Western Norway
University of
Applied Sciences

# BACHELOR'S THESIS

## Word embeddings for recommending semantically similar support tickets

**Francis Soliman Dyrhovden**

**Espen Norvang**

**Morten Sund**

Bachelor's in Computer Science

Department of Computer Science, Electrical Engineering and Mathematical Sciences

Supervisor: Tosin Daniel Oyetoyan

Submission Date: 03.06.2021

I confirm that the work is self-prepared and that references/source references to all sources used in the work are provided, cf. Regulation relating to academic studies and examinations at the Western Norway University of Applied Sciences (HVL), § 10.

Western Norway University of Applied Sciences

Faculty of engineering and science

Department of Computer Science, Electrical Engineering and Mathematical Sciences

| | |
|---|---|
| *Rapportens tittel:*<br>Word embeddings for recommending semantically similar support tickets<br>Vektorisering av ord for å finne semantisk lignende brukerdefinerte feilmeldinger | *Dato:*<br>*03.06.2021* |
| *Forfatter(e):*<br>Francis Soliman Dyrhovden, Espen Norvang, Morten Sund | *Antall sider u/vedlegg: 36*<br>*Antall sider vedlegg: 4* |
| *Studieretning:*<br>Dataingeniør / Informasjonsteknologi | *Antall disketter/CD-er:*<br>0 |
| *Kontaktperson ved studieretning:*<br>Tosin Daniel Oyetoyan | *Gradering:*<br>Ingen |
| *Merknader:* | |

| | |
|---|---|
| *Oppdragsgiver:*<br>Vizrt | *Oppdragsgivers referanse:*<br>EB-16: Semantic similarity |
| *Oppdragsgivers kontaktperson:*<br>Nils Haldorsen | *Telefon:*<br>971 39 130 |

*Sammendrag:*

Denne rapporten beskriver utviklingen av en maskinlæringsmodell for å finne semantiske likheter mellom innkommende og eksisterende brukerdefinerte feilmeldinger.

This report describes the process of developing a machine learning model to find semantically similarities between incoming and existing support tickets.

*Stikkord: Semantic textual similarity, natural language processing, machine learning, word embeddings, information retrieval*

Høgskulen på Vestlandet, Fakultet for ingeniør- og naturvitskap

Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen

Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

# PREFACE

This report documents the background, research and work surrounding the bachelor's project *Word embeddings for recommending semantically similar support tickets* provided by Vizrt through Western Norway University of Applied Sciences. The report was written by Francis Soliman Dyrhovden, Espen Norvang and Morten Sund.

We would like to thank Vizrt for an incredibly exciting and challenging bachelor's thesis, and a special thanks to our external supervisor Nils Haldorsen for his interest and enthusiasm in this project.

A final thanks to our internal supervisor at Western Norway University of Applied Sciences, Tosin Daniel Oyetoyan, for his extensive domain knowledge and guidance throughout the project.

# TABLE OF CONTENT

# 1 INTRODUCTION

Vizrt is a world leading provider of visual storytelling tools for media content creators. Their long list of customers includes big media companies such as CNN, CBS, NBC, Fox, BBC and many more. They offer software-based solutions for highly demanding tasks such as real-time 3D graphics, studio automation, media asset management and journalist story tools. Viz Mosart is one of these products and is used for automating and connecting different components together in one single software. As with most other software, Viz Mosart is also prone to bugs, malfunction and user errors. This results in support tickets being submitted, and they are now looking to improve the process of handling these tickets.

## 1.1 Motivation and goal

Vizrt wants to find out if their customer support section can be improved by discovering semantic textual similarity between incoming support tickets and previously solved issues. The goal is to develop and train a machine learning model that, when presented with an incoming support ticket, can provide a list with the most semantically similar issues that already exist.

## 1.2 Context

Due to their large customer base, Vizrt receives support tickets on a regular basis. The tickets come in different forms and languages. Vizrt must go through each of these tickets manually to check if the problem can be solved, or if it has been solved before. This requires a lot of time and human resources, which could be spent elsewhere. By using all the existing data, the new solution could make it easier and less time consuming to work with in the future for incoming tickets.

## 1.3 Limitations

During the project we had to set some limitations that could impact the result of the project. The limitations that were set were mainly defined by the time period, resources and the scope of the project. Machine learning algorithms generally perform better with more training data; thus, lack of data can be a severe limitation for the result (Halevy, Norvig, & Pereira, 2009). In addition, the data provided by Vizrt is unlabelled. This limits our options for testing and evaluating the model.

The support tickets that arrive at Vizrt's support department often include the names of their customers. It has been agreed that this information is kept confidential, and all parties have signed a non-disclosure agreement. This will put some restrictions on how we work with the data, and how we discuss our results.

The group had little to no experience with working with machine learning and semantic similarity. The group had to spend some time in the beginning to get an overview of these topics, which affected the total time available to work on the project.

The world of machine learning is near to endless, with thousands of different ways to tackle your problems. By reading relevant literature and articles, we have attempted to narrow the scope of solutions we could use, since it would be too time consuming for us to test all of them.

## 1.4 Resources

For the machine learning model to work for the specific purpose, it must have relevant training data. The client must provide this for the task to be solved. Furthermore, machine learning often require a lot of processing power, especially if the training data is of the required size. Data and processing power will be the thesis' most critical resources, and both are provided by the client.

The task will be solved in the programming language Python, using the development environment Jupyter Notebooks. Furthermore, the task will be solved with different libraries for machine learning, in addition to libraries for preprocessing text.

Our domain knowledge about media production and its related software is at best limited. In order to verify that the results are satisfactory, we will require employees from Vizrt to evaluate if the system serves its purpose or not. Frequent meetings with our supervisors at both HVL and Vizrt have been important during the development process to ensure that we moved in the right direction.

## 1.5 Organization of the report

The report is structured into ten chapters. The first three chapters is reserved for the preparations and planning of the project, the next three chapters explains how the product was developed in detail along with the result, and the remaining chapters discuss and conclude the thesis' followed by literature, resources and appendices.
The first chapter presents a brief introduction to the project including limitations and relevant resources. The second chapter provides a more detailed description of the project along with initial requirements and specifications. Chapter three discuss different solutions, technologies, project plan and evaluation that we have been relevant. Chapter four detail the description of the final chosen product design and architecture. In the fifth chapter we present our evaluation methods along with presentation of the results. We then have a discussion and evaluate the consequences of the final project result in chapter six. In chapter seven, we make our conclusion and discuss possible further works for the project. In the following chapter we list our sources and references which were used for writing the thesis and solving our problem. Finally, chapter nine includes details about different risk factors that are related to our project, along with Gantt diagram. It also includes detailed description on how to use our solution.

# 2 PROJECT DESCRIPTION

This chapter describes the project's origin and initial requirements. This involves the practical background of the project, information about the project owner, previous work, initial solution idea and literature background.

## 2.1 Practical background

The project features an initial exploration on behalf of Vizrt to determine if their ecosystem can benefit from a machine learning-based model, where the goal is to streamline their customer support department. Recent advances in the topics of natural language processing and machine learning show promising results in determining the semantic textual similarity between texts.

An important part of this project is to try several setups with different word embeddings and machine learning models against the highly domain specific data in order to determine which implementation should be used for the final product. This will require several iterations of testing and tuning of parameters within the models, in addition to deciding on what will be a good evaluation metric.

### 2.1.1 Project owner

Vizrt has its own service departments that receive support tickets from customers, which is time consuming and costly. This has led to Vizrt considering the use of advanced technology such as machine learning to solve the problem more efficiently.

### 2.1.2 Previous work

Vizrt has not done any previous work on the particular idea. This has allowed us to start fresh and make most decisions ourselves, including the choice of programming language, frameworks and development environments.

Although Vizrt has not worked on this earlier, there is extensive research available on the topics of Natural Language Processing (NLP) and Semantic Textual Similarity (STS), and great progress has been made in recent years. This project leverages state-of-the-art techniques that have shown good results in determining how semantically similar two pieces of text are.

### 2.1.3 Initial requirements specification

The initial requirements for this task were mainly to explore whether Vizrt could benefit from using a machine learning model based on semantic textual similarity or not. The model would be trained on existing data, and when provided with a new, unseen support ticket, it should return a list of the *n* most similar tickets that have been previously solved.

### 2.1.4 Initial solution idea

The initial solution idea was to train a machine learning model based on recent advances in natural language processing and semantic textual similarity. In addition to the initial requirements, it should include a similarity score that can be used by Vizrt to set a threshold to determine what is recognized as similar. For this system to be relevant in the future, it would be beneficial to have functionality for retraining and updating the machine learning model. Figure 1 illustrates the initial idea as a stand-alone prediction system that can be used by Vizrt's customer support department. The flow of the system is as follows:

1. Vizrt receives a support ticket from a customer.
2. The support ticket is fed through a pipeline, a set of instructions that are executed in a fixed order, that performs preprocessing on it to make it ready for the machine learning model. The support ticket is now regarded as a query.
3. The query is fed to the machine learning model.

4. The model lists the *n* most similar support tickets that have been solved earlier and are above a set threshold for similarity score.
5. A Vizrt employee evaluates the list of similar tickets and gives feedback to the system so that it can retrain itself to improve for the next incoming query.

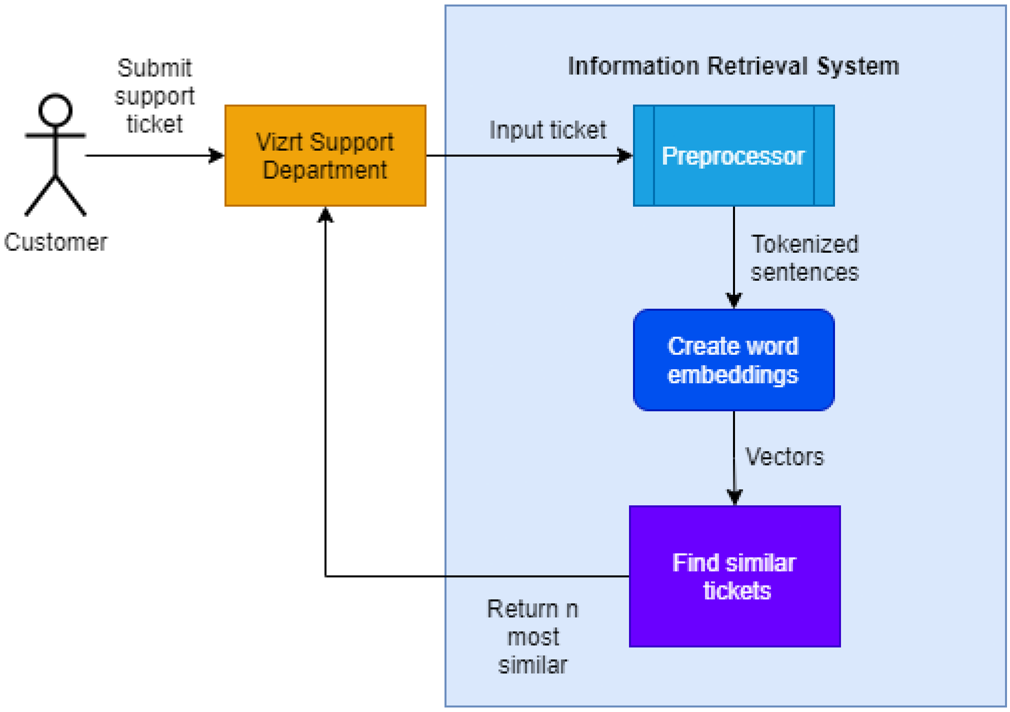


*Figure 1. Initial solution idea.*

## 2.2 Literature background

There is a large amount of relevant research available on the topics of machine learning and natural language processing, dating back to the middle of the 19th century (Turing, 1950).

Information retrieval is the process of retrieving relevant information from unstructured data. It is a wide term that is greatly explained in a 2009 research paper from Cambridge UP (Manning, Raghavan, & Schütze, 2009). This is an important subject in this thesis as the goal is to retrieve the contextual meaning from incoming queries automatically.

Word embeddings are representations of words that allow semantically similar words to have a similar representation. A study proposed by scientists at Google shows an efficient way of doing word embeddings by turning them into vectors (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013). There is another study by Google that builds upon this, but instead of doing word embeddings it does the embedding on a document level (Le & Mikolov, 2014). These are relevant resources for this project as the data supplied by Vizrt is textual data that would need to be transformed to a numerical representation.

