0.088**

Results were obtained for every combination except for TFIDF (bi) with K-means, due to computation time being restricted by the lab environment Colab. Overall best performance was obtained from the baseline of extracting largest messages. Second best performance was obtained by FastText together with TextRank. Many of the other algorithms were struggling and performed close to Random.

4.3.2 Noise Reduction Experiment

This experiment applied noise reduction experiment as explained in section 3.8. This reduction removed messages starting with "Vera:" and messages shorter than 5 tokens. It removed 67% of all messages and 13% of all important messages. The following table present the result obtained from evaluating the summarization models on the reduced data.

Table 4.7: Result from noise reduction experiment

		ROUGE-1	ROUGE-2
TESTS			
TFIDF (uni)	K-means	0.076	0.040
	TextRank	0.067	0.038
	LSA	0.063	0.031
TFIDF (bi)	K-means	0.060	0.028
	TextRank	0.062	0.029
	LSA	0.067	0.034
S-BERT	K-means	0.079	0.040
	TextRank	0.076	0.051
FastText	K-means	0.093	0.047
	TextRank	**0.134**	**0.071**
	TextRank + WMD	0.134	0.071
BASELINE			
	Random	0.062	0.031
	Largest	**0.155**	**0.084**

Results were obtained for every combination. By performing the noise reduction experiment the results improved, increasing ROUGE even for the Random baseline. The best performing models did not change when performing the noise reduction experiment, the baseline of extracting largest messages and FastText together with TextRank still obtained the best performance.

4.3.3 Qualitative Evaluation

From the previous sections, two summarization models did obtain the highest ROUGE score: FastText with TextRank and the baseline of extracting largest messages. These two models were used for the qualitative evaluation. Since the noise reducing experiment improved almost all scores, the reduced data was used during the qualitative evaluation. The result from the qualitative evaluation is presented below in Table 4.8.

Table 4.8: Result from qualitative evaluation

	Largest summarization	**FastText + TextRank**
Category 1	2	1
Category 2	1	1
Category 3	0	1
Category 4	0	1
Category 5	0	1
Category 6	0	1
Category 7	1	1
Category 8	0	1
Category 9	1	1
Category 10	1	1
Category 11	1	0
Category 12	1	0
Category 13	1	1
Category 14	1	0
Category 15	0	1
Total points	**10**	**12**
Total #0′s	**6**	**3**

In Table 4.8 above the scores are reported for each summary (one summary from each model for each category). The scores are given as explained in section 3.7.3, where 1′s or 2′s are given when a summary can explain the category and 0′s given when the summary fails to explain the category. In Table 4.8 the **Total points** are the sum of all scores for a summarization model and **Total #0′s** are the number of times the model failed to explain the summary. The result from the qualitative evaluation show that FastText + TextRank performed best. It obtained both higher total points and lower number of zeros.

5 Discussion

In this chapter the results are discussed and explained. The research questions are answered. Further, the method chapter and other aspects of the thesis are discussed and criticized. Potential developments and other paths of the thesis are discussed.

5.1 Results

In this section, the results from the conducted tests are presented. Here, the research questions are answered.

5.1.1 RQ1: Evaluation of Clustering Models

In order to decide which clustering model that solves the problem in the best way, the metrics used in this thesis need to be interpreted and weighted against each other. A high Silhouette Index means that the clustering model is finding differences and similarities well, while a high Adj. Rand Index means that the clusters produced by the algorithm capture the same categories as the ones in the test set. It is desirable for a good clustering model to have high values of both metrics, but a high Adjusted Rand Index is slightly more important since it measures the model's ability to capture categories within the data, which is ultimately the goal for Visma. One metric that was not anticipated to be of importance was outlier rate. It is tricky to evaluate because there are no suggestions in literature on what a good outlier rate is. However, it is not desirable to have a large outlier rate, since a large portion of the data will not be part of the clusters produced by the model.

It is clear that the most promising text representation for this task is TFIDF (uni). Compared to the results of the other text representations (Table 4.2), TFIDF proved to deliver consistently better results. TFIDF (bi) produced a representation that was about 8 times larger than TFIDF (uni), which increased calculation time dramatically. Even though a bigram representation in theory should result in a better contextual representation (explained in 2.1.1), the performance was slightly worse than using a unigram representation.

S-BERT with HDBSCAN performed best compared to the other S-BERT models. This is probably due to HDBSCAN's ability to recognize different densities within the embeddings of the data. Comparing the result from HDBSCAN with TFIDF (uni) and S-BERT, the higher Silhouette Index suggests that S-BERT captures differences and similarities better than TFIDF (uni). However, the lower Adj. Rand Index suggests that S-BERT does not capture the categories from the test set as well as TFIDF (uni). It can be concluded that the pre-trained S-BERT model used did not do a good job in finding categories within the data, at least for Swedish within this domain. Since the pre-trained S-BERT model is multilingual, it might not be specific enough. It is possible that another pre-trained S-BERT model might have produced better embeddings for this use case. It is also possible that a fine-tuning training step could improve performance and would be interesting to investigate further.

FastText produced the highest Silhouette Score using DBSCAN and HDBSCAN out of all clustering models (see Table 4.2). However, the low Adj. Rand Index of zero suggests that the ability to find categories within the data is close to random. When taking a closer look at the produced clusters from this model, there are only two clusters produced. One for Swedish chats and one for English chats, which is not very informative about actual contents. This could be explained by the use of a Swedish pre-trained model. The FastText representation from the English chats will then differ greatly from those created from Swedish chats, hence being put in different clusters. It was expected that HDBSCAN would be able to find different categories within the Swedish chats, since it can handle different densities within the data. It might still be possible that eliminating English chats from the data set might bring out the differences within the Swedish chats in a better way which might produce a more meaningful clustering. If removing English chats, the relative distances between the Swedish chats might increase, making it easier to find different aspects within those chats. It was decided not to do this, since some English chats will exist in the real use case.

The idea behind testing WMD as a distance between chats, calculated from FastText embeddings, was that it would be a better way of producing contextualized embeddings compared to FastText with average word embeddings. Especially since each text being represented is quite large. DBSCAN and HDBSCAN was not able to produce stable results using WMD. When trying different values for the parameter ϵ one of two things happened. Either all data point were put in one single cluster or all data point were considered outliers. One reason for the unsuccessful clustering might be that the distances between the data points are too small. Again, the pre-trained FastText model might be the cause of this. The results from the WMD models are presented, however the metric Silhouette Index could not be calculated since there is only one cluster in the data and distances to nearby clusters can not be calculated. In the literature, WMD are evaluated on different data sets with different document length, from small twitter messages to longer sports news articles. For all data sets WMD showed a performance better than both TFIDF and LSA 2.1.2.2. It does not explain the poor performance achieved from clustering. Rather, it further confirms that using WMD would be a good idea for the task in this thesis.

Generally, both DBSCAN and HDBSCAN produces a very high outlier rate. It is difficult to say what a reasonable value is. However, 90% is problematic since most of the data is removed from the produced clusters. No obvious patterns in the data can explain why this is. On the other hand, K-means, which does not produce any outliers, will put chats in clusters that are not meaningful. A middle ground would be ideal here, but it cannot be achieved with either of the models. The large outlier rate could be explained by that the distances between chats are too similar throughout the data for the used clustering algorithms to find meaningful differences. When comparing which categories from the test set that K-means and HDBSCAN managed to capture, they were almost the same. This means that the model had the same ability to capture categories, but widely different cluster sizes. It would be interesting to investigate whether the additional chats from K-means adds anything meaningful or whether they just added added noise.

When running the noise reduction experiment, see Table 4.3, overall higher Silhouette Index and lower Adj. Rand Index values were achieved. This means that the clustering algorithms created more dense clusters, separated from each other, but that information about the categories were lost when restricting the length. S-BERT with DBSCAN produced a high Silhouette Index and Adj. Rand Index and that looks great at first sight, but at the expense of a 97% outlier rate, which makes it an infeasible solution. Another model, S-BERT with HDBSCAN performs the best from the noise reduction experiment. The Adj. Rand Index and outlier rate is on par with the best TFIDF models but the Silhouette Index achieves a higher score. It is arguably better than the results on the same model from the original test. All results considered, no model from the noise reduction experiment performs better than the best models from the original test results. It is possible that doing the length restriction in another way would prove to be more successful, for example by only removing certain short messages. Overall the length restriction did not manage to produce results that was better than without it.

