MASTER'S THESIS 2023

Using Transformers To Improve Search Functions

Alice Berggren, Linnea Palmblad

ISSN 1650-2884
LU-CS-EX: 2023-26

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2023-26

Using Transformers To Improve Search Functions

Evaluering av sökalgoritmer för en chattapplikation med hjälp av Transformers

Alice Berggren, Linnea Palmblad

Using Transformers To Improve Search Functions

(An Application for a Smart Messaging System)

Alice Berggren
al5028be-s@student.lu.se

Linnea Palmblad
li3776pa-s@student.lu.se

June 29, 2023

Master's thesis work carried out at Telavox.

Supervisors: Jesper Gunnarsson, jesper.gunnarsson@telavox.com
Pierre Nugues, pierre.nugues@cs.lth.se

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

Abstract

Following the explosion of information available on the internet and in digital databases, search functions have become essential for efficient information retrieval (IR). Transformers have become increasingly popular in recent years for creating search functions due to their ability to handle natural language processing tasks with remarkable accuracy. Yet, many applications and companies still do not utilize this technology and instead use keyword search methods. This can lead to irrelevant or incomplete search results that force organizations to spend time manually searching for relevant results. To this end, we investigated the performance of transformer-based models on the sentence-pair tasks from the GLUE benchmark, representing different linguistic properties. Additionally, we evaluated the best-performing models on the IR benchmark CISI and on a dataset containing chat messages and queries taken from a messaging application. In this thesis, we show that the SBERT encoder model outperformed the other models based on the established benchmarks in our experiment. Furthermore, we developed a web application to measure the performance when using the SBERT model in combination with a cross-encoder and we found that the combination of a pre-trained cross-encoder and a retrieval model further increases the IR performance.

Keywords: NLP, Search Function, SBERT, Transformers, Cross-Encoder

Acknowledgements

We would like to thank Elias Vernersson and Niklas Bruce for their invaluable assistance during the course of our Master's Thesis at Telavox. They provided valuable support in conducting our experiments, and their dedication and time was greatly appreciated.

Additionally, for his input and expertise throughout the entire process of our thesis work, we also extend our appreciation to our supervisor at Telavox, Jesper Gunnarsson.

This work has been supported by the company Telavox which provided their office space, hardware, and support of their team members at the office. This has also been instrumental in the accomplishment of our research.

Lastly, we would like to express our deepest gratitude to our supervisor at LTH, Pierre Nugues, for reading drafts of this paper and whose expertise has been indispensable to our work. Without his guidance and knowledge, our research would not have been possible.

Contents

Chapter 1
Introduction

In today's digital age, the availability of vast amounts of data has created a phenomenon known as information overflow. In the realm of communication applications, particularly in the context of instant messaging systems, the phenomenon of information overflow becomes even more pronounced. As companies increasingly rely on online communication platforms to facilitate internal collaboration and engage with customers, the sheer volume of messages exchanged on these platforms can quickly become overwhelming. This abundance of data poses significant challenges in effectively managing, organizing and extracting valuable insights from chat conversations. The unstructured nature of chat messages further aggravates the problem, as relevant information can be scattered across numerous threads and lost amidst the noise.

Therefore, addressing the issue of information overflow in chat applications is of utmost importance to enable efficient navigation, retrieval, and analysis of crucial data. In natural language processing (NLP), this process of obtaining relevant information from extensive data collection based on user queries is called information retrieval (IR). Developing an IR system tailored for chat environments can help company employees and customers find the information they want securely and efficiently, even as the company grows.

Many of these IR systems struggle to understand the nuances of human language and context since they are mainly based on keyword or vector-based search models. These models are good at finding similar words but often fail to understand the purpose and intent behind language, leading to inaccurate results. Transformers is an alternative architecture that has proven suitable for understanding the complexity of human language and is considered state-of-the-art.

1.1 Problem definition

Many companies rely on search functions based on keyword search or vector space techniques such as tf-idf and BM25. These approaches gained popularity due to their simplicity,

efficiency, and ability to handle large volumes of text data. Treating documents as a collection of words or vectors allows companies to quickly retrieve relevant information based on specific keywords or similarity measures.

However, one of the significant limitations of keyword-based and vector space search functions is their inability to handle context effectively as described by Qaiser and Ali (2018). These approaches treat documents and search queries as unordered collections of individual terms without considering the relationships between words or the text's overall structure. As a result, they often fail to capture the nuances and semantics embedded in the context of the text, leading to sub-optimal search results.