Another study by Yang et al. proposes an approach that combines information retrieval with word embeddings to find similar bug reports (Yang, Lo, Xia, Bao, & Sun, 2016). Their approach showed promising results that even outperformed an existing state-of-the-art similar bug recommendation system, *NextBug*. This paper is highly relevant as their goal and motivation are quite like our objective mentioned in chapter 1.1.

# 3 PROJECT DESIGN

This chapter elaborates and discuss possible approaches to solve the task and based on this select an approach to move further with. It will also provide an overview of the tools and programming languages that will be used before it explains the development method. Lastly, the evaluation method will be presented.

## 3.1 Possible approaches

There are several ways to approach the problem. We must try different information retrieval and word embedding techniques against each other to determine which setup gives us the most accurate results in terms of recommending semantically similar support tickets.

### 3.1.1 Embedding techniques

To be able to use words in machine learning models the words must be presented in a numerical form, often in the form of a vector. There are several different techniques to achieve this, and we must do a many-to-many test with the word embeddings and the different methods of calculating semantic similarity to ensure we are using the optimal solution.

#### 3.1.1.1 Embedding alternative 1 – Word2Vec

Word2Vec is a popular technique used in natural language processing that is efficient of estimating word representations in vector space (Mikolov, Chen, Corrado, & Dean, 2013). There are two algorithms that can be used within Word2Vec to calculate these vectors, Skip-Gram and “Continuous bag of words”. Skip-Gram would be the most obvious choice for this project as it works well with small datasets and is good at representing less frequent words (Riva, 2021). We would also have to look at the possibility of using a pre-defined set of vectors and not just the ones derived from our training data to compare the performance.

#### 3.1.1.2 Embedding alternative 2 – Doc2Vec

Another technique that is more relevant for finding embeddings for whole documents rather than words is Doc2Vec (Le & Mikolov, 2014). It is heavily based on the Word2Vec technique but will hold a vector for each document as well as a vector for each word in the document. This is a promising approach for our case as we are looking for similarity on a document level.

#### 3.1.1.3 Embedding alternative 3 – BERT

*Bidirectional Encoder Representation from Transformers* (BERT) is a modern framework for natural language processing (Devlin, Chang, Lee, & Toutanova, 2019). It utilizes transformers that is a type of multi-layered neural network architecture (Vaswani, et al., 2017). Selecting this approach would require using an already trained model that is intended for semantic similarity, such as *stsb-mpnet-base-v2* (Pretrained Models, 2021).

#### 3.1.1.4 Embedding alternative 4 – TF-IDF and combining embedding techniques

*Term Frequency-Inverse Document Frequency* (TF-IDF) is a statistical measure that calculates in which degree a word is helpful for distinguishing different documents from another (Manning, Raghavan, & Schütze, 2009). It is done by checking how often a word is present in a single document and how many documents it is included in. If a word occurs several times in one document, but rarely occurs in others it will be given a high score. This way we can weight non-determining words with a low score even if it occurs often.

The scores from TF-IDF can be combined with the word embeddings (i.e., Word2Vec or Doc2Vec) to avoid that our similarity calculation will include words that does not contribute to the distinguishing of documents.

### 3.1.2 Techniques for measuring semantic similarity

Text that has been processed by word embedding techniques is on a numerical form that can be processed by mathematically based calculations to find the semantic similarity.

There are several techniques to do this calculation and we would evaluate the different approaches.

#### 3.1.2.1 Similarity measure alternative 1 – Cosine Similarity

Cosine similarity is a metric that is used to determine how close two vectors appear in the vector space. It uses the angle between vectors to calculate the similarity where a small angle gives a higher similarity score. It is calculated by the formula below:

$$similarity = \cos(\theta) = \frac{\mathrm{A} \cdot \mathrm{B}}{|\mathrm{A}||\mathrm{B}|}$$

#### 3.1.2.2 Similarity measure alternative 2 – Word Mover's Distance (WMD)

WMD is another metric that instead of measuring angles of vectors it measures the minimum distance one must travel in vector space from one word vector to reach another word vector (Kusner, Sun, Kolkin, & Weinberger, 2015). A short distance in vector space indicates that the words are similar. The distance between two vectors is calculated by the following minimization problem:

$$\min_{T \geq 0} \sum_{i,j=1}^{n} T_{ij}\, c(i,j)$$

*Subject to*

$$\sum_{j=1}^{n} T_{ij} = d_i \ \forall i \in \{1, \dots, n\}$$

$$\sum_{i=1}^{n} T_{ij} = d'_j \ \forall j \in \{1, \dots, n\}$$

#### 3.1.2.3 Similarity measure alternative 3 – Combination

A possible approach that can be considered is combining the techniques above. This will give us a total score that could be used to determine the most similar documents in our dataset. An easy way of solving this would be to simply take the average of the two measurements.

### 3.1.3 Discussion of alternative approaches

There had to be done a lot of exploration and testing on each approach, as well as not excluding other promising approaches referred in research that came up during the development process. We also had to experiment with trying different combination of techniques before we could confidentially conclude on the best approach.

## 3.2 Selected approach

The process of selecting the best approach has been a major part of this thesis as there was a lot of experimenting to be done. We ended up selecting Doc2Vec as the embedding technique and the similarity measurement done by WMD. We will elaborate on the detailed design of the model in chapter 4, as well as more details about how the different algorithms and models were compared to conclude on the approach in chapter 0.

## 3.3 Selection of tools and programming languages

For this project, the task will be solved in the programming language Python, using the development environment Jupyter Notebooks. Because of confidentiality reasons, the data to be worked on may never leave the AWS Windows Server provided by Vizrt. An added benefit of using this server is the ability to quickly scale processing power and memory if needed.

### 3.3.1 Python

Python is a general-purpose and object-oriented coding language and was first released in 1991 designed by Guido van Rossum. Python offers concise and readable code, and with its rich technology stack that offers a variety of libraries, it has become a popular programming language for machine learning.

We chose to use Python as our programming language based on available documentation and machine learning libraries that was relevant for our project.

### 3.3.2 Jupyter Notebook

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations and visualizations and narrative text. Uses include data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more (Project Jupyter, 2021).

We were already familiar with Jupyter Notebook from earlier courses but chose it for several reasons. It is lightweight and user-friendly making it easier to set up, but also for our client to understand what we have done using charts, diagrams and other visualisations that Jupyter Notebook offers. It also allows us to write markdown text in-between code which is helpful for separating different code blocks as well as describing what they do.

### 3.3.3 AWS Windows Server

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment (Amazon Web Services, 2021).

Our server was provided by the client for security reasons regarding the confidential data that we needed access to. It also provided us with enough processing power for working with the data and for training our machine learning model.

## 3.4 Project development method

There are many flexible methods that give a team of developers a good workflow. These methods give developers the ability to respond quickly to changes and deal with uncertainties immediately. The methods aim to improve the quality of the product, while at the same time offering a good workflow for the team.

### 3.4.1 Development method

The development for this project was carried out in an agile manner, using small iterations to quickly develop a minimum viable project (MVP). This made it easier to perform continuous evaluation with the project owner, in addition to better risk management.

A Kanban board was used to keep track of tasks that need to be handled. A Kanban board consists of tasks to be done in the sprint, with an overview of *product backlog* (which tasks the product is missing), *sprint backlog* (what is to be done in the current sprint), *to-do* (the next thing to be done), *on-going* (what is being done now and by whom) and *done* (what has been done so far in the project).

### 3.4.2 Project Plan

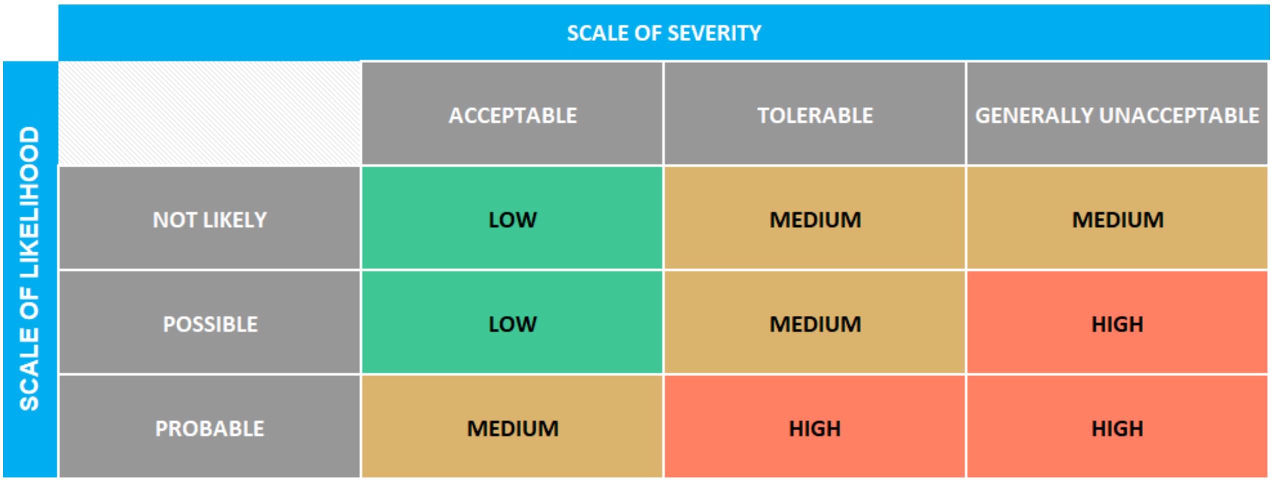
The group has chosen to use a Gantt chart to plan the work on the bachelor thesis. The plan is divided into a planning phase and a development phase. After having an initial meeting with the project owner in week 9, work on the project was set to begin in week 11. The planning phase consists of getting an overview of the task, including literature search and deciding on an approach. The milestone for this phase is to come up with a shortlist of different possible solutions to satisfy the specification requirements.

The next phase is development, where exploration is done towards the milestone goal of selecting a final model to work with. Furthermore, the plan is to improve and tune the selected model. Evaluation occurs throughout the development phase.

Chapter 9.2 in the appendix shows the progress plan for the work that started in week 9 and is scheduled to be completed in week 24.

### 3.4.3 Risk management

Risk management is an important topic to consider when planning a new project. It is done to make everyone involved aware of the risks that may arise during the project, potentially saving time, resources and other hazards that may harm the projects progress. The complete risk analysis can be found in chapter 9.1 in the appendix. It consists of activities that may cause dangers, its probability for it to occur and the severity. Together, these two factors multiplied represent the overall risk factor shown in Figure 2 below.



| | | SCALE OF SEVERITY | | |
|---|---|---|---|---|
| | | ACCEPTABLE | TOLERABLE | GENERALLY UNACCEPTABLE |
| SCALE OF LIKELIHOOD | NOT LIKELY | LOW | MEDIUM | MEDIUM |
| | POSSIBLE | LOW | MEDIUM | HIGH |
| | PROBABLE | MEDIUM | HIGH | HIGH |

*Figure 2. Risk assessment matrix*

## 3.5 Evaluation method

Evaluating the results proved to be one of the biggest challenges for this project. The data had no labels, which made it difficult to validate the results produced by the various models. Hence, it was necessary to devise an evaluation method that did not rely solely on WMD or cosine similarity to determine how well the model performed. A continuous evaluation was performed during the development phase to ensure that a good model would be selected for the project.

The end results of the project were evaluated together with Vizrt to determine if using a word embedding model was viable for their needs.

# 4 DETAILED DESIGN

This chapter provides a detailed description of the component's architecture and how its built.  It will describe the process, which resources that have been utilized and the usage and implementation of the component.

## 4.1 Frameworks and Libraries

This project has been dependent on several resources that either made the development process possible or at least made it easier. In this subchapter we will elaborate on the most used and significant ones.

### 4.1.1 Pandas

*Pandas* is an open-source library for manipulating data in Python (pandas, 2021). It includes methods of transforming different data structures such as XML and CSV-files into two-dimensional labelled data structures called dataframes. It also includes operations to manipulate these dataframes.

### 4.1.2 Natural Language Toolkit

Natural Language Toolkit or *nltk* provides helpful functions for developing the pre-processing pipeline (NLTK 3.6.2 documentation, 2021). The library includes functions for tokenizing, stemming and lemmatizing text as well as a corpus of stop words in the English language.