Out of the tested models, it is difficult to pick one clear winner. Each have their advantages and disadvantages. Two protruding models are TFIDF (uni) with K-means and TFIDF (uni) with HDBSCAN. DBSCAN produces similar results as HDBSCAN, with the drawback of it being sensitive to the choice of parameters. When changing data size and chat length, different parameters are required which makes DBSCAN produce less consequent results than HDBSCAN and is thereby less suitable for the task. However, the high outlier rate is still a problem, which is why the K-means model is competitive. It takes all data points into account with the drawback of producing more noisy clusters compared to the more pure clusters created by the HDBSCAN model. Pure clusters in this case means that the clusters contain chats from one dominant category from the test set, whereas noisy clusters contains chats from mixed categories. Both models are promising and will be considered going forward.

5.1.2 RQ2: Evaluation of Data Increase

The goal of investigating how performance is impacted by the data size is to find how robust the performance of the models are. For the first model, TFIDF (uni) with K-means (see Table 4.4), at 2 000 chats, the performance is in line with the baseline and plateaus. This suggests that the analysis should not be performed on less than 2 000 chats. When increasing the number of chats, Silhouette Index stays constant while Adj. Rand Index increases. This means that the quality of the clusters increases as the number of chats increase, which is a desirable property to have.

For the second model, TFIDF (uni) with HDBSCAN (see Table 4.5), the performance reaches a plateau when using 1 000 chats. When using less than 1 000 chats, the performance decreases drastically. Surprisingly, when increasing the number of chats over 1 000, the Adj. Rand Index decreases marginally. However, Silhouette Index and outlier rate is kept constant above the threshold. This means that the model has a more difficult time to capture different categories when more chats are analyzed, which is not a desirable property. However, the decreasing Adj. Rand Index could be explained by the decreasing ratio of chats that was used for calculation of Adj. Rand Index. Since only the test set data could be used for this calculation, as the number of chats increased the proportion of the test data decreased. When using 2000 chats, 80% of the chats are from the test data set. When using 10 000 chats, only 16% of the chats are from the test data set. As this ratio decreases, Adj. Rand Index provides a worse picture of what categories the clusters are actually capturing.

Comparing the two models, TFIDF (uni) with K-means is more robust but produces noisy clusters whereas TFIDF (uni) with HDBSCAN produces more pure clusters at the risk of missing important aspects in the data. Both models needs to be evaluated against the specific use case investigated in this thesis, it is up to preference which model is the most useful.

5.1.3 RQ3: Evaluation of Summarization Models

K-Means and LSA are trying to find and extract different characteristics to the summary, while TextRank prefers messages that are similar to a lot of other messages, which often results in one characteristic. TextRank can also extract different characteristics, but only when the other characteristics are described by enough messages. This difference is clear when reading the resulting summaries. Only when TextRank misses the important characteristic, extracting different characteristics improved the ROUGE measure, see Table 4.8. When producing the summaries K-Means sometimes extracts a message from a small but well separated cluster. One example is English messages, there are a few English chats in the data and in K-Means these can be chosen to be included in the summary. To avoid this one could try to create a larger number of clusters than messages to extract. For example, create 12 clusters and then extract one message from the 10 largest clusters. The same applies for LSA but here one would create a larger number of topics. There are versions of TextRank that also tries to extract different aspects by penalizing similar messages after extracting each message.

Noise reduction experiment improves the results, see Table 4.7. It removes both messages containing less than 5 words and messages containing the substring “Vera:”, as can be read in section 3.8. Performing noise reduction removed 13% of all important messages while it removed 67% of all messages. This could explain why noise reduction improves the result, even for random summarization. Because it increases the proportion of important messages and therefore also the probability to extract an important message.

The baseline Largest summarization performed best in terms of ROUGE measure. Here it is important to notice that ROUGE is favoring large summaries because it is a recall measure, see section 2.3.5. It is calculated by dividing the number of co-occurring n-grams with the total n-grams in the referents summary. This gives a large candidate summary a larger probability to match n-grams without penalizing it for being bigger. FastText generally produced larger summaries than the others and is also favored by this. However, large messages could still be better. They could give more context, be more well written and therefore explain the category at hand in a better way. This is supported by the fact that important messages tend to be longer than general messages, 20 against 12 in average word length.

The choice of text representation has a large impact on the summaries produced. All combination together with TFIDF and S-BERT obtained a score equal to random, while FastText managed to obtain a score better than random and is considered to perform best when summarizing this data.

All TFIDF summarization models did obtain low scores, often equal to random. When investigating the summaries, TFIDF seem to favor messages that are repeated in the data, for example standardized answers from the customer support. For TFIDF it is important that two messages contain the same exact words to increase similarity. TFIDF will differentiate between two synonyms and two spellings of the same word. Also, when comparing two identical sentences, even when these messages are short, they will obtain a low cosine distance which increases their probability to be extracted to the summary. The messages being summarized are quite short and that also impacts the result negatively. With few words to be represented, the TFIDF vectors becomes even more sparse and this makes it harder for cosine distance to produce a fair score.

S-BERT also seems to favor short but not necessarily identical messages. Together with TextRank it extracts many messages that are written after the customers problem are solved. Messages that are expressing gratitude. This can explain the lower ROUGE measure for S-BERT with TextRank. For K-means it is clear that different aspects are found, as discussed above, where one message in the summary usually expresses gratitude and others could be a greeting or a standard chatbot message. Greetings and stopwords, like the Swedish word for thanks, are not removed since the pre-processing steps are not used for S-BERT. One could argue that these staring/ending messages should be removed when trying to create a summary. Probably it is true in this case, they do not add any valuable information and when running the noise reduction experiment these starting/ending messages are removed, because they often are shorter than five words. The chatbot messages are also removed in the noise reduction experiment and this gave an improvement on the result. After noise reduction, S-BERT is slightly better than random for both model combinations.

FastText obtained the best performance for all models. Both models with FastText seem to favor longer messages to be extracted to the summary. The question is why it does not favor the shorter messages, as TFIDF and S-BERT. Longer messages tend to have more common words, that not are stop words, which may allow longer messages to be more similar. For TFIDF the importance of these common words are lowered by the IDF term and S-BERT aims to find the overall meaning of the sentence caring less about the common words in themselves. FastText is also supposed to incorporate the meaning of the words in their representations. This could allow FastText models to see beyond the short identical messages that TFIDF and S-BERT get stuck on. S-BERT is also supposed to capture meaning, by capturing context of the entire sentence. Therefore, S-BERT was also expected to see past these short messages. But, as discussed above, not using pre-processing before S-BERT could be a difference that matters. The longer two texts becomes, the more similar they should become when using FastText average word embedding. However, WMD should probably be able to separate longer messages better.

When using the models, FastText with average word embedding obtained the same score as FastText with WMD. One conclusion of that could be that average word embedding keeps the differences in these messages due to their shorter length. To investigate this one can read through the summaries to see if they do extract exactly the same messages. One could also investigate the similarity between some messages in the data, both shorter and longer, to compare the similarity they get from the two representations.

The summarization model combinding TFIDF (bi) with K-Means was not able to run in Colab. TFIDF (bi) produced eight times longer representations than TFIDF (uni), and with the large number of messages in the data, the computation complexity made the environments time restriction run out. One could try this experiment on another environment. But this was not prioritized since the interest in this model combination was low due to two main reasons. One, the other results did not indicate that this model would be any better. Two, the high computation complexity did not seem reasonable to use in a real word application.