More advanced search methods have emerged to address this limitation, leveraging NLP techniques and machine-learning algorithms. These approaches aim to capture the contextual information, semantic relationships, and the text's overall meaning. Neural network-based models, such as transformers, have demonstrated significant advancements in understanding the context of text by considering word relationships and capturing the overall meaning of the text. Because of this, researchers have achieved remarkable advancements in accuracy on many NLP tasks, including text classification (Yang et al., 2019), language understanding (Wang et al., 2019), machine translation (Lample and Conneau, 2019), and text summarization (Lewis et al., 2019). By leveraging the attention mechanisms and learning representations in transformers that capture contextual information, the application of transformer-based models also holds great potential for enhancing search functions. This surpasses the limitations of the widespread keyword-based and vector space techniques.

In this thesis, we investigated how to use transformer-based models to improve a search function in a chat application for the Swedish telecommunications company, Telavox. Telavox currently have a database containing over 300,000,000 chat messages that there is no way of efficiently search through except from manually. So here, an efficient search algorithm is crucial to ease the navigation of content. We used the models to transform the sentences into sentence embeddings that capture the context and relevancy of the chat messages based on semantic similarity. We explore five different encoder models: tf-idf, BM25, InferSent, SBERT, and DPR. See Chapter 4. Additionally, we evaluate the addition of a cross-encoder to re-rank the fetched relevant documents for a more accurate search function. See Chapter 6.

1.2 Contribution Specifications

In summary, our contributions encompass three key aspects.

- In Chapter 2, we present a newly curated IR dataset featuring meticulously annotated data. This dataset serves as a valuable resource for evaluating the capabilities of IR systems, capturing the intricate complexities and nuances associated with contemporary IR tasks specifically within chat applications.
- In Chapter 4, we demonstrate the efficiency of utilizing the Transformer-based model SBERT for IR. Through extensive evaluation, SBERT surpasses other models significantly, generating superior natural language embeddings for semantic similarity search.
- Lastly, in Chapter 6, we evaluate an approach that combines a cross-encoder with a bi-encoder, effectively optimizing accuracy outcomes while ensuring computational

efficiency. This innovative framework enhances the overall performance of the IR system on the Telavox chat dataset.

Additionally, we hope our research findings provide valuable insights and contributions that can benefit fellow researchers and practitioners involved in developing domain-specific search engines. By sharing our theoretical framework and implementation methodology, we aim to inspire and guide others interested in conducting further research in this field or leveraging these techniques within their respective company domains. We believe that the knowledge and approaches presented in this study have the potential to serve as a foundation for future advancements and innovations in the realm of search engine development, facilitating enhanced IR and discovery in specific domains.

1.3 Related works

In this section, we explain the dimensions fundamental to our work: What kind of search function architectures there are as well as their advantages and disadvantages. We highlight works in areas relevant to ours.

Keyword search

In IR and search systems, keyword search has been a widely adopted approach for many years. While keyword search is simple to implement, it has several limitations. One major limitation is the keyword search's inability to handle the semantic meaning of words. Hussan (2020) explain that since keyword search relies solely on matching exact terms, it often fails to capture the subtleties and semantic nuances present in a user's query.

Additionally, keyword-based systems may suffer when there are multiple meanings of a word and when there are different words with similar meanings, leading to imprecise and incomplete search results.

For instance, consider a search query for 'Apple', as illustrated in Figure 1.1. In a keyword-based search, the system would retrieve documents containing the term 'Apple' without differentiating between the fruit or the technology company and not considering incorrectly spelled versions of the term. This lack of context awareness can lead to nonrelevant search results, limiting the effectiveness of these search functions. These limitations have motivated

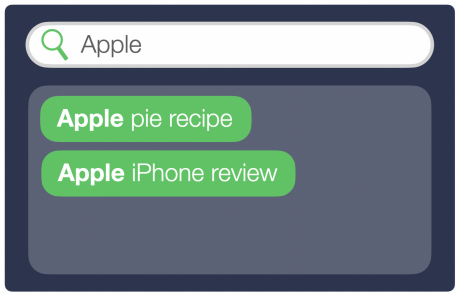


Figure 1.1: Example of the retrieved results in a keyword-based search.

researchers to explore more advanced techniques in order to overcome the challenges posed by keyword search and improve the accuracy of search results.