### 4.1.3 Gensim

*Gensim* is a library that contains methods of representing textual documents as semantic vectors (What is Gensim?, 2021). It includes algorithms such as Word2Vec, Doc2Vec and more. These are unsupervised algorithms that only require plain text and no labels.

### 4.1.4 Scikit-learn

*Scikit-learn* is a package that consist of several useful tools for machine learning (scikit-learn 0.24.2 documentation, 2021). Scikit-learn has relevant methods for this project such as TF-IDF vectorizing text and the creation of pipelines.

### 4.1.5 Google Translate

Google Translate is utilized by using Google's official API. This is done by sending queries as HTTP requests to the API, which returns the translated text.

### 4.1.6 SymSpell

SymSpell is a library consisting of algorithms that is used for correcting spelling errors (SymSpell, 2021). It uses a dictionary lookup to suggest alternative spellings that is within an error rate.

### 4.1.7 Others

*Numpy* is a library that provides useful data structures such as multi-dimensional arrays and matrices. It also includes functions and operations to be used on other data structures.

## 4.2 Architecture

The architecture of the system is built upon different layers that provides functionalities. There is a distinct separation of each layer with the goal of developing a maintainable and dynamic solution.

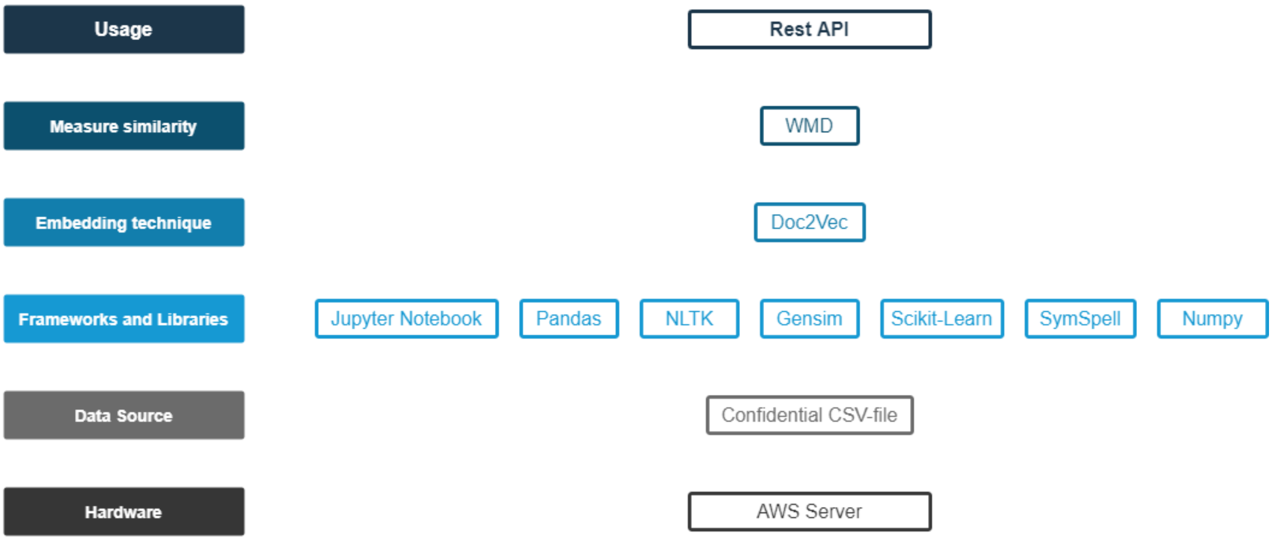


*Figure 3. Layered architecture*

Figure 3 is a representation of the architecture, which consists of six layers:

***Hardware*** – This layer provides computational power and storage for the whole system. For this task we chose to use an AWS Server for security and scalability.

***Data Source*** – Contains the provided data for our project. This data is confidential and was transferred directly to the AWS Server for security reasons. The data is stored in a CSV file.

***Framework and Libraries*** – Provides all tools and resources that are needed when working with machine learning.

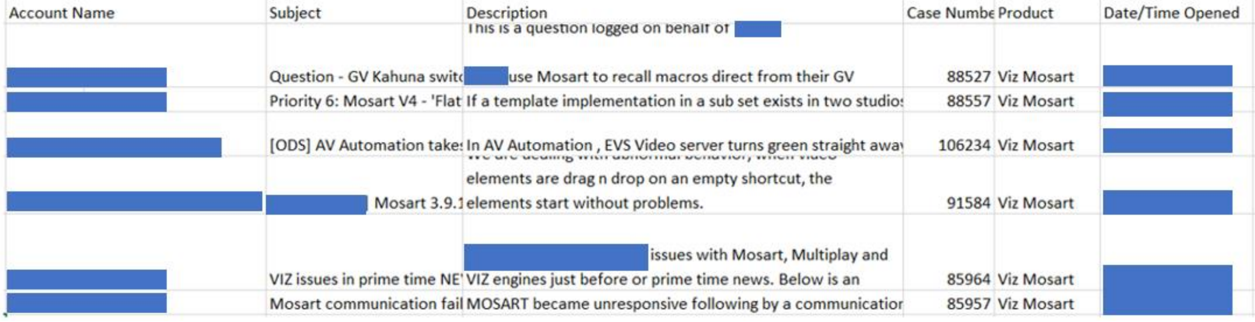
***Embedding technique*** – This layer contains the actual model to be trained on the given data.

***Measure similarity*** – A layer that contains measuring techniques to be used when outputting results to the user of the system.

***Usage*** – Contains the chosen solution for usage of the system. The final goal is an implementation through a Restful Web Service (REST API).

## 4.3 Import of data

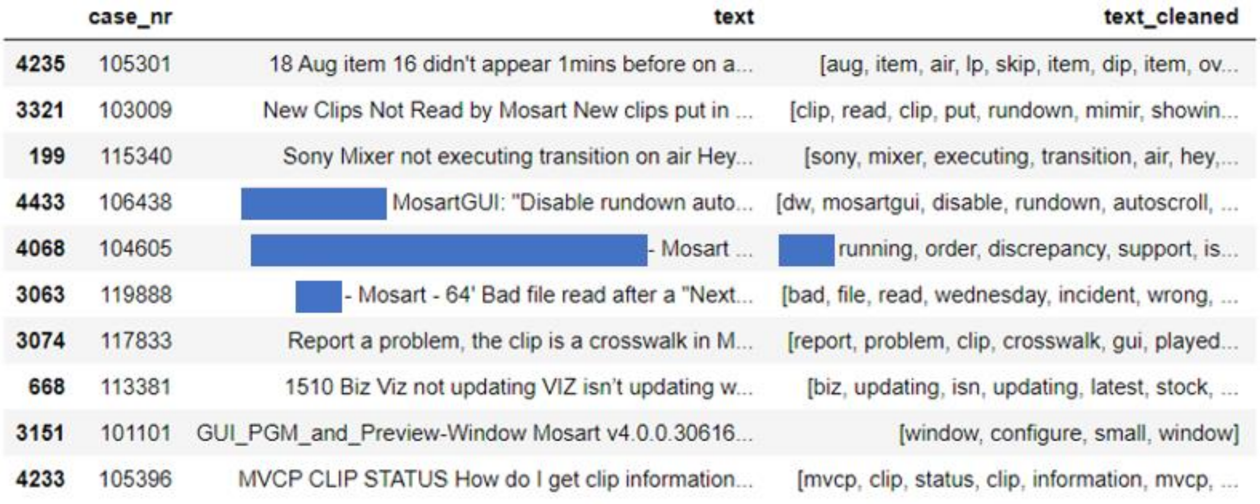
The provided data was originally a CSV-file (Comma-separated values) where the data is stored as plain text. The *pandas* library was utilized to read the CSV-file and transform it into a dataframe. It was also used for manipulating the dataframes to the desired format and content.



| Account Name | Subject | Description | Case Numbe | Product | Date/Time Opened |
|---|---|---|---|---|---|
| [redacted] | Question - GV Kahuna switc | This is a question logged on behalf of [redacted]<br>[redacted] use Mosart to recall macros direct from their GV | 88527 | Viz Mosart | [redacted] |
| [redacted] | Priority 6: Mosart V4 - 'Flat | If a template implementation in a sub set exists in two studios | 88557 | Viz Mosart | [redacted] |
| [redacted] | [ODS] AV Automation takes | In AV Automation , EVS Video server turns green straight away | 106234 | Viz Mosart | [redacted] |
| [redacted] | [redacted] Mosart 3.9.1 | We are dealing with abnormal behavior, when video<br>elements are drag n drop on an empty shortcut, the<br>elements start without problems. | 91584 | Viz Mosart | [redacted] |
| [redacted] | VIZ issues in prime time NE' | [redacted] issues with Mosart, Multiplay and<br>VIZ engines just before or prime time news. Below is an | 85964 | Viz Mosart | [redacted] |
| [redacted] | Mosart communication fail | MOSART became unresponsive following by a communicatior | 85957 | Viz Mosart | |

*Figure 4. Unprocessed data. Customer names and dates are hidden for confidentiality reasons.*

Figure 4 shows the unprocessed data. We found that the most valuable data fields for our task were Subject, Description and Case Number. The subject gives us concise information about reasoning for the ticket, the description provides in depth information and important words for our model, and the case number is important for traceability and identification of the ticket. Therefore, these three data fields were extracted to a dataframe. Figure 5 shows the data after extracting and preprocessing.



| | case_nr | text | text_cleaned |
|---|---|---|---|
| 4235 | 105301 | 18 Aug item 16 didn't appear 1mins before on a... | [aug, item, air, lp, skip, item, dip, item, ov... |
| 3321 | 103009 | New Clips Not Read by Mosart New clips put in ... | [clip, read, clip, put, rundown, mimir, showin... |
| 199 | 115340 | Sony Mixer not executing transition on air Hey... | [sony, mixer, executing, transition, air, hey,... |
| 4433 | 106438 | [redacted] MosartGUI: "Disable rundown auto... | [dw, mosartgui, disable, rundown, autoscroll, ... |
| 4068 | 104605 | [redacted] - Mosart ... | [redacted] running, order, discrepancy, support, is... |
| 3063 | 119888 | [redacted] - Mosart - 64' Bad file read after a "Next... | [bad, file, read, wednesday, incident, wrong, ... |
| 3074 | 117833 | Report a problem, the clip is a crosswalk in M... | [report, problem, clip, crosswalk, gui, played... |
| 668 | 113381 | 1510 Biz Viz not updating VIZ isn't updating w... | [biz, updating, isn, updating, latest, stock, ... |
| 3151 | 101101 | GUI_PGM_and_Preview-Window Mosart v4.0.0.30616... | [window, configure, small, window] |
| 4233 | 105396 | MVCP CLIP STATUS How do I get clip information... | [mvcp, clip, status, clip, information, mvcp, ... |

*Figure 5. Data extracted and preprocessed. Customer names are hidden for confidentiality reasons.*

## 4.4 Preprocessing of data

The provided data contains unstructured tickets with variations in length, structure, type of inquiry and languages.

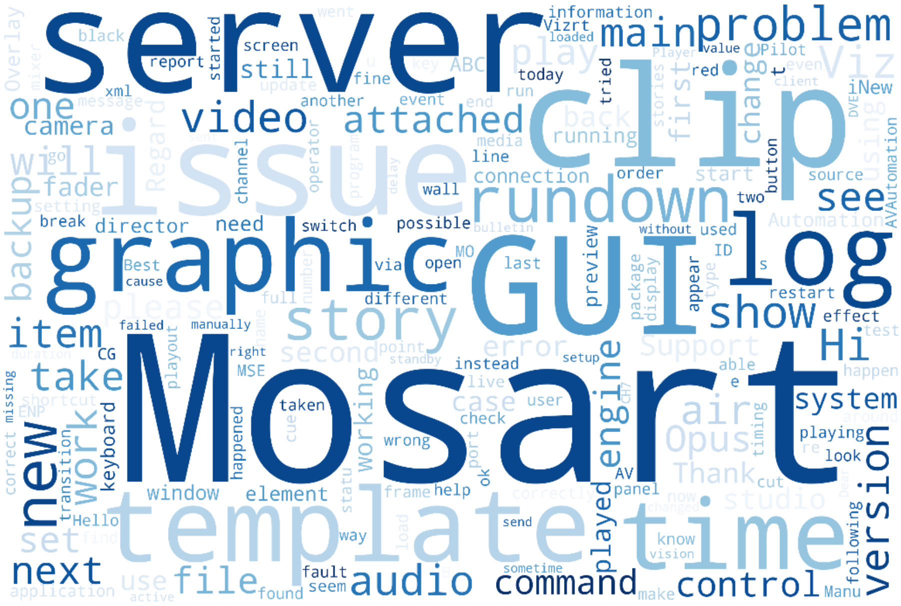


*Figure 6. Word cloud before preprocessing. The size of the words relates to the frequency of occurrence in the data.*

Figure 6 illustrates the most common words in the data. To prepare the data before it can be used to train the model, it must be preprocessed using different techniques:

***Translating non-English text*** – Due to clients from all around the world, some support tickets are written in other languages than English. This caused problems since all tickets that were written in the same language were evaluated by the model as similar, because they contained similar words. By using Google Translate official API all tickets were translated before further preprocessing.