Three models for extractive summarization has been compared before [50], and their implementations would correspond to our models using TFIDF. Their result showed that TextRank gained a higher ROUGE measure for both DUC 2002 and 2004. Our result differs from this previous work, typically TextRank performs worse than both K-means and LSA when combined with TFIDF. But all our TFIDF modles obtained low scores, scores comparing to the Random baseline, see first rows in Table 4.6. Only when combining TextRank with FastText, it performs better than K-means, this combination was also the best model in terms of

ROUGE. But still it did not obtain higher ROUGE than the baseline of extracting the largest messages. It was expected that the largest summarizer would obtain high ROUGE scores because ROUGE favors longer summaries (this is further discussed below in section 5.2), but it was not expected to obtain the highest score.

5.1.4 RQ4: Qualitative Evaluation (of Summarization Models)

In the qualitative evaluation the summarization model combining FastText and TextRank obtained the best total points. This model also obtained the fewest 0's, it did only fail to explain the category 3 times. Largest summarization did receive the highest ROUGE measure but in the qualitative evaluation it failed to explain the category 6 times, often resulting in extracting messages that seem to belong to other categories. This means that the model misses an important part of the chats that explains the category. Largest summarization is very sensitive to noise in the data and will extract anything large. If there is one noisy message and that message is the largest it will be extracted.

The qualitative evaluation could be improved on some points. It could involve some additional people. It was only performed by one person, this was because one person was considered an expert at the company and had a lot more knowledge about the chat data compared everyone else. However, this person was the only end user of the product, and was therefore considered enough. The qualitative evaluation could also be extended to more categories to summarize, either by summing Category 1 again but on new data or by summing categories smaller than category 15. To perform the evaluation on new data a larger test set had to be done. Summarizing like this, on the entire data set, was decided to be prioritized because a normal month for the company would hold twice as many chats. Further, the top 15 categories is decided to be of interest and the other categories are considered to be too small.

Two problems from the test data creation did also appear here. One, the problem of defining categories, making some chats possible to categorize under two categories. Two, some chats do talk about two problems and the customer support do solve both of these problems. In both cases, there are messages in the data that could mislead the user, and if these are extracted they could damage the clarity of the extracted summary. The best model, FastText with TextRank, did have less struggle with this. Probably due to it favoring, and extracting similar messages. Whereas combination with K-Means and LSA would potentially catch these when trying to capture different aspects.

A short and concise summary is probably more helpful than a long one, since it will take less time to read though and understand. This probably also depends on the task and preference of the end user. Therefore the qualitative evaluation yields an important comparison between Largest and FastText + TextRank. When reading the resulting summaries it seems as the models performing as random did have shorter summaries, that was not as helpful at explaining the category. Some categories are also more complex to define and in those cases, larger messages need to be extracted in the summary in order to capture enough context. When reading through the produced summaries it does not seem to exist a model creating summaries with low ROUGE that still is good at explaining the category.

5.2 Method

This section discusses and criticizes the method used in this thesis. Some limitations and unexpected behavior are mentioned and how it has impacted the results.

5.2.1 Test Data Creation

The creation of the data set was made to be similar to the current way of analyzing chats at the company. With the goal to evaluate the models according to this current way of analyzing. There were some challenges when deciding how the test data should be created. A category could always be divided into multiple smaller categories. Categories could also be grouped together into large, more including, categories.

Some chats could touch several problems, which makes them difficult to put into one single category. For example, a customer could be describing two separate problems at the same time or a customer could be describing a problem that falls in between two of the predefined categories. The categories themselves are very different, some are heavily bound to single keywords, while others are more difficult to recognize. For categories bound to a single keyword, the best way to extract those might be to simply search for that specific term. For others, the context might be very important. Because of this, some categories might be easier to cluster or to summarize.

Since the tested models were evaluated against the labeled categories, models that capture categories with the same granularity as the labeled data achieves higher scores. This makes it difficult to know whether models with lower scores are bad at clustering the data, or if they capture a different granularity of categories. A Clustering model can also capture useful patterns within the chats that are not present in the labeled chats, which also would contribute negatively towards its score. A different approach would be to have a few top-line categories and then divide them into smaller categories, like a tree tree structure. This would make it possible to evaluate on which level of granularity that each tested model has captured.

5.2.2 Pre-processing

In section 5.1.3, it was discussed that identical messages were extracted to the summary, it could be messages from the chatbot or standard replies from the customer support. These messages could be of different length, which is why not all of them were removed by the noise reduction experiment. The noise reduction did still improve the results when extracting a summary, but not for clustering. Clustering are potentially impacted by these identical messages. Some of these identical messages are category specific, and are only replied to one topic, which would increase the distance between two chats from different topics. Whereas some of these messages are more general, appearing in all categories, which would decrease the distance between two chats from different categories. One example is when the customer support use remote control on the customer's computer to solve the problem. This happens for many of the categories and when it does the customer support send a long message with the instructions to start remote control. This message will increase the similarity between two chats, even if these chats are from different categories. This message will also have a large probability to be extracted in the summary. The focus of this thesis has been on the different models for clustering and extractive summarization and to improve the results going forward, more effort could be put into the pre-processing step. Using the knowledge from the company expert, one could try to add a smart filter for some chatbot messages and standard replies. A filter that would remove generic messages, and keep the ones that are specific for a category.

For data that contain a lot of noise the pre-processing step will probably need more attention and become more important. In a large portion of the literature, pre-processing does not get so much attention. In many cases the used test data is easy to process, for example in the DUC conferences, and there are no need for smart pre-processing steps. Probably, in real a world application of NLP the data will contain more noise, and the pre-processing need to be smart and domain specific.

spaCy was used in the pre-processing step, as explained in section 3.4. But if spaCy is not used in other steps, removing it would decrease the RAM usage when not loading the model and it would also reduce computational complexity in the pre-processing step. If one would like to replace spaCy in the pre-processing step one can look at ICU-Tokenizer [1] for the tokenization step. Lowercasing, number and stop word removal can be implemented with basic Python functions or by adding already existing packages.

5.2.3 FastText Limitations

Since FastText performed well on summarization but not on clustering, a deeper investigation took place to understand why. One contributing factor was found, the Kungbib spaCy model did not work as expected. Some words did not get a representation and the resulting vector was filled with zeros. When FastText encounters an unknown word it should split the word into wordpieces, until it can represent each wordpiece and then sum the wordpieces to obtain the representation for the unknown word. Therefore every single word should be able to obtain a representation. To understand the impact of this, a test was conducted trying to understand how many words in the data that do miss a representation. It was done by looking at all words remaining after pre-processing and sorting them by their TF (term frequency) score. Then calculating the proportion of unknown words in the top X. Top 100 is the 100 words with the highest TF score. Top 200 is the 200 words with the highest TF score and so on. Below is a table that show the proportion of unknown words for each Top X.

Table 5.1: Words with missing FastText representation

Top X TF words	% of missing FastText representation
100	2.0
200	2.5
300	2.3
400	2.0
500	2.0
600	2.2
700	2.6
800	3.6
900	3.8
1 000	3.7
2 000	5.4
3 000	6.2
4 000	8.5
5 000	13.3
6 000	17.3
7 000	20.2
8 000	22.1
9 000	23.7
All unigrams: 9 615	24.6

As can be seen in the table above, 24.6% of all words did not obtain a representation, which is a lot. But for the words with a higher TFIDF score the result is better and there are fewer unknown words. In top 1000 words there are only 3.7% unknown words. There is a risk that category specific words can not be represented, which would negatively impact the result, especially for clustering. To improve this, one could potentially use the original FastText

---

[1] https://mingruimingrui.github.io/ICU-tokenizer/index.html

model[2]. That is the same model that should be implemented in the Kungbib spaCy model, but there it does not seem to work properly. The original FastText model exists for 157 different languages [18], one model for each language, including Swedish.