Vector space search

Vector space search was introduced by Salton and Lesk (1965) and is a widely employed technique in IR systems, aiming to address the limitations of keyword-based approaches. In vector space search, documents and queries are represented as high-dimensional vectors in a vector space. The vectors capture the statistical properties of the words or terms in the documents and queries, typically using methods like tf-idf or BM25. The vector space model allows for more flexible and nuanced matching based on similarity rather than strict keyword matching. By measuring the degree of similarity between vectors, the system can retrieve documents that are similar to the query, even if they do not share all the exact same terms. This approach enables more flexible matching based on similarity instead of strict keyword matching.

While vector space search has shown promise in improving search accuracy, it still faces challenges. This is because the approach is based on the bag-of-words concept, where each document is represented as a collection of individual words, disregarding the order and context in which they appear. Therefore, it struggles with capturing long-range dependencies and understanding complex relationships within documents.

Similarly to keyword search, vector space techniques retrieve documents based on the similarity of individual word vectors without considering the broader meaning or intended context of the query. Therefore, it can not handle negation very well. For example, as illustrated in Figure 1.2, the search query ‘Pie recipe without apple‘ will have a very high similarity score with a document that contains the sentence ‘Apple pie recipe‘. While the documents may have high similarity scores, they would be considered bad matches because it fails to capture the intended meaning and context of the query.

This highlights one of the limitations of vector space search approaches, as they primarily rely on term frequencies without understanding the semantic relationships or contextual information, leading to potential mismatches in search results. Furthermore, it may encounter difficulties when faced with out-of-vocabulary terms or rare words.

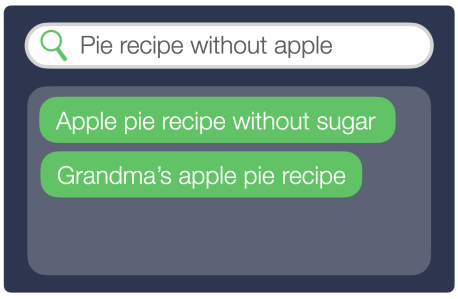


Figure 1.2: Example of the retrieved results in a vector space based search.

Transformers in search applications

Transformers have emerged as a powerful architecture in various NLP tasks, including search applications. The architecture was introduced by Vaswani et al. (2017) for the purpose of translation tasks. Since then, the use of transformers in search functions has gained significant attention due to their ability to capture contextual dependencies and model long-range relationships effectively. Ghojogh and Ghodsi (2020) thoroughly explain how the self-attention mechanism that transformers employ allows them to attend to all input tokens simultaneously. This mechanism helps transformers understand the context of a sentence based on its surrounding words. This capability is particularly beneficial in search applications where the retrieval of relevant information relies on understanding the semantics and relationships between words or phrases.

Figure 1.3 illustrates an example of a search using transformers. In context based search, the search query 'How do you make a dessert? ' will have a very high similarity score with a document that contains recipes and instructions relevant for making desserts, even though the sentences do not share a single word. Using keyword search the two sentences would not be a match, and using vector space search the resulting similarity score would at best be very low, but utilizing Transformers, the context and meaning of the sentences will be included in the similarity score, resulting in a high similarity score.

Transformers can be useful for a number of NLP tasks for enhancing search functions. Several studies have demonstrated the effectiveness of using transformers in tasks such as document retrieval (Zhang et al., 2021), question-answering (Lukovnikov et al., 2019), and semantic search (Muennighoff, 2022). Additionally, in studies by Yamoun et al. (2022) and Karpukhin et al. (2020) it has been shown that finetuning pre-trained Transformer models on specific search domains or using specialized architectures like Dense Passage Retrieval (DPR) and Sentence-BERT (SBERT) can significantly improve search accuracy and efficiency. The ability of Transformers to capture complex patterns, handle semantic nuances, and exploit large-scale pre-training has positioned them as a promising approach for enhancing search applications and addressing the limitations of traditional keyword-based or vector space search methods. We will describe some of the different Transformer architectures in more detail in Chapter 3.

Figure 1.3: Example of the retrieved results in a context based search.

Chapter 2

Datasets

To conduct a comprehensive evaluation of our IR system, it was crucial to consider three key components and corresponding measures. Firstly, we needed a way to assess the accuracy and completeness of the retrieval results by measuring how effectively the IR system retrieves relevant