***Lowercasing*** – All words were converted to lowercase to reduce the number of total words in the dataset, since capital letters rarely affects the meaning of the word.

***Remove non-alphabetical characters*** – The support tickets often contain a lot of numbers and other non-alphabetical characters that brings along more noise than crucial information.

***Spelling correction*** – Since the support tickets are written manually, misspelling will occur. By applying spelling correction using the SymSpell library we managed to reduce the number of unique words by approximately 3000. Analyzing the data after applying spelling correction, we found that ignoring words with four characters or less gives better results since the data contains a lot of abbreviations which were wrongly corrected. We also created a list of domain-specific words for the spelling corrector to ignore due to several observations of mis corrections.

***Stop words*** – Stop words are words that adds little to no information, such as “it”, “a”, “the” and so on. The Natural Language Toolkit provides a generic list of these words which was used to remove from the tickets, effectively reducing noise in the text. We also added our own domain-specific words to the list.

***Lemmatization*** – Lemmatization is a word-processing technique that considers the context of a word and converts it to its base form. For example, it converts the word “running” to “run” that helps us reducing the total number of unique words. This was done by using Natural Language Toolkit.

***Tokenizing*** – To prepare the data as input for machine learning models, all words are split into separate tokens using Natural Language Toolkit. These tokens are then used to prepare the vocabulary for the chosen machine learning model to use.

All the listed techniques combined creates an efficient pipeline for preprocessing of our data and gives a solid ground for the machine learning model to be trained on.

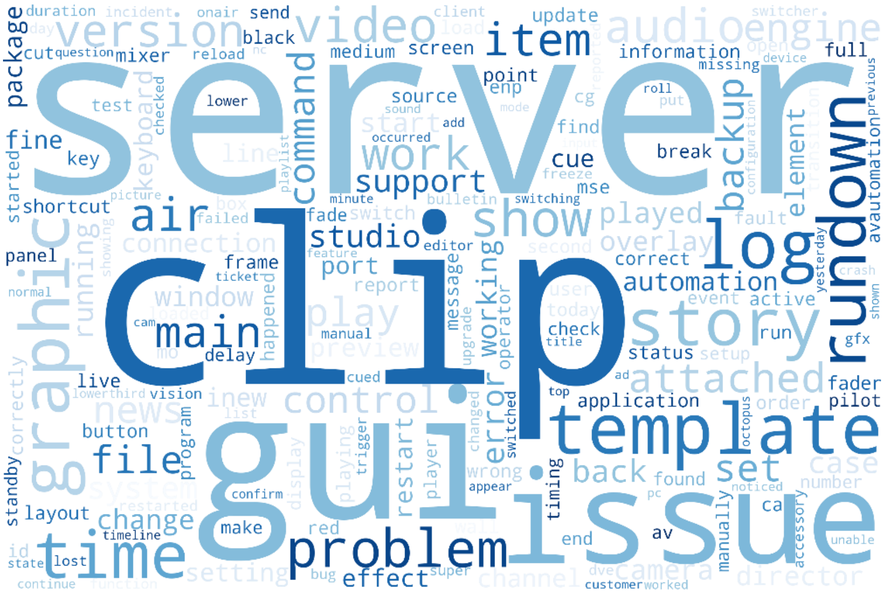


*Figure 7. Word cloud after preprocessing. The size of the words relates to the frequency of occurrence in the data.*

Figure 7 illustrates the most common words after preprocessing. By comparing it to Figure 6 there is a clear difference in the selection of meaningful words.

## 4.5 Training the model

As mentioned in chapter 3.2 - Selected approach we have chosen Doc2Vec as our preferred type of machine learning model. Doc2Vec is based on the Word2Vec technique but will hold a vector for each document as well as a vector for each word in the document.

When instantiating a Doc2Vec model there are two different algorithms to choose between: Distributed Memory (DM) and Distributed Bag of Words (DBOW). We chose Distributed Bag of Words, and the reason for this is further described in chapter 5.2.1 - Model selection results. Distributed Bag of Words is an embedding approach that:

> […] ignore the context words in the input but force the model to predict words randomly sampled from the paragraph in the output. In reality, what this means is that at each iteration of stochastic gradient descent, we sample a text window, then sample a random word from the text window and form a classification task given the Paragraph Vector. (Le & Mikolov, 2014).

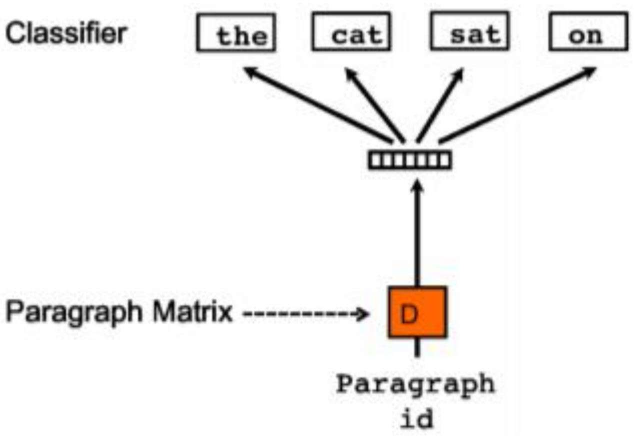


*Figure 8. Distributed Bag of Words technique (Le & Mikolov, 2014)*

Before the training starts, the data is split into a training and a test set. The reasoning behind this is that the model is only trained on the training set, which makes testing of the model more realistic as the model has not seen the test data before. The training data is then loaded to the model.

When initiating a Doc2vec model there are multiple parameters to tweak to ensure the best possible fit for the data. These are optional, but crucial for getting good results:

***Window*** – This is the maximum distance between the current and predicted word within a sentence. This means that having a wider window will cause more words being related to each other, and a narrower window the opposite. The default window size of 5 was chosen since it gave us the best results overall.

***Epochs*** – The number of iterations over the data during training. Too many iterations may cause overfitting, which means feeding the model more data than necessary catching noisy data. On the other hand, too few iterations and lack of training may cause a poor model. Considering our relatively small dataset, we chose 100 epochs after several tests.

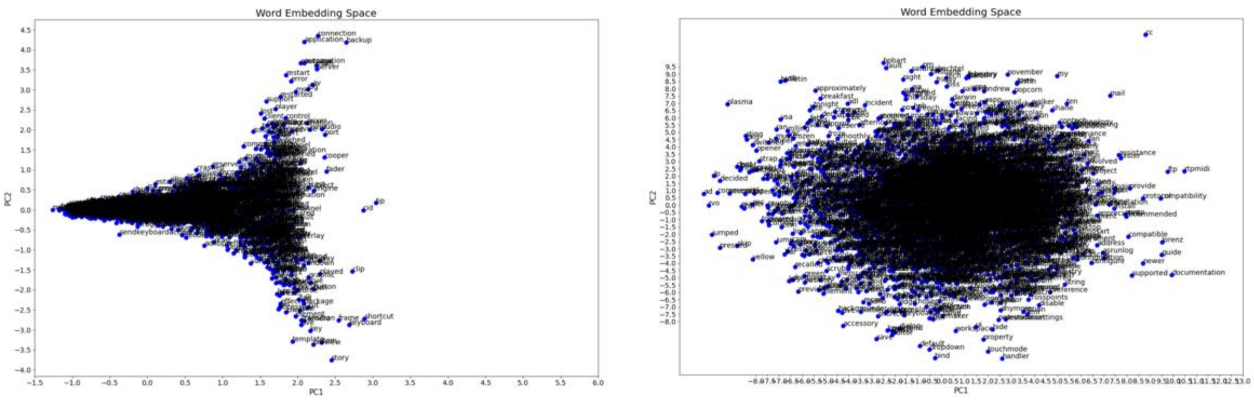


*Figure 9. A Principal Component Analysis of the vocabulary after 10 and 100 epochs*

***Min_count*** – Skips all words with total frequency lower than this number. This was set to 5 to reduce irrelevant words and noise in the text.

***Vector_size*** – Dimensionality of the feature vectors, meaning each document being mapped to a point in an n-dimensional space. According to research done by TensorFlow Team (TensorFlow Team, 2017) they claim that embedding vector size should be the 4th root of the number of categories. Based on this and several tests we found that a vector size of 12 gave the best results.

Once these parameters are decided along with several more, the model can now be trained on the data.

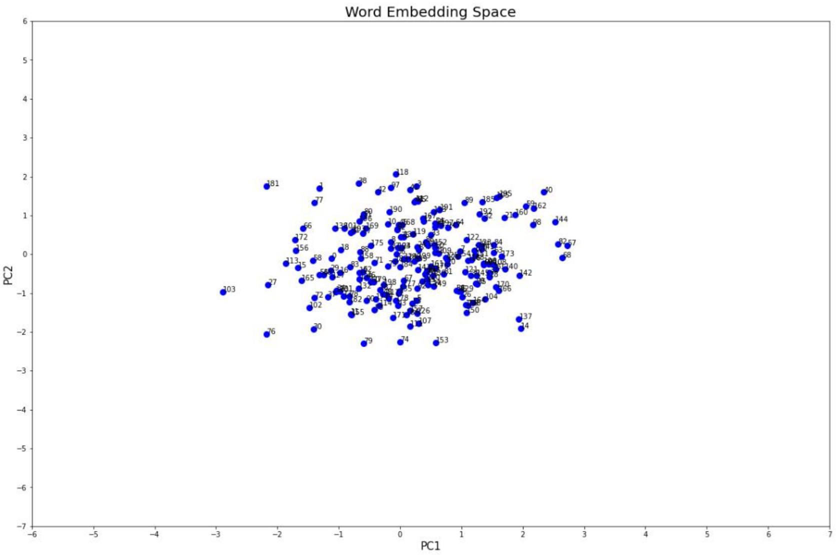


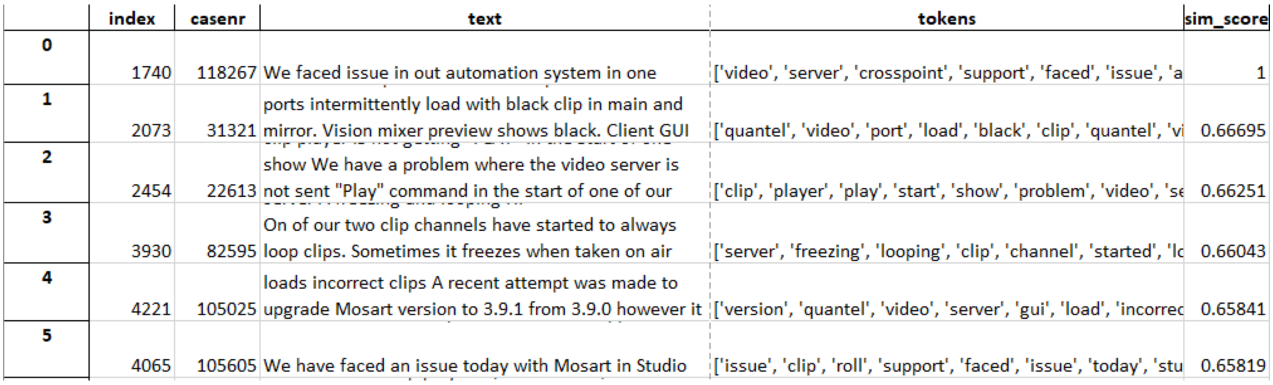
*Figure 10 - A Principal Component Analysis (PCA) showing the first 200 support tickets after training*

Figure 10 illustrates how the first 200 tickets will look in a two-dimensional vector space after training. The distance between two points tells how semantically similar they are. When a new incoming ticket is presented to the model, it will be transformed to a vector and return the five closest vectors in the vector space prior to this.