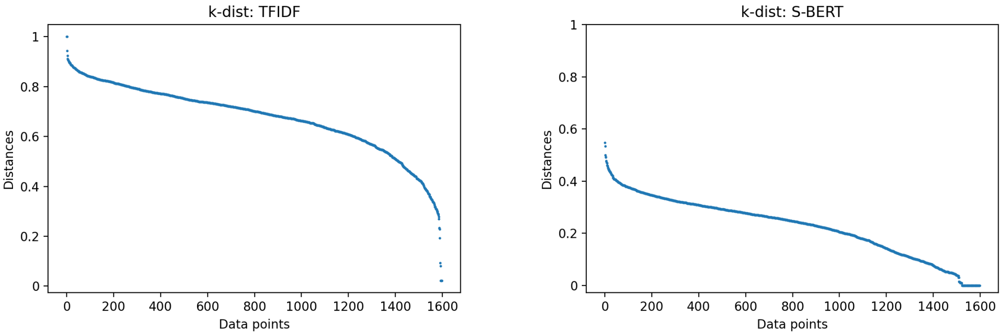
5.2.4 Parameter Turning

There is no golden standard for picking hyper parameters for clustering models, especially for DBSCAN. For K-means, it was relatively straightforward. Cluster size was picked to be 30 for all conducted experiments, which went along with the produced elbow plot. The test data set contained 40 categories, which also suggests that 30 is a good number since the smallest 10 categories only make up 3% of the test data.

minPts for DBSCAN and HDBSCAN was set to 3 for all tests, due to the model's reasonable and stable clusters. The most difficult parameter to set was ϵ for DBSCAN. Initially, a value from a kdist plot (see section 3.6.1) was picked. As can be seen in the produced kdist plots, see Table 5.1, all plots follow the same pattern. However, the kdist plots for FastTest and WMD have flatter curves which means that the outlier rate is more sensitive to ϵ. This might explain why performance using FastText and WMD was poor. In the S-BERT kdist plot below, some values with distance 0 can be seen to the left. This means that these chats basically have the same representations as their 5 closest neighbors, which is not positive. The picked values for ϵ worked as a baseline. However, the produced clusters were not very helpful. In many cases only 1 cluster was produced. Some adjustments were necessary in order to achieve stable output. As a consequence of this, the outlier rate increased. This means that no heuristic or rule for picking ϵ was found. A different ϵ value was needed for each text representation, even for a specific text representation, it is not clear what value for ϵ produced the best clustering. This makes DBSCAN unstable and difficult to use for this application. This was especially true when trying to use WMD together with DBSCAN and HDBSCAN, where no stable result was achieved.

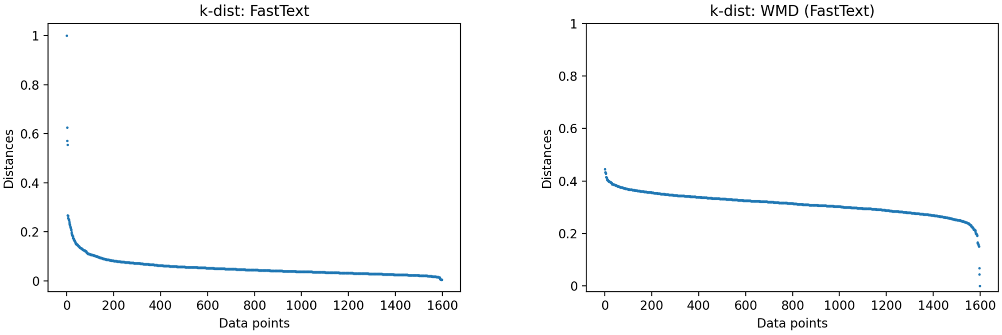
---

[2] https://fasttext.cc/docs/en/crawl-vectors.html



(a) kdist for TFIDF (k=5)

(b) kdist for S-BERT (k=5)



(c) kdist for FastText (k=5)

(d) kdist for WMD (k=5)

Figure 5.1: Kdist plots for the representations used in the thesis

5.2.5 Distance Measures & Clustering

It was decided to use cosine distance wherever possible in this thesis, for both clustering and summarization. This was done because cosine distance supposedly is a better measure for distance for the text representations used in this thesis, see 2.2 Distance Measures. In hindsight, it would have been interesting to have the use of cosine distance compared to a baseline of Euclidean distance to see if it improved the performance of the tested models.

Compared to Euclidean distance, cosine distance is not a “real” distance measure in a mathematical sense, since it does not satisfy the triangle inequality. It might also be unintuitive. For example, when examining data points in a plot, data points “close” to each other are only close in a Euclidean distance way of measuring. This might make cosine distance questionable to use as a distance measure in clustering algorithms. However, it is widely used in literature on similar tasks. `scikit-learn` is a well known and reputable library, and the fact that its implementation of K-means did not support cosine distance was a bit alarming. However, nothing pointed towards that K-means with a cosine distance measure was not a legitimate model. Therefore, another implementation that allowed the use of a cosine distance measure was found. All other clustering algorithms, implemented using `scikit-learn` had support for cosine distance.

5.2.6 Dimensionality reduction

Text representations faces the “curse of dimensionality” due to its large and sparse representations. This effect is especially large for TFIDF, since it has significantly larger vector length compared to S-BERT and FastText. In some cases, clustering algorithms that are effective on small dimensions perform worse on high dimensional data [2]. Steinbach et al. conclude that dimension order is one of the problems affecting a cluster analysis, because when the number of dimensions goes towards infinity, the difference between the closest and furthest data point goes to zero [47]. Adding pre-processing steps (as in this thesis) before creating a TFIDF representation decreases the dimensionality by reducing the amount of measured terms and thereby potentially increases quality. Pruning is also a technique that is commonly used for TFIDF, to reduce dimensionality by removing rare terms. Removing terms that appear in less than 1% of the documents seem to be common when pruning, although little research has been done investigating the impact of pruning [11]. Other techniques to handle large dimensions are dimensionality reduction algorithms. Kadhim et al. tried dimension reduction with LSA and then performed K-means clustering, reducing dimensions from 5 000 - 10 000 to 100, and obtained approximately the same accuracy [24]. In conclusion it is not certain that a clustering algorithm performs well on large dimension data. At the same time it is not certain that using a dimension reduction technique will improve the results. It comes down to the specific problem and data, therefore it is important to develop and evaluate the algorithm on the task at hand.

Dimensionality reduction is not a topic that was investigated in this thesis. It would be interesting to investigate to understand if it would reduce the effect of the "curse of dimensionality", discussed above. It is possible that any of the clustering models tested in this thesis would have its performance increased by using some dimensionality reduction technique as an additional pre-processing step. On one hand, reducing the dimensionality would make it easier for the clustering models to produce meaningful clusters. But on the other hand, reducing the dimensionality will decrease the amount of information contained in the text representation. The question is if performance will increase as a consequence of these effects.

5.2.7 Alternative Approaches

Another approach for text representation would be using a ranking system, like BM25 [41]. BM25 is normally used for information retrieval tasks, such as ranking the relevance of a set of documents relative to a search query. The approach is similar to TFIDF, with the addition of taking in to account the probability of occurrence for each word. The effect of term frequency (TF) is dampened by taking into account the length of the document in relation to other documents in the corpus. Running BM25 on each document against all other documents the output will be a similarity matrix, which can be converted to a distance matrix and used as input for a clustering algorithm. BM25 would then be an alternative to TFIDF that could be used together with DBSCAN and HDBSCAN. BM25 could not be used with K-means since K-means does not support a pre-computed distance matrix. It would be interesting to investigate the usefulness of this method on the task in this thesis.

Supervised classification approaches would be possible to use instead of a clustering approach. A supervised approach is trained according to the categories and therefore it may perform better at separating them. To get a brief understanding of this area, a small test was conducted. One supervised approach, XLM-RoBERTa [3] classification implemented with SimpleTransformers [4], were trained and evaluated. It obtained very low accuracy. When Training on 1 200 chats and evaluating on 400 chats the model classified everything as the biggest category. Both when trying to learn all 40 categories or only the top 15 categories. The assumption made here was that more data is needed to be able to evaluate such a supervised approach. However, for the specific use case at Visma, it is not sure that a classification approach is preferred. With classification, the possible categories predicted by the model are fixed. In order to introduce new categories, new training data would need to be created and

---

[3] https://huggingface.co/transformers/model_doc/xlmroberta.html

[4] https://simpletransformers.ai/docs/classification-models/

presented to the model.

A promising extractive summarizing model that was not investigated in this thesis was Bayesian Sentence-based Topic Models (BSTM) [50]. It is a sentence based summarization model that take term-document and term-sentence associations into account. The approach is bayesian and based on Latent Dirichlet Allocation (LDA). The main idea for LDA is that documents are represented as a random mixture of latent topics, and every topic is a probability distribution over the words. In addition, BSTM builds the probability distribution of selecting a sentence given a topic. BSTM also use Kullback Lieber divergence (KL), which measures how similar two distributions are. This allows one to show that a good summary is similar (have a low divergence) to the original document. BSTM showed good performance on DUC 2002 and 2004. It was also compared to the three extractive summarization models in this thesis (models combined with TFIDF) and it obtained a higher ROUGE score on both data sets.

Unfortunately, to our knowledge, there is no implementation of BSTM to use out of the box. There are other similar approaches: HIERSUM [19] and BAYESUM [8], but no implementation of them was found either. However, there was potential in using a more plain LDA approach. Building an extractive summarization model from scratch using scikit-learns LDA model as a base. Due to time restrictions in this thesis, other parts were prioritized, and this model was not implemented. This model seem promising, but it is unsure if it would improve the performance over the models tested in this thesis. Because it builds on a TFIDF representation, which clearly obtained lower scores than summarization models using FastText.

5.2.8 Source Criticism

The majority of references are research papers proposing algorithms or models, while others include books that on a more top-line level cover areas such as NLP. The subject area, and especially neural models, has produced quite a large amount of scientific publications in recent years. Therefore, many research papers do not have a particularly high number of citations, which would otherwise be desirable. From a reliable source point of view, there is a trade-off between the number of citations and the freshness of the source. For very recent publications, it was important to remember that, while potentially being innovative and groundbreaking, might be faulty and incorrect. For other subject areas processed in this thesis, like clustering, the original sources were well known and extensively cited. When referencing software implementation, which has been done through footnotes, mostly well-known sources were used. These include packages like `scikit-learn` and `spaCy`. When using other, lesser-known packages, the source code was investigated to see if the implementations were correct.

5.3 The Work in a Wider Context

This section will discuss ethical and social aspects related to this thesis and subject area.

5.3.1 Product Improvements

The idea behind this analysis tool is to allow for more and better improvements of the product, by surfacing problems with the company's product. This hopefully could facilitate the work for Visma's customers, when using the products. Potentially this also allow the customer success teams to put less time aside for analyzing the chat data, and more time into other valuable tasks, such as helping the customer. In general, many companies do save chat data, but few utilize its entire potential, so maybe this can also inspire others to try to develop systems solving similar tasks.

5.3.2 Bias

Analyzing with clustering could favor large and common problems, with the risk of not surfacing small but critical problems. Clustering also introduces a problem for the analysis. The tool will group chats in one way, and there could be other ways of clustering the chats that will make other problems visible.

There is always a potential risk to introduce bias in the analysis tool. Bias can be introduced in several steps of the development process. (1) In pre-processing. (2) When training, or in this case when deciding to use a pre-trained model. (3) From the data when performing the analysis. (4) In the evaluation of the model, both in metrics used and when evaluating qualitatively. (5) In the interpretation of the result. (6) In future adjustments of the model when trying to improve the results. One has to consider biases in all steps, bias that could cause harm in any way. An example of possible bias introduced by the used metrics is if they were boosted by a certain pattern in the chats. This could favor some specific pre-processing steps that would falsely suggest increased performance, judged by the metrics, while the actual usefulness of the produced cluster decreases. To tackle this issue, it would help to introduce a qualitative evaluation step between each iteration to ensure that an increase in the metric values corresponds to a better model.

5.3.3 User Privacy

The analysis tool will handle personal information and with the use of an extractive summarization model, the private information could be extracted and presented in a summary. If personal information is extracted to a summary, the information is very exposed. Therefore, it will be important to have routines and rules on who can read these summaries and how these summaries can be used. Due to user privacy it is also important how the model is implemented and how the output is stored, also to not violate any laws protecting user privacy. One could also consider adding an anonymization step to tackle this issue.

6 Conclusion

This study investigates a tool for analyzing customer chat dialogues, by comparing a wide variety of models on the task at hand. Combining three text representations (TFIDF, S-BERT and FastText) with three clustering algorithms (K-means, DBSCAN and HDBSCAN) and three extractive summarization models (K-means, LSA and TextRank).

The result from clustering shows that there are two models proven to be the most promising, TFIDF with K-means and TFIDF with HDBSCAN. There are trade-offs with each model and in order to decide which model is the most useful, user preferences need to be considered. For clustering, it is shown that the models using S-BERT and FastText perform subpar to the models using TFIDF. The results also show that the two best performing models are not sensitive to increasing the amount of data analyzed. However, these models have a lower limit of chats that is needed to produce stable results with predictable accuracy. The noise reduction experiment made the clustering algorithms produce more separated clusters, at the expense of not distinguishing categories as accurately. A problem when clustering was that many chats from different categories had a low distance between each other. Sometimes the content could be very similar, with some pre-written messages appearing across all categories. Future work could focus on the pre-processing step, trying to remove similarities to increase the distance between chats from different categories, to improve clustering performance.

The result from summarization shows that two models, FastText with TextRank and the baseline of extracting the largest messages, obtained the highest ROUGE measure. Longer messages seemed to contain more valuable information and the result was improved with the noise reduction experiment. Models using TFIDF and S-BERT did all obtain results similar to random. When evaluating the two best-performing models qualitatively, the summaries produced by FastText with TextRank were more often preferred and did fail to explain the category fewer times.

In future work, it would be interesting to try a supervised approach. It is expected that a supervised classification approach would be better at finding categories in the data compared to an unsupervised approach. A small test was conducted for one supervised approach, XLM-

RoBERTa[1] classification implemented with SimpleTransformers[2], and it did not perform well at all. The assumption made here was that more data is needed to be able to evaluate such a supervised approach. However, for the specific use case at Visma, it is not certain that a classification approach is preferred. Another improvement that could be investigated is training S-BERT or FastText, specifically for this use case. This could potentially improve the produced embeddings, giving the clustering algorithms an easier time separating the data.

NOTE:
After the thesis was finished, some further testing of the clustering algorithms was performed without an increase in clustering performance. In the end, the analysis tool was not implemented at the company.

---

[1] https://huggingface.co/transformers/model_doc/xlmroberta.html

[2] https://simpletransformers.ai/docs/classification-models/

Bibliography

[1] Charu C Aggarwal and ChengXiang Zhai. “A survey of text clustering algorithms”. In: *Mining text data*. Springer, 2012, pp. 77–128.

[2] Ira Assent. “Clustering high dimensional data”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2.4 (2012), pp. 340–350.

[3] Federico Barrios, Federico López, Luis Argerich, and Rosa Wachenchauzer. “Variations of the Similarity Function of TextRank for Automated Summarization”. In: (2016).

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. “Enriching word vectors with subword information”. In: *Transactions of the Association for Computational Linguistics* 5 (2016), pp. 135–146.

[5] Sergey Brin and Lawrence Page. “The anatomy of a large-scale hypertextual web search engine”. In: *Computer networks and ISDN systems* 30.1-7 (1998), pp. 107–117.

[6] Ricardo JGB Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. “Hierarchical density estimates for data clustering, visualization, and outlier detection”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 10.1 (2015), pp. 1–51.

[7] Jianpeng Cheng and Mirella Lapata. “Neural Summarization by Extracting Sentences and Words”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016, pp. 484–494.

[8] Hal Daumé III and Daniel Marcu. “Bayesian Query-Focused Summarization”. In: *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. 2006, pp. 305–312.

[9] David L Davies and Donald W Bouldin. “A cluster separation measure”. In: *IEEE transactions on pattern analysis and machine intelligence* 2 (1979), pp. 224–227.

[10] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. “Indexing by latent semantic analysis”. In: *Journal of the American society for information science* 41.6 (1990), pp. 391–407.

[11] Matthew Denny and Arthur Spirling. “Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it”. In: *When It Misleads, and What to Do about It (September 27, 2017)* (2017).

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2019, pp. 4171–4186.