## 4.6 Usage and implementation of the software

The software is to be used as an independent module. At first, Vizrt will experiment by using the model manually with copy and pasting the queries for input and review the output. If the results satisfy the expectations, the next step will be to integrate Vizrt's internal system with the software through a web service.

When Vizrt receives a new support ticket, they can either manually or through an integrated web service forward the ticket to the model. The model will then process the ticket and output the five most similar tickets.



| | index | casenr | text | tokens | sim_score |
|---|---|---|---|---|---|
| 0 | 1740 | 118267 | We faced issue in out automation system in one | ['video', 'server', 'crosspoint', 'support', 'faced', 'issue', 'a | 1 |
| 1 | 2073 | 31321 | ports intermittently load with black clip in main and mirror. Vision mixer preview shows black. Client GUI | ['quantel', 'video', 'port', 'load', 'black', 'clip', 'quantel', 'vi | 0.66695 |
| 2 | 2454 | 22613 | show We have a problem where the video server is not sent "Play" command in the start of one of our | ['clip', 'player', 'play', 'start', 'show', 'problem', 'video', 'se | 0.66251 |
| 3 | 3930 | 82595 | On of our two clip channels have started to always loop clips. Sometimes it freezes when taken on air | ['server', 'freezing', 'looping', 'clip', 'channel', 'started', 'lc | 0.66043 |
| 4 | 4221 | 105025 | loads incorrect clips A recent attempt was made to upgrade Mosart version to 3.9.1 from 3.9.0 however it | ['version', 'quantel', 'video', 'server', 'gui', 'load', 'incorrec | 0.65841 |
| 5 | 4065 | 105605 | We have faced an issue today with Mosart in Studio | ['issue', 'clip', 'roll', 'support', 'faced', 'issue', 'today', 'stu | 0.65819 |

*Figure 11. Example of output from the model*

Figure 11 shows how the output can look. The first row shows the ticket itself that was sent in, while the next rows are the five most similar tickets that have been solved before. *Sim_score* indicates the similarity score compared to the incoming ticket rating from 0 to 1, where 0 is the worst possible score and 1 being the best possible score. *Text* displays the ticket's original and unprocessed text, while *tokens* show the different tokens after preprocessing to give an indication of the content in the ticket.

The user of the software can use this information to evaluate the results and do further investigation using the *casenr* to trace the chosen cases in Vizrt's internal system.

# 5 EVALUATIONS

Literature suggests that cosine similarity and Word Mover's Distance (WMD) are good metrics for determining semantic textual similarity between texts (Sitikhu, Pahi, Thapa, & Shakya, 2019). However, they do not necessarily provide good results for this highly domain specific task. Although we may end up with high similarity scores for documents, we may still end up with a poor result because of the nature of the two evaluations metrics. This led to a custom evaluation method being devised.

## 5.1 Evaluation method

The evaluation method for this project consists of two main parts. The first part is a continuous evaluation with the external supervisor to ensure that the best performing embedding model was selected. It was necessary to evaluate and compare several word embeddings models, hyperparameters and evaluation measures against each other. For this project, a large part revolved around selecting a word embedding model, as well as determining which one of the two similarity scores gave the better results. Together with Nils from Vizrt, we came up with an evaluation method that would ensure qualitatively better results than what would be achieved by only using cosine similarity or WMD. Evaluation was performed in the following way.

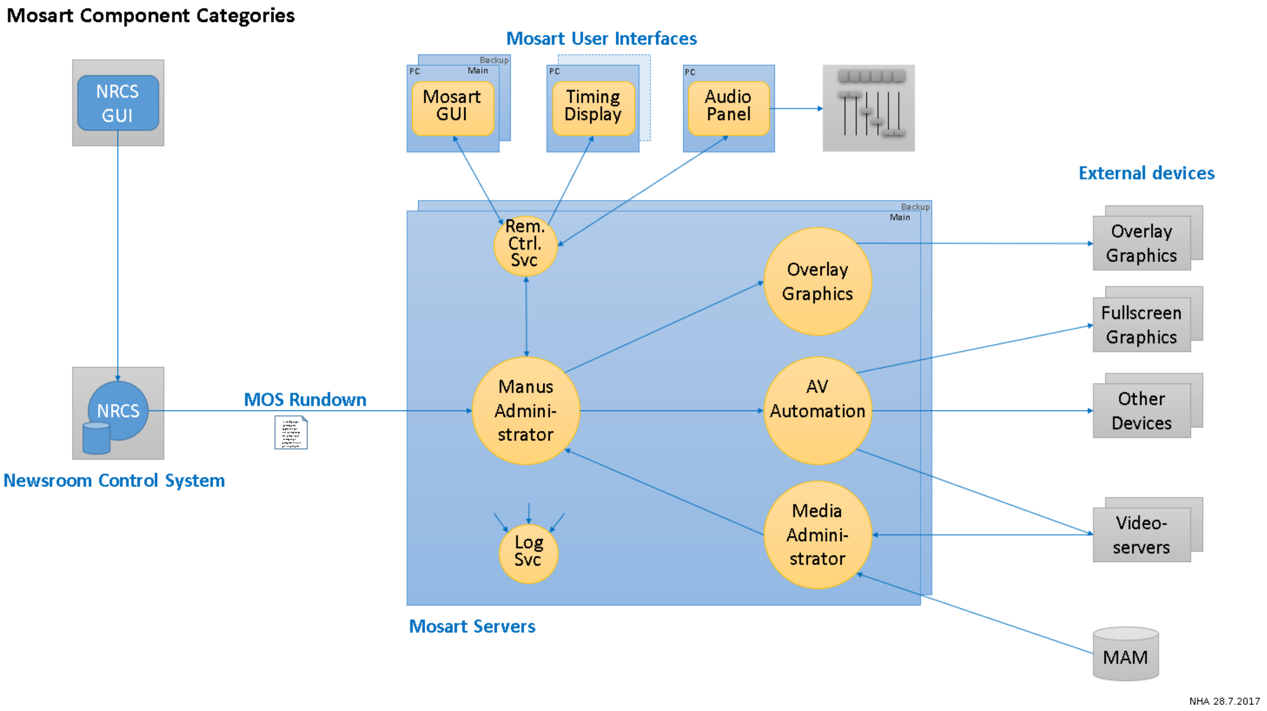
Five query documents of relatively recent dates were randomly selected. These were considered as our test set. In addition, a selection of embedding models and similarity metrics were chosen based on recent and relevant research:

**Embedding models:** Word2Vec, Doc2Vec, TF-IDF, BERT

**Evaluation metrics:** WMD and Cosine Similarity

For each document in the test set, each model would retrieve the *n* most similar documents. For each model, the five query documents and their *n* most similar documents, including their respective similarity score, were written to disk for Nils to perform manual evaluation. The evaluation used two scores:

**Area Score (0 – 10):** A score describing how well the model can recommend similar documents that fall within the same area, component or data stream. Examples of area can be video control flow from photographer, via editing, to producer who plays it out through Viz Mosart. It could also be a specific third-party device, such as a video server or audio mixer. Figure 12 shows an overview of the Viz Mosart architecture.



NHA 28.7.2017

*Figure 12. Overview of Mosart Component Categories.*

**Function Score (0 – 10):** Describes how well the model is able to recommend similar tickets based on their functional operations, such as init, start, stop, read or save.

For each query, we sum the total score for area and function respectively, and average it over the number of considered tickets *n*. The result is a score of 0-10 for each query. Finally, the scores for model each are summed together, where the total maximum for each of area and function is 50. The following formulas are used for determining scores:

**Average area score** $a_{avg}$ per query, across *n* potentially similar documents

$$a_{avg} = \frac{1}{\mathrm{n}} \sum_{i=1}^{n} a_i \, , a \in \{0,1, \dots ,10\}$$

**Total area score** $a_{tot}$, for all five queries

$$\mathrm{a_{tot}} = \sum_{\mathrm{i}=1}^{5} \mathrm{a_{avg_i}}$$

**Average functional score** $f_{avg}$ per query, across *n* potentially similar documents

$$\mathrm{f_{avg}} = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{f_i} \, , \mathrm{f} \in \{0,1, \dots ,10\}$$

**Total functional score** $f_{tot}$ for all five queries

$$f_{tot} = \sum_{\mathrm{i}=1}^{5} \mathrm{f_{avg_i}}$$

The results from this evaluation are presented in detail in the following section.

The second part of the evaluation process was done after a final model was selected and trained to a point where the results were as good as we could achieve with the available data. Vizrt performed a final qualitative evaluation manually to see if the model is able to produce similar documents when presented with a new query, and to decide if this project has reached its goals.

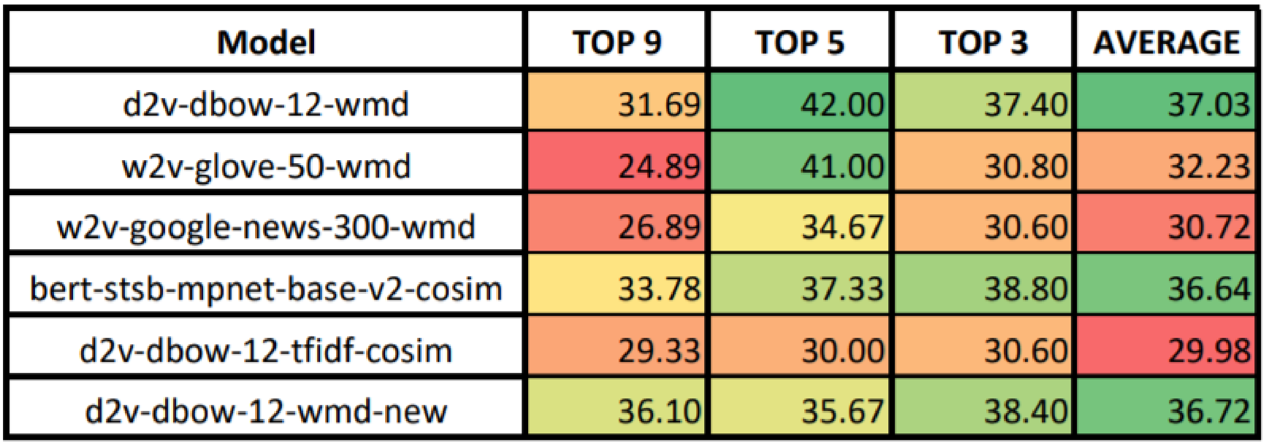
## 5.2 Evaluation results

This section will present the results generated by our two-part evaluation. The first section contains the results from the extensive model selection process, followed by the final evaluation results.

### 5.2.1 Model selection results

After having performed multiple evaluations on different setups, we found that the best performing model was Doc2Vec with embedding size 12 and training algorithm set to Distributed Bag of Words (DBOW) would be the best choice for Vizrt’s domain. Table 1 shows the obtained results from the final evaluation for model selection. The maximum total score for area and function combined is 100. From the results it is obvious that the selected model is far from achieving a full score. However, due to the limitations discussed in section 1.3, in addition to the chosen method of evaluation, comparably low scores are considered good enough to move further with.

*Table 1 - Average scores across top 9, top 5 and top 3 similar documents*

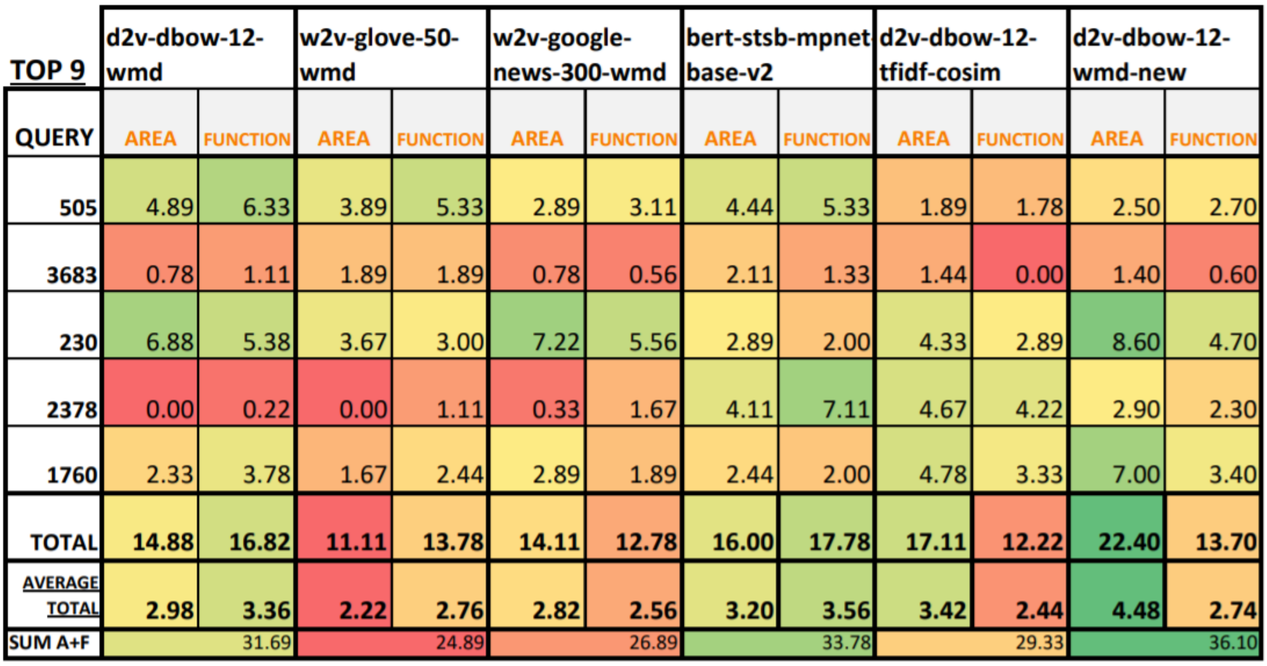


| Model | TOP 9 | TOP 5 | TOP 3 | AVERAGE |
|---|---|---|---|---|
| d2v-dbow-12-wmd | 31.69 | 42.00 | 37.40 | 37.03 |
| w2v-glove-50-wmd | 24.89 | 41.00 | 30.80 | 32.23 |
| w2v-google-news-300-wmd | 26.89 | 34.67 | 30.60 | 30.72 |
| bert-stsb-mpnet-base-v2-cosim | 33.78 | 37.33 | 38.80 | 36.64 |
| d2v-dbow-12-tfidf-cosim | 29.33 | 30.00 | 30.60 | 29.98 |
| d2v-dbow-12-wmd-new | 36.10 | 35.67 | 38.40 | 36.72 |

Although both Doc2Vec and BERT achieved good scores, it was decided that Doc2Vec would be the best model for this project because it is easier to use and implement compared to BERT.

Table 2 shows the obtained results when evaluated on top nine similar documents. The scores for query 3683 are notably low for most models, both in area and function. This could indicate that there are few existing tickets that are similar, but also that the model does a poor job of discovering semantic similarities between the query and existing documents. More results are included in chapter 9.4 of the appendix.

*Table 2. Model scores evaluated on top 9 similar documents.*



| TOP 9 | d2v-dbow-12-wmd | | w2v-glove-50-wmd | | w2v-google-news-300-wmd | | bert-stsb-mpnet base-v2 | | d2v-dbow-12-tfidf-cosim | | d2v-dbow-12-wmd-new | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QUERY | AREA | FUNCTION | AREA | FUNCTION | AREA | FUNCTION | AREA | FUNCTION | AREA | FUNCTION | AREA | FUNCTION |
| 505 | 4.89 | 6.33 | 3.89 | 5.33 | 2.89 | 3.11 | 4.44 | 5.33 | 1.89 | 1.78 | 2.50 | 2.70 |
| 3683 | 0.78 | 1.11 | 1.89 | 1.89 | 0.78 | 0.56 | 2.11 | 1.33 | 1.44 | 0.00 | 1.40 | 0.60 |
| 230 | 6.88 | 5.38 | 3.67 | 3.00 | 7.22 | 5.56 | 2.89 | 2.00 | 4.33 | 2.89 | 8.60 | 4.70 |
| 2378 | 0.00 | 0.22 | 0.00 | 1.11 | 0.33 | 1.67 | 4.11 | 7.11 | 4.67 | 4.22 | 2.90 | 2.30 |
| 1760 | 2.33 | 3.78 | 1.67 | 2.44 | 2.89 | 1.89 | 2.44 | 2.00 | 4.78 | 3.33 | 7.00 | 3.40 |
| TOTAL | 14.88 | 16.82 | 11.11 | 13.78 | 14.11 | 12.78 | 16.00 | 17.78 | 17.11 | 12.22 | 22.40 | 13.70 |
| AVERAGE TOTAL | 2.98 | 3.36 | 2.22 | 2.76 | 2.82 | 2.56 | 3.20 | 3.56 | 3.42 | 2.44 | 4.48 | 2.74 |
| SUM A+F | 31.69 | | 24.89 | | 26.89 | | 33.78 | | 29.33 | | 36.10 | |

### 5.2.2 Final evaluation results

Although a model has been selected after extensive tests, we may not be certain that it will perform well and stable enough for it to be used as intended. This must be concluded when testing it out in real-time, with it being integrated as a part of the workflow in the service departments. Furthermore, if it turns out that the chosen model is not a viable approach the project has still been beneficial for Vizrt with the research that has been done.

After evaluation from Vizrt, their initial fear of insufficient data may have been proven to be true. From the results, it is clear to see that some rows score well or poorly regardless of model. This suggests that the use of grammatically correct language is crucial. In general, they are left with the impression that there is too little data to train the model properly. They conclude that the thesis has shown the difficulty of using relatively small texts, relatively few cases and varying language use.

# 6 DISCUSSION

This chapter contains a discussion around the consequences of the chosen approaches on the obtained results and how they have been influenced by the choices made throughout the project. It will also discuss how the results could have been improved if the work were to be done again.

## 6.1 Consequences

This section will present the different consequences resulting from the chosen approaches and choices that have been made.

### 6.1.1 Limited training data for chosen model type

By choosing to train a machine learning model rather than a pretrained model, more data to work with usually means a greater chance of getting better results. The size of the available training data is limited and there is a risk of the model not generalizing well on new incoming data.

### 6.1.2 Weak basis for evaluation

Due to the data not having labels, in addition to the aforementioned lack of training data, it was difficult to verify the model's performance. The devised evaluation method was also prone to bias and error, due to having a single person perform the evaluation.

### 6.1.3 Lack of time

Due to lack of time, we have only experimented with a small selection of the potential models available in the field of semantic textual similarity and natural language processing. Therefore, we cannot confidently state that the selected model is the definite optimal solution.

## 6.2 Possible improvements

If the work on this project was to be done again, there are several other choices and improvements that could have been done. It would be highly beneficial to obtain more data to train the model on. This could for instance be done by scraping different manuals and instructions for Viz Mosart and the other software developed by Vizrt. Another improvement would be to build a ground truth, which in essence means to create a set of documents and give them labels that could be used for verification of the model's performance. It could also be a good idea to use the same, devised evaluation method, but have at least two people to evaluate independently to see if they give the same scores.

# 7 CONCLUSIONS AND FURTHER WORK

In this chapter we will conclude on the result of the project and give possible ways of improving the product if it is to be developed further.

## 7.1 Concluding on the goal of the project

The goal of this project was to develop a machine learning model that could find semantically similar support tickets that have been solved before when queried with an incoming support ticket. We are confident that we have obtained this goal as the chosen model is able to generalize unseen support ticket and suggest cases that are within the degree of similarity that is satisfying for Vizrt and their intended use-case.

## 7.2 Further work

Although the initial goal was obtained there is also several possible areas to improve upon if Vizrt wishes to continue developing this project in the future. Obtaining more domain specific data to continue training the chosen model would likely improve the results of the prediction. Another way of improving the models would be to build a ground truth for the test-data, although this would include some manual labour for Vizrt. Another possible way of further developing the product is by expanding it to fit the domains of other departments within Vizrt. Lastly, it would be possible to build an API or embedding the model within a graphical user interface that could be an independent component that could fit in the systems at Vizrt.

### 7.2.1 Obtaining additional domain specific data

Vizrt are receiving new support tickets that are to be solved on a regular basis, which results in continuous accumulation of more domain specific data that could be used to further train the model. This means that Vizrt could decide to continue to develop this project in the future when there is more training data available. It could also be possible to scrape additional data from internal resources that Vizrt have available, such as product manuals or reports within the issue tracking system.

### 7.2.2 Building a ground truth

By having a ground truth, which in essence would provide labels for some of the data, it would be easier to use well known validation and evaluation methods. In addition, fine-tuning the hyperparameters of the model would be more efficient and accurate if there were labels on some of the data. By having labels, even the slightest degree of improvements could be measured by using formulas for accuracy and precision.

### 7.2.3 Expand model to fit within other departments

Expanding the model to fit within other departments at Vizrt is a possible approach for further developing the project. This was suggested by Nils from Vizrt, as he could see the benefits from implementing a machine learning component in different parts of their system. This would not take too much effort as our defined components and functions are loose coupled and could fit easily fit other domains, given that there is enough data.

### 7.2.4 Embedding the model

It could be interesting to embed the machine learning component from this thesis into either Vizrt’s systems or into a web solution. This way the user could interact with the model to test out different parameters and alter the result given from the model, such as number of similar reports. Embedding the model could also allow the opportunity to further train the model by the user giving it feedback on its predictions, this could mean that the model would continuously improve while being in use.

# 8 REFERENCES

(2021). Hentet fra SymSpell: https://github.com/wolfgarbe/SymSpell

Amazon Web Services. (2021). *AWS Amazon*. Hentet fra https://aws.amazon.com/ec2/?ec2-whats-new.sort-by=item.additionalFields.postDateTime&ec2-whats-new.sort-order=desc

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Gensim. (2021). *Doc2vec paragraph embeddings*. Hentet fra https://radimrehurek.com/gensim/models/doc2vec.html

Halevy, A., Norvig, P., & Pereira, F. (2009). The Unreasonable Effectiveness of Data.

Kusner, M. J., Sun, Y., Kolkin, N. I., & Weinberger, K. Q. (2015). From Word Embeddings To Document Distances.

Le, Q., & Mikolov, T. (2014). Distributed Representations of Sentences and Documents. *Proceedings of the 31st International Conference on Machine Learning,*, ss. 1188-1196.

Manning, C. D., Raghavan, P., & Schütze, H. (2009). Introduction to Information Retrieval.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. Hentet fra Cornell University: https://arxiv.org/abs/1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). *Distributed Representations of Words and Phrases and their Compositionality.* arXiv.org.

*NLTK 3.6.2 documentation*. (2021). Hentet fra Natural Language Toolkit: https://www.nltk.org/

*pandas*. (2021). Hentet fra https://pandas.pydata.org/

*Pretrained Models*. (2021). Hentet fra Sentence-Transfomers documentation: https://www.sbert.net/docs/pretrained_models.html

Project Jupyter. (2021, April 2). *Jupyter*. Hentet fra https://jupyter.org/

Ranasinghe, T., Orasan, C., & Mitkov, R. (2019, September). Enhancing Unsupervised Sentence Similarity Methods with Deep Contextualised Word Representations. *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, ss. 994-1003.

Riva, M. (2021). *Word Embeddings: CBOW vs Skip-Gram*. Hentet fra Baeldung: https://www.baeldung.com/cs/word-embeddings-cbow-vs-skip-gram

Sanjeev Arora, Y. L. (2017). A Simple But Tough-To-Beat Baseline for Sentence Embeddings. Princeton University.

Schwaber, K., & Sutherland, J. (2017, November). *Scrumguides.* Hentet fra https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100

*scikit-learn 0.24.2 documentation*. (2021). Hentet fra scikit-learn: machine learning in Python: https://scikit-learn.org/stable/

Sitikhu, P., Pahi, K., Thapa, P., & Shakya, S. (2019). *A Comparison of Semantic Similarity Methods for Maxium Human Interpretability.*

*SymSpell*. (u.d.). Hentet fra wolfgarbe/SymSpell: https://github.com/wolfgarbe/SymSpell

TensorFlow Team. (2017, November 20). *Google Developers*. Hentet fra Introducing TensorFlow Feature Columns: https://developers.googleblog.com/2017/11/introducing-tensorflow-feature-columns.html

Turing, A. (1950). Computing Machinery And Intelligence.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention Is All You Need.