[13] Günes Erkan and Dragomir R Radev. "Lexrank: Graph-based lexical centrality as salience in text summarization". In: *Journal of artificial intelligence research* 22 (2004), pp. 457–479.

[14] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.

[15] Aysu Ezen-Can and Kristy Elizabeth Boyer. "Unsupervised classification of student dialogue acts with query-likelihood clustering". In: *Educational Data Mining 2013*. 2013.

[16] Thomas S Ferguson. "A Bayesian analysis of some nonparametric problems". In: *The annals of statistics* (1973), pp. 209–230.

[17] Yihong Gong and Xin Liu. "Generic text summarization using relevance measure and latent semantic analysis". In: *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. 2001, pp. 19–25.

[18] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. "Learning Word Vectors for 157 Languages". In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*. 2018.

[19] Aria Haghighi and Lucy Vanderwende. "Exploring content models for multi-document summarization". In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 2009, pp. 362–370.

[20] Lawrence Hubert and Phipps Arabie. "Comparing partitions". In: *Journal of classification* 2.1 (1985), pp. 193–218.

[21] Reza Nurul Gayatri Indah, Rice Novita, Oktaf Brillian Kharisma, Rian Vebrianto, Suwanto Sanjaya, Tuti Andriani, Wardani Purnama Sari, Yulia Novita, Robbi Rahim, et al. "DBSCAN algorithm: twitter text clustering of trend topic pilkada pekanbaru". In: *Journal of Physics: Conference Series*. Vol. 1363. 2019.

[22] Jungsun Jang, Yeonsoo Lee, Seolhwa Lee, Dongwon Shin, Dongjun Kim, and Haechang Rim. "A novel density-based clustering method using word embedding features for dialogue intention recognition". In: *Cluster Computing* 19.4 (2016), pp. 2315–2326.

[23] Armand Joulin, Édouard Grave, Piotr Bojanowski, and Tomáš Mikolov. "Bag of Tricks for Efficient Text Classification". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. 2017, pp. 427–431.

[24] Ammar Ismael Kadhim, Yu-N Cheah, and Nurul Hashimah Ahamed. "Text document preprocessing and dimension reduction techniques for text document clustering". In: *2014 4th International Conference on Artificial Intelligence with Applications in Engineering and Technology*. IEEE. 2014, pp. 69–73.

[25] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. "From word embeddings to document distances". In: *International conference on machine learning*. PMLR. 2015, pp. 957–966.

[26] Chin-Yew Lin. "Rouge: A package for automatic evaluation of summaries". In: *Text summarization branches out*. 2004, pp. 74–81.

[27] Yang Liu. "Fine-tune BERT for Extractive Summarization". In: *arXiv e-prints* (2019), arXiv–1903.

[28] James MacQueen et al. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[29] Leland McInnes, John Healy, and Steve Astels. "hdbscan: Hierarchical density based clustering". In: *The Journal of Open Source Software* 2.11 (2017), p. 205.

[30] Rada Mihalcea and Paul Tarau. "Textrank: Bringing order into text". In: *Proceedings of the 2004 conference on empirical methods in natural language processing*. 2004, pp. 404–411.

[31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space". In: *arXiv e-prints* (2013), arXiv–1301.

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. "Distributed representations of words and phrases and their compositionality". In: *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*. 2013, pp. 3111–3119.

[33] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond". In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. 2016, pp. 280–290.

[34] Paula S Newman and John C Blitzer. "Summarizing archived discussions: a beginning". In: *Proceedings of the 8th international conference on Intelligent user interfaces*. 2003, pp. 273–276.

[35] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. *The PageRank citation ranking: Bringing order to the web.* Tech. rep. Stanford InfoLab, 1999.

[36] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. "Bleu: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002, pp. 311–318.

[37] William M Rand. "Objective criteria for the evaluation of clustering methods". In: *Journal of the American Statistical association* 66.336 (1971), pp. 846–850.

[38] Nils Reimers, Iryna Gurevych, Nils Reimers, Iryna Gurevych, Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2019.

[39] Ehud Reiter. "A structured review of the validity of BLEU". In: *Computational Linguistics* 44.3 (2018), pp. 393–401.

[40] Pengjie Ren, Furu Wei, Zhumin Chen, Jun Ma, and Ming Zhou. "A redundancy-aware sentence regression framework for extractive summarization". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2016, pp. 33–43.

[41] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

[42] Peter J Rousseeuw. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.

[43] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. "A metric for distributions with applications to image databases". In: *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*. IEEE. 1998, pp. 59–66.

[44] Gerard Salton and Christopher Buckley. "Term-weighting approaches in automatic text retrieval". In: *Information processing & management* 24.5 (1988), pp. 513–523.

[45] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. "DBSCAN revisited, revisited: why and how you should (still) use DBSCAN". In: *ACM Transactions on Database Systems (TODS)* 42.3 (2017), pp. 1–21.

[46] Amit Singhal et al. "Modern information retrieval: A brief overview". In: *IEEE Data Eng. Bull.* 24.4 (2001), pp. 35–43.

[47] Michael Steinbach, Levent Ertöz, and Vipin Kumar. "The challenges of clustering high dimensional data". In: *New directions in statistical physics*. Springer, 2004, pp. 273–309.

[48] Jiliang Tang, Xufei Wang, Huiji Gao, Xia Hu, and Huan Liu. "Enriching short text representation in microblog for clustering". In: *Frontiers of Computer Science* 6.1 (2012), pp. 88–101.

[49] Robert L Thorndike. "Who belongs in the family?" In: *Psychometrika* 18.4 (1953), pp. 267–276.

[50] Dingding Wang, Shenghuo Zhu, Tao Li, and Yihong Gong. "Multi-document summarization using sentence-based topic models". In: *Proceedings of the ACL-IJCNLP 2009 conference short papers*. 2009, pp. 297–300.

[51] Jianhua Yin, Daren Chao, Zhongkun Liu, Wei Zhang, Xiaohui Yu, and Jianyong Wang. "Model-based clustering of short text streams". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018, pp. 2634–2642.

[52] Jianhua Yin and Jianyong Wang. "A dirichlet multinomial mixture model-based approach for short text clustering". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 233–242.

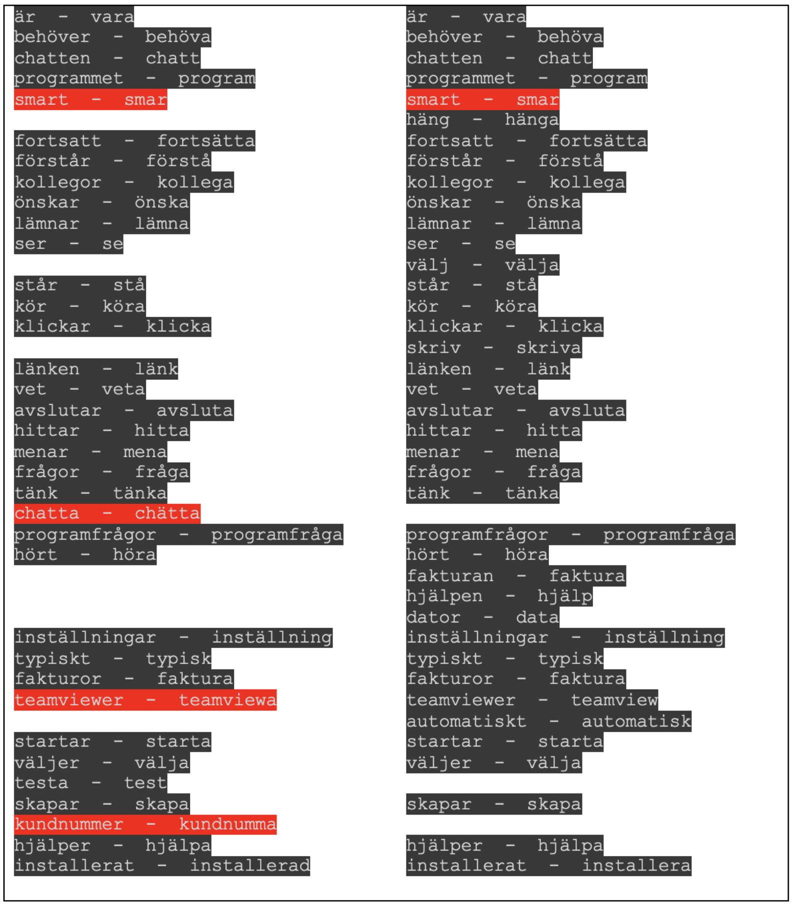
[53] Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. "Neural Document Summarization by Jointly Learning to Score and Select Sentences". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 654–663.

A Stop Words

['säger', 'fick', 'nedersta', 'vilka', 'säga', 'dagen', 'alltid', 'andras', 'ofta', 'i', 'alla', 'senare', 'sjutton', 'dina', 'sju', 'varit', 'inte', 'över', 'gör', 'sade', 'din', 'deras', 'några', 'mitt', 'siste', 'ditt', 'större', 'hans', 'hennes', 'varifrån', 'högst', 'beslutat', 'fjärde', 'till', 'innan', 'för', 'åttonde', 'adjö', 'ert', 'ute', 'borta', 'igår', **'okej'**, 'övermorgon', 'elfte', 'enligt', 'men', 'tio', 'ibland', 'nedre', 'bra', 'möjligtvis', 'vänstra', 'gälla', 'ner', 'förra', 'ni', 'nittonde', 'tidigare', 'den', 'längre', 'överst', 'flesta', 'åttionde', 'nödvändigt', 'bättre', 'fler', 'sedan', 'sin', 'haft', 'behöva', 'möjligt', 'gjorde', 'fem', 'även', 'lättast', 'sista', 'vi', 'beslut', 'nionde', 'nittio', 'bli', 'övre', 'lätt', 'idag', 'inga', 'olikt', 'vårt', 'oftast', 'annat', 'genast', 'viktig', 'mera', 'sagt', 'nio', 'blivit', 'tjugo', 'varsågod', 'mig', 'båda', 'före', 'länge', 'smått', 'imorgon', 'sjuttonde', 'mindre', 'legat', 'alltså', 'rätt', 'gällt', 'jämfört', 'inom', 'att', 'mest', 'detta', 'aderton', 'skulle', 'senast', 'adertonde', 'finns', 'kunnat', 'värre', 'slutligen', 'bäst', 'elva', 'sjuttio', 'tillsammans', 'gick', 'vilket', 'vill', 'långsam', 'genom', 'något', 'bort', 'ursäkt', 'dagarna', 'olika', 'någonting', 'eftersom', 'bakom', 'sexton', 'tvåhundra', 'sextio', 'arton', 'ligger', 'dig', 'vad', 'mina', 'rakt', 'vänster', 'dag', 'om', 'gott', 'ingenting', 'trettonde', 'lika', 'från', 'ja', 'hellre', 'med', 'många', 'tolv', 'inne', 'hundra', 'och', 'viktigt', 'sjätte', 'hon', 'fjorton', 'gå', 'dock', 'vems', 'kanske', 'kommit', 'sina', 'enkelt', 'knappast', 'lilla', 'tills', 'ur', 'åtta', 'tjugotvå', 'ha', 'litet', 'får', 'tionde', 'femton', 'bara', 'följande', 'fin', 'hur', 'fyrtionde', 'hundraen', 'nödvändiga', 'gärna', 'var', 'aldrig', 'långt', 'tjugonde', 'henne', 'lite', 'liten', 'ska', 'därför', 'delen', 'ett', 'noll', 'enkla', 'har', 'inför', 'upp', 'vidare', 'nödvändig', 'dagar', 'långsamt', 'en', 'mellan', 'helst', 'små', 'göra', 'hundraett', 'sextionde', 'samma', 'långsammast', 'annan', 'komma', 'när', 'hög', 'femtionde', 'honom', 'också', 'han', 'nitton', 'tack', **'tja'**, 'höger', 'tolfte', 'trettionde', 'ligga', 'blev', 'godare', 'sent', 'vilken', 'sjuttionde', 'vår', 'dem', 'långsammare', 'någon', 'sex', 'in', 'behövde', 'ut', 'oss', 'går', 'tretton', 'viktigast', 'kvar', 'möjligen', 'viktigare', 'hit', 'sig', 'ettusen', 'fanns', 'tjungo', 'här', 'få', 'av', 'flera', 'kr', 'kunde', 'likställd', 'nederst', 'finnas', 'mer', 'bådas', 'det', 'min', 'vart', 'ännu', 'bland', **'vera'**, 'framför', 'på', 'under', 'gjort', 'andra', 'helt', 'utan', **'hejsan'**, 'tidig', 'era', 'heller', 'man', 'åtminstone', 'möjlig', 'allt', 'mot', 'fyrtio', **'hej'**, 'godast', 'snart', 'så', 'eller', 'kommer', 'störst', 'två', 'fyra', 'jag', 'minst', 'nästa', 'tjugoett', 'inget', 'nummer', 'stor', 'som', 'beslutit', 'nu', 'dess', 'lättare', 'förlåt', 'ned', 'utanför', 'efter', 'enkel', 'gäller', 'tjugoen', 'längst', **'tjenare'**, 'redan', 'femtio', 'de', 'nödvändigtvis', 'allas', 'sist', 'ingen', 'sextonde', 'åttio', 'er', 'trettio', 'måste', 'god', 'fjortonde', 'gått', **'ok'**, 'tidigast', 'kan', 'igen', 'likställda', 'sämst', 'hade', 'nr', 'mittemot', 'nittionde', 'behövas', 'tre', 'femte', 'stort', 'vid', 'tidigt', 'del', 'vara', 'varken', 'första', 'artonde', 'än', 'behövt', 'nog', 'tredje', 'vem', 'stora', 'fått', 'dit', 'fram', 'sitt', 'tjugotre', 'där', 'goda', 'femtonde', 'kom', 'då', 'kunna', 'inuti', 'mycket', 'verkligen', 'sjunde', 'högre', 'nej', 'varför', 'blir', 'du', 'våra', 'skall', 'sämre']
Bold words are not standard stopwords in Kungbib spaCy model.