*Visual Paradigm*. (2020). Hentet fra https://www.visual-paradigm.com/scrum/what-is-sprint-in-scrum/

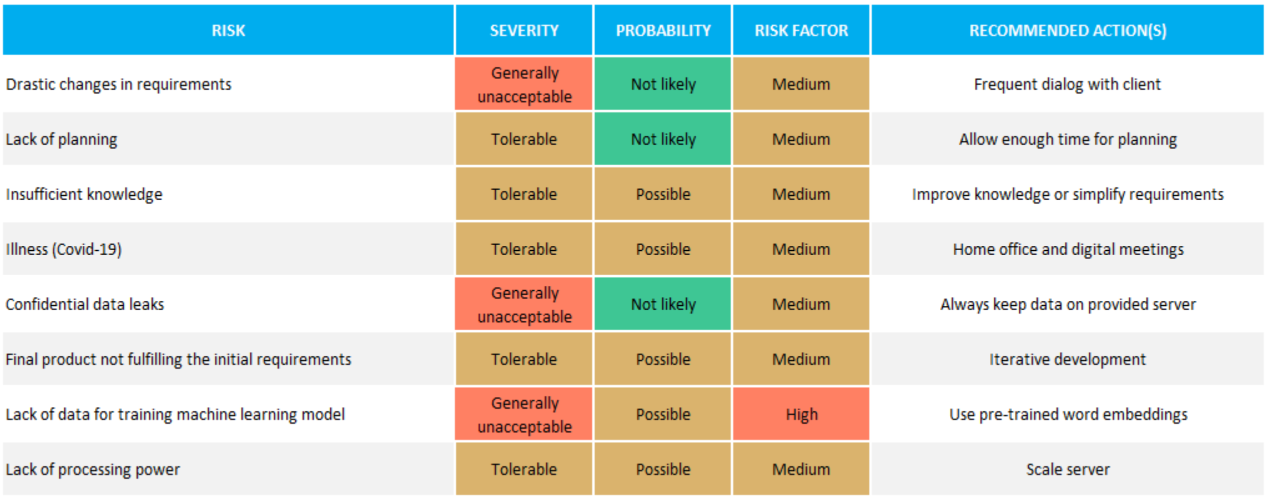
*What is Gensim?* (2021). Hentet fra gensim: https://radimrehurek.com/gensim/intro.html

Xu, B., Ye, D., Xing, Z., Xin, X., Chen, G., & Li, S. (2016, August). Predicting Semantically Linkable Knowledge in Developer Online Forums via Convolutional Neural Network. *ASE 2016: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, ss. 51-62.

Yang, X., Lo, D., Xia, X., Bao, L., & Sun, J. (2016). Combining Word Embedding with Information Retrieval to Recommend Similar Bug Reports. *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, ss. 127-137. doi:10.1109/ISSRE.2016.33

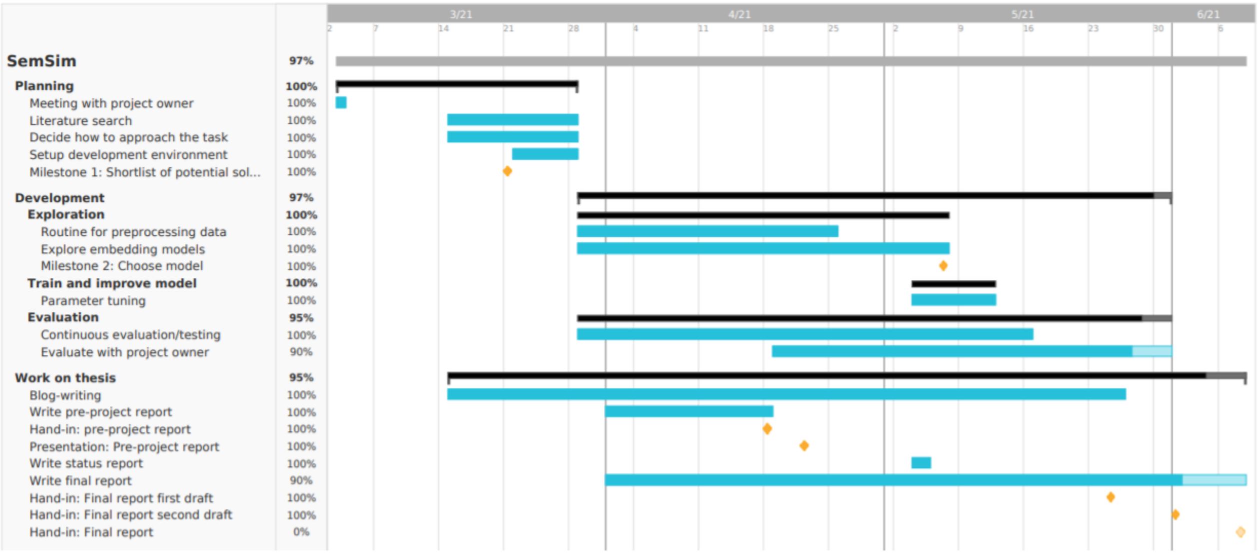
# 9 APPENDIX

## 9.1 Risk list



| RISK | SEVERITY | PROBABILITY | RISK FACTOR | RECOMMENDED ACTION(S) |
|---|---|---|---|---|
| Drastic changes in requirements | Generally unacceptable | Not likely | Medium | Frequent dialog with client |
| Lack of planning | Tolerable | Not likely | Medium | Allow enough time for planning |
| Insufficient knowledge | Tolerable | Possible | Medium | Improve knowledge or simplify requirements |
| Illness (Covid-19) | Tolerable | Possible | Medium | Home office and digital meetings |
| Confidential data leaks | Generally unacceptable | Not likely | Medium | Always keep data on provided server |
| Final product not fulfilling the initial requirements | Tolerable | Possible | Medium | Iterative development |
| Lack of data for training machine learning model | Generally unacceptable | Possible | High | Use pre-trained word embeddings |
| Lack of processing power | Tolerable | Possible | Medium | Scale server |

## 9.2 GANTT diagram



| | |
|---|---|
| **SemSim** | **97%** |
| **Planning** | **100%** |
| Meeting with project owner | 100% |
| Literature search | 100% |
| Decide how to approach the task | 100% |
| Setup development environment | 100% |
| Milestone 1: Shortlist of potential sol... | 100% |
| **Development** | **97%** |
| **Exploration** | **100%** |
| Routine for preprocessing data | 100% |
| Explore embedding models | 100% |
| Milestone 2: Choose model | 100% |
| **Train and improve model** | **100%** |
| Parameter tuning | 100% |
| **Evaluation** | **95%** |
| Continuous evaluation/testing | 100% |
| Evaluate with project owner | 90% |
| **Work on thesis** | **95%** |
| Blog-writing | 100% |
| Write pre-project report | 100% |
| Hand-in: pre-project report | 100% |
| Presentation: Pre-project report | 100% |
| Write status report | 100% |
| Write final report | 90% |
| Hand-in: Final report first draft | 100% |
| Hand-in: Final report second draft | 100% |
| Hand-in: Final report | 0% |

## 9.3 User manual

The Jupyter Notebook contains all the work done in this project and is left on the AWS server for Vizrt to keep. In order to reproduce the notebook:

1. Open Anaconda Prompt.
2. Navigate to the root directory of the project.

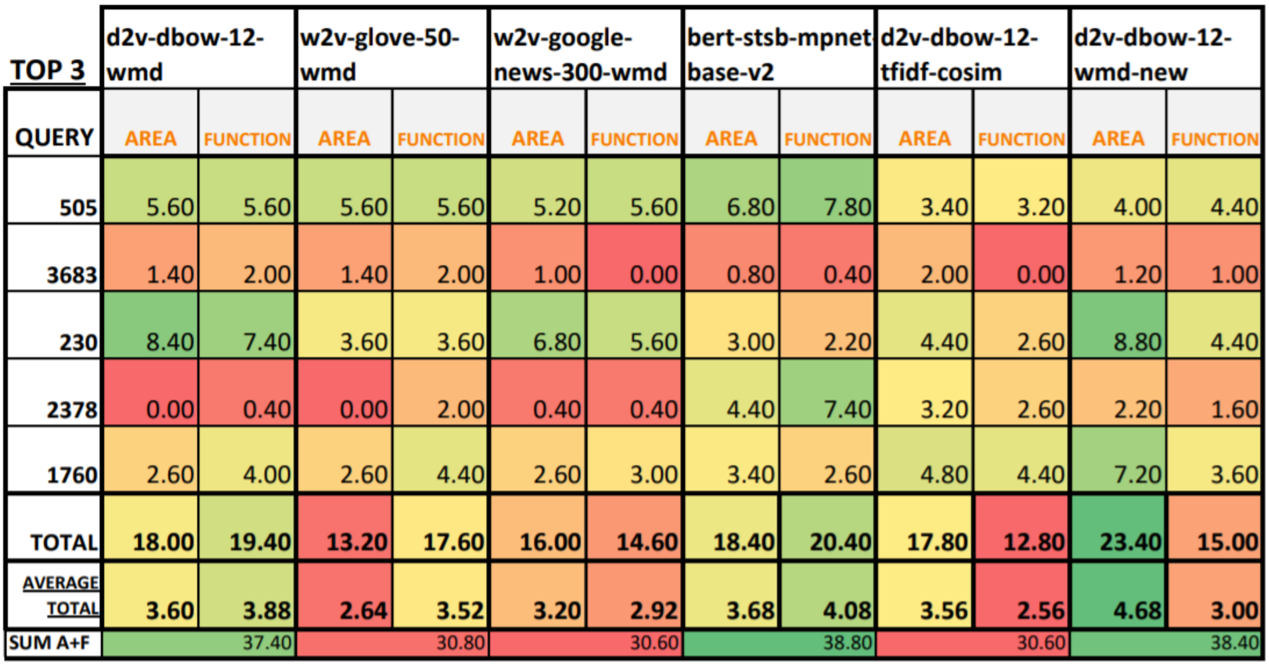
3. Run the command *jupyter notebook* to open the Jupyter Notebook web application.
4. The web application will run on http://localhost:8888/tree and the results will be available to read and reproduce in the notebook ModelExploration.ipynb. **Note:** running the full notebook again will take a long time.

In addition, there are several python modules that include custom utility functions.

## 9.4 Evaluation results

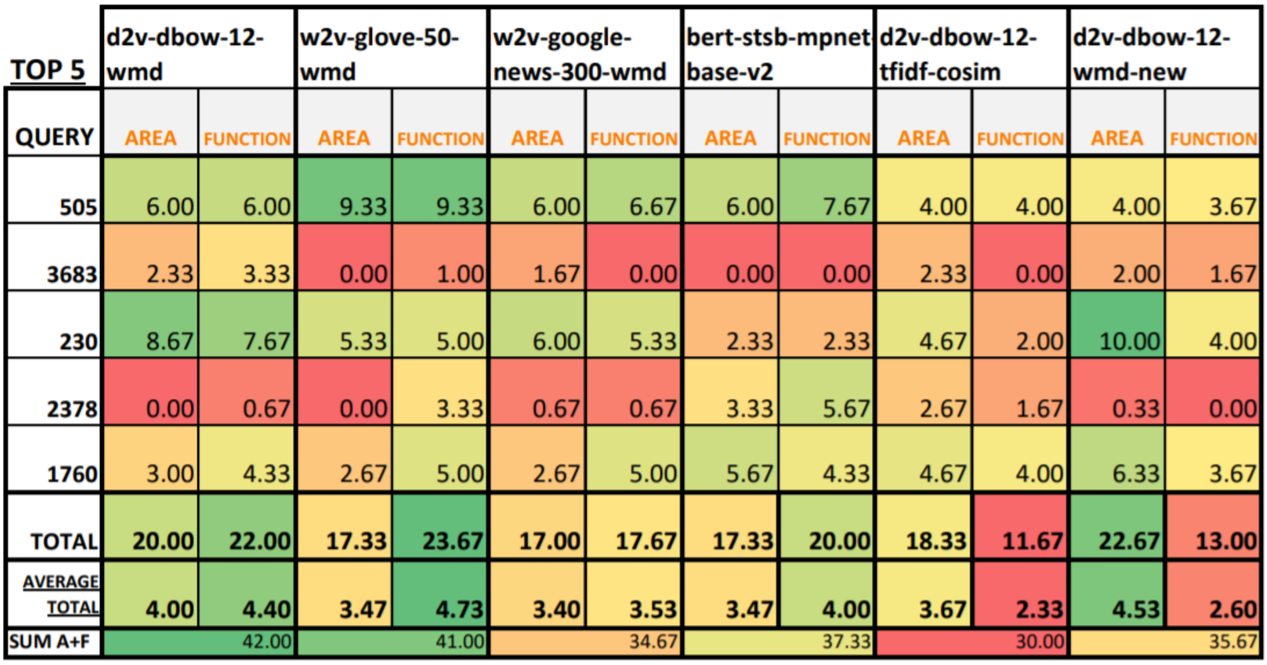
Below are the results for the model selection evaluation.