B Pre-study of Lemmatizer

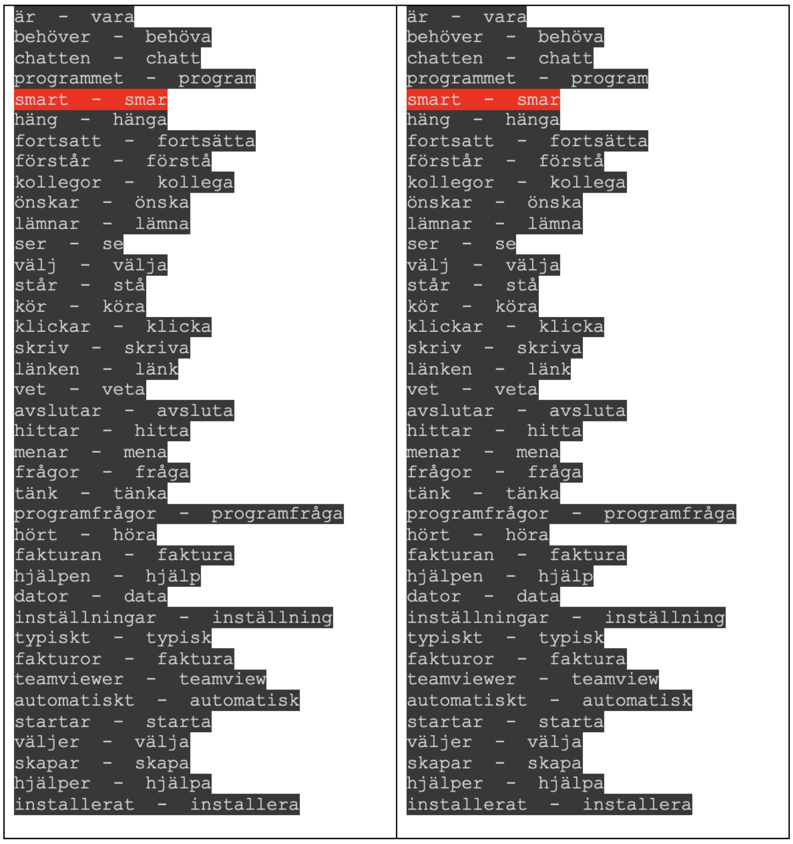
UDpipe vs Stanza



```
är  -  vara                         är  -  vara
behöver  -  behöva                  behöver  -  behöva
chatten  -  chatt                   chatten  -  chatt
programmet  -  program              programmet  -  program
smart  -  smar                      smart  -  smar
                                    häng  -  hänga
fortsatt  -  fortsätta              fortsatt  -  fortsätta
förstår  -  förstå                  förstår  -  förstå
kollegor  -  kollega                kollegor  -  kollega
önskar  -  önska                    önskar  -  önska
lämnar  -  lämna                    lämnar  -  lämna
ser  -  se                          ser  -  se
                                    välj  -  välja
står  -  stå                        står  -  stå
kör  -  köra                        kör  -  köra
klickar  -  klicka                  klickar  -  klicka
                                    skriv  -  skriva
länken  -  länk                     länken  -  länk
vet  -  veta                        vet  -  veta
avslutar  -  avsluta                avslutar  -  avsluta
hittar  -  hitta                    hittar  -  hitta
menar  -  mena                      menar  -  mena
frågor  -  fråga                    frågor  -  fråga
tänk  -  tänka                      tänk  -  tänka
chatta  -  chätta
programfrågor  -  programfråga      programfrågor  -  programfråga
hört  -  höra                       hört  -  höra
                                    fakturan  -  faktura
                                    hjälpen  -  hjälp
                                    dator  -  data
inställningar  -  inställning       inställningar  -  inställning
typiskt  -  typisk                  typiskt  -  typisk
fakturor  -  faktura                fakturor  -  faktura
teamviewer  -  teamviewa            teamviewer  -  teamview
                                    automatiskt  -  automatisk
startar  -  starta                  startar  -  starta
väljer  -  välja                    väljer  -  välja
testa  -  test
skapar  -  skapa                    skapar  -  skapa
kundnummer  -  kundnumma
hjälper  -  hjälpa                  hjälper  -  hjälpa
installerat  -  installerad         installerat  -  installera
```

Figure B.1: Result comparing UDpipe and Stanza

Stanza applied on No Kungbib spaCy lemma vs Kungbib spaCy lemma



```
är  -  vara
behöver  -  behöva
chatten  -  chatt
programmet  -  program
smart  -  smar
häng  -  hänga
fortsatt  -  fortsätta
förstår  -  förstå
kollegor  -  kollega
önskar  -  önska
lämnar  -  lämna
ser  -  se
välj  -  välja
står  -  stå
kör  -  köra
klickar  -  klicka
skriv  -  skriva
länken  -  länk
vet  -  veta
avslutar  -  avsluta
hittar  -  hitta
menar  -  mena
frågor  -  fråga
tänk  -  tänka
programfrågor  -  programfråga
hört  -  höra
fakturan  -  faktura
hjälpen  -  hjälp
dator  -  data
inställningar  -  inställning
typiskt  -  typisk
fakturor  -  faktura
teamviewer  -  teamview
automatiskt  -  automatisk
startar  -  starta
väljer  -  välja
skapar  -  skapa
hjälper  -  hjälpa
installerat  -  installera
```

```
är  -  vara
behöver  -  behöva
chatten  -  chatt
programmet  -  program
smart  -  smar
häng  -  hänga
fortsatt  -  fortsätta
förstår  -  förstå
kollegor  -  kollega
önskar  -  önska
lämnar  -  lämna
ser  -  se
välj  -  välja
står  -  stå
kör  -  köra
klickar  -  klicka
skriv  -  skriva
länken  -  länk
vet  -  veta
avslutar  -  avsluta
hittar  -  hitta
menar  -  mena
frågor  -  fråga
tänk  -  tänka
programfrågor  -  programfråga
hört  -  höra
fakturan  -  faktura
hjälpen  -  hjälp
dator  -  data
inställningar  -  inställning
typiskt  -  typisk
fakturor  -  faktura
teamviewer  -  teamview
automatiskt  -  automatisk
startar  -  starta
väljer  -  välja
skapar  -  skapa
hjälper  -  hjälpa
installerat  -  installera
```

Figure B.2: Result comparing without and with Kungbib spaCy lemmatization

C Data Set Example

The example below shows how a conversation between a client and customer support can look like. It is representative of all chats used within this thesis. The clients starts the conversation with the chat bot, and after a while it continues on with a human customer support. As can be seen, some misspelling do occur.

```
> Chatten är placerad i kö.
> Du kan hjälpa oss att bli bättre!
> Berätta hur du upplevde kontakten med oss efter chatten.
> Din rådgivare Tomas har kommit in i chatten
> Välkommen, jag heter Tomas, vad kan jag hjälpa dig med?
> Summary:
> product:  eEkonomi Smart
> Conversation:
> Vera:  Hej, jag är chatbot Vera.  Vad kan jag hjälpa dig med?
(Skriv kort och koncist!)
> Kund:  Hej, jag har råkat radera ej bokförda bankhändelser som
skulle ligga kvar.  Nu stämmer inte mitt saldo.  Hur gör jag nu?
Kan jag hitta de raderade bankhändeslerna?
> Vera:  Det går inte att radera en bankhändelse som har kommit
in via bankkopplingen.  Detta eftersom händelsen finns på ditt
kontoutdrag och därför också ska bokföras.
> Har du redan bokfört händelsen via en manuell verifikation
rekommenderar vi att du korrigerar verifikationen under Bokföring -
Verifikationer.  Matcha sedan bankhändelsen som vanligt!
> Kund:  Jag har fler frågor
> Vera:  Absolut, välj en kategori nedan eller skriv din fråga.
> Kund:  Men hur hittar jag dessa?
> Vera:  Det där förstod jag inte.  Skriv din fråga <b>kort</b> och
<b>koncist</b> eller börja om guiden.
> Vera:  Jag ska se efter om en kollega finns tillgänglig och kan
hjälpa dig.  Häng kvar!
```

```
> Vera:  Välj produkten du behöver hjälp med.
> Kund:  eEkonomi Smart
> Hej, jag har råkat radera ej bokförda bankhändelser som skulle
ligga kvar.  Nu stämmer inte mitt saldo.  Hur gör jag nu?  Kan jag
hitta de raderade bankhändeslerna?
> Hej, låt mig se om de kan räddas, ett ögonblick
> Det dröjer tyvärr en stund till, men du är inte bortglömd.
> Ok
> Det är dessvärre så att du inte kan backa tillbaka och få tillbaka
dem.  Du behöver kolla på banken och ersätta de händelser som du
inte har kvar i eEkonomi manuellt.
> Ok, men de kommer inte att gå att få tillbaka visa från banken då
det legat några från 2015.  Kommer det att bli en massa problem när
jag kommer deklarera nu alltså?
> Hur menar du, är de försvunna bankhändelserna från 2015?
> När jag för över från min bank så bokför jag in visma de bankhändelser
men några som jag redan bokfört har hamnat under ej bokförda
bankhändelser.  Då jag vet att dessa har blivit bokförda sedan
tidigare så började jag ta bort några och sen upptäckte jag att det
diffade mellan banken och visam
> Alltså mellan bankkontot och visma.  Saldot ökade i visma, jämfört
med banken
> Ja, det låter underligt.  De borde ju inte påverka saldot om de
inte var bokförda...
> Jag tror att vi kanske ska göra så att jag ber en kollega ringa
upp dig för att undersöka det närmare.  Vilket telefonnummer kan vi
nå dig på?
> Phone Number
> Vad har du för kundnummer?
> Customer Number
> Tack, jag ska ordna så att du blir uppringd.  Men eftersom jag
bara kan skriva ett kortare meddelande till kollegan så r det bra om
du beskriver ärendet från början så att de får alla detaljer.
> Ok
> Tack
> Ha en trevlig dag och återkom om du har andra frågor!
> Desamma
> Vi vill veta vad du tycker!  Hur upplevde du din kontakt med oss?
> Survey Link
```