*Table 3. Evaluation results for top 9 similar documents.*



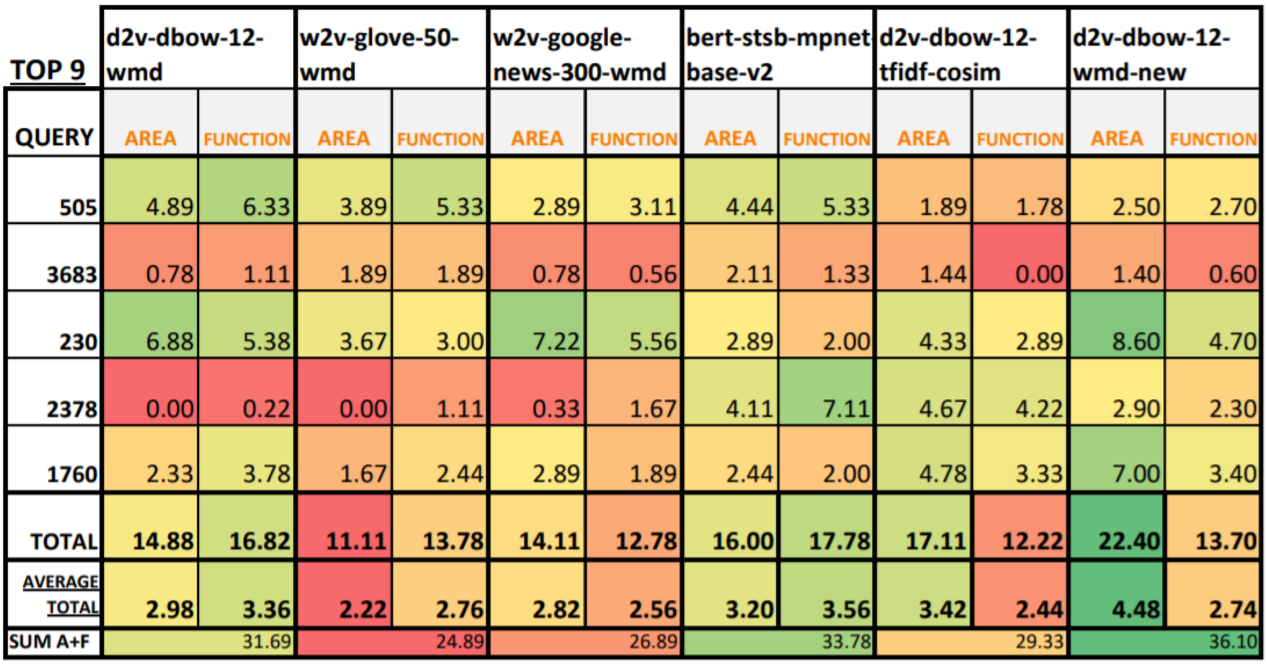
| TOP 3 | d2v-dbow-12-wmd | | w2v-glove-50-wmd | | w2v-google-news-300-wmd | | bert-stsb-mpnet base-v2 | | d2v-dbow-12-tfidf-cosim | | d2v-dbow-12-wmd-new | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QUERY | AREA | FUNCTION | AREA | FUNCTION | AREA | FUNCTION | AREA | FUNCTION | AREA | FUNCTION | AREA | FUNCTION |
| 505 | 5.60 | 5.60 | 5.60 | 5.60 | 5.20 | 5.60 | 6.80 | 7.80 | 3.40 | 3.20 | 4.00 | 4.40 |
| 3683 | 1.40 | 2.00 | 1.40 | 2.00 | 1.00 | 0.00 | 0.80 | 0.40 | 2.00 | 0.00 | 1.20 | 1.00 |
| 230 | 8.40 | 7.40 | 3.60 | 3.60 | 6.80 | 5.60 | 3.00 | 2.20 | 4.40 | 2.60 | 8.80 | 4.40 |
| 2378 | 0.00 | 0.40 | 0.00 | 2.00 | 0.40 | 0.40 | 4.40 | 7.40 | 3.20 | 2.60 | 2.20 | 1.60 |
| 1760 | 2.60 | 4.00 | 2.60 | 4.40 | 2.60 | 3.00 | 3.40 | 2.60 | 4.80 | 4.40 | 7.20 | 3.60 |
| **TOTAL** | **18.00** | **19.40** | **13.20** | **17.60** | **16.00** | **14.60** | **18.40** | **20.40** | **17.80** | **12.80** | **23.40** | **15.00** |
| AVERAGE TOTAL | **3.60** | **3.88** | **2.64** | **3.52** | **3.20** | **2.92** | **3.68** | **4.08** | **3.56** | **2.56** | **4.68** | **3.00** |
| SUM A+F | 37.40 | | 30.80 | | 30.60 | | 38.80 | | 30.60 | | 38.40 | |

*Table 4. Evaluation results for top 5 similar documents*



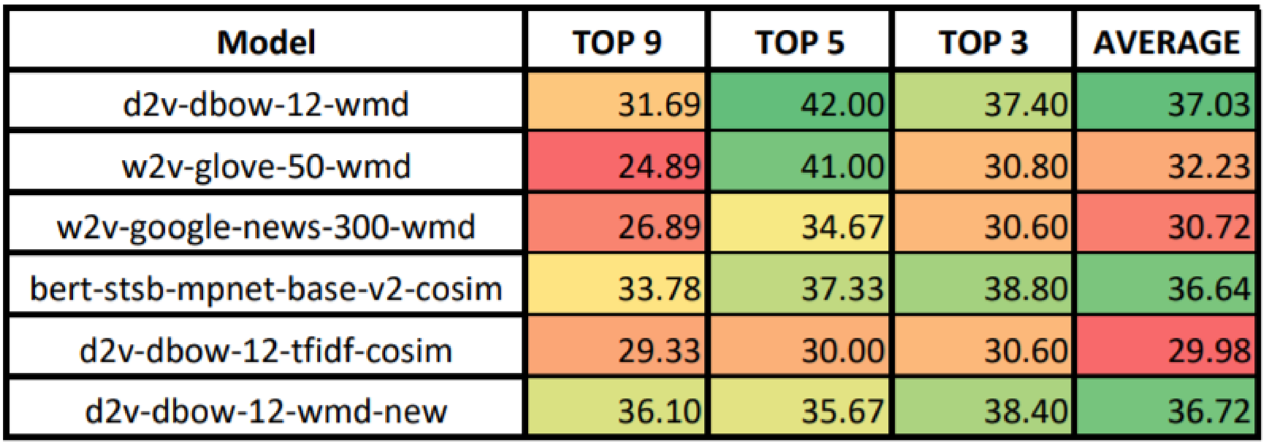
| TOP 5 | d2v-dbow-12-wmd | | w2v-glove-50-wmd | | w2v-google-news-300-wmd | | bert-stsb-mpnet base-v2 | | d2v-dbow-12-tfidf-cosim | | d2v-dbow-12-wmd-new | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QUERY | AREA | FUNCTION | AREA | FUNCTION | AREA | FUNCTION | AREA | FUNCTION | AREA | FUNCTION | AREA | FUNCTION |
| 505 | 6.00 | 6.00 | 9.33 | 9.33 | 6.00 | 6.67 | 6.00 | 7.67 | 4.00 | 4.00 | 4.00 | 3.67 |
| 3683 | 2.33 | 3.33 | 0.00 | 1.00 | 1.67 | 0.00 | 0.00 | 0.00 | 2.33 | 0.00 | 2.00 | 1.67 |
| 230 | 8.67 | 7.67 | 5.33 | 5.00 | 6.00 | 5.33 | 2.33 | 2.33 | 4.67 | 2.00 | 10.00 | 4.00 |
| 2378 | 0.00 | 0.67 | 0.00 | 3.33 | 0.67 | 0.67 | 3.33 | 5.67 | 2.67 | 1.67 | 0.33 | 0.00 |
| 1760 | 3.00 | 4.33 | 2.67 | 5.00 | 2.67 | 5.00 | 5.67 | 4.33 | 4.67 | 4.00 | 6.33 | 3.67 |
| TOTAL | **20.00** | **22.00** | **17.33** | **23.67** | **17.00** | **17.67** | **17.33** | **20.00** | **18.33** | **11.67** | **22.67** | **13.00** |
| AVERAGE TOTAL | **4.00** | **4.40** | **3.47** | **4.73** | **3.40** | **3.53** | **3.47** | **4.00** | **3.67** | **2.33** | **4.53** | **2.60** |
| SUM A+F | 42.00 | | 41.00 | | 34.67 | | 37.33 | | 30.00 | | 35.67 | |

*Table 5. Evaluation results for top 5 similar documents.*



| TOP 9 | d2v-dbow-12-wmd | | w2v-glove-50-wmd | | w2v-google-news-300-wmd | | bert-stsb-mpnet base-v2 | | d2v-dbow-12-tfidf-cosim | | d2v-dbow-12-wmd-new | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QUERY | AREA | FUNCTION | AREA | FUNCTION | AREA | FUNCTION | AREA | FUNCTION | AREA | FUNCTION | AREA | FUNCTION |
| 505 | 4.89 | 6.33 | 3.89 | 5.33 | 2.89 | 3.11 | 4.44 | 5.33 | 1.89 | 1.78 | 2.50 | 2.70 |
| 3683 | 0.78 | 1.11 | 1.89 | 1.89 | 0.78 | 0.56 | 2.11 | 1.33 | 1.44 | 0.00 | 1.40 | 0.60 |
| 230 | 6.88 | 5.38 | 3.67 | 3.00 | 7.22 | 5.56 | 2.89 | 2.00 | 4.33 | 2.89 | 8.60 | 4.70 |
| 2378 | 0.00 | 0.22 | 0.00 | 1.11 | 0.33 | 1.67 | 4.11 | 7.11 | 4.67 | 4.22 | 2.90 | 2.30 |
| 1760 | 2.33 | 3.78 | 1.67 | 2.44 | 2.89 | 1.89 | 2.44 | 2.00 | 4.78 | 3.33 | 7.00 | 3.40 |
| TOTAL | **14.88** | **16.82** | **11.11** | **13.78** | **14.11** | **12.78** | **16.00** | **17.78** | **17.11** | **12.22** | **22.40** | **13.70** |
| AVERAGE TOTAL | **2.98** | **3.36** | **2.22** | **2.76** | **2.82** | **2.56** | **3.20** | **3.56** | **3.42** | **2.44** | **4.48** | **2.74** |
| SUM A+F | 31.69 | | 24.89 | | 26.89 | | 33.78 | | 29.33 | | 36.10 | |

*Table 6. Average scores across top 9, top 5 and top 3 similar documents.*



| Model | TOP 9 | TOP 5 | TOP 3 | AVERAGE |
|---|---|---|---|---|
| d2v-dbow-12-wmd | 31.69 | 42.00 | 37.40 | 37.03 |
| w2v-glove-50-wmd | 24.89 | 41.00 | 30.80 | 32.23 |
| w2v-google-news-300-wmd | 26.89 | 34.67 | 30.60 | 30.72 |
| bert-stsb-mpnet-base-v2-cosim | 33.78 | 37.33 | 38.80 | 36.64 |
| d2v-dbow-12-tfidf-cosim | 29.33 | 30.00 | 30.60 | 29.98 |
| d2v-dbow-12-wmd-new | 36.10 | 35.67 | 38.40 | 36.72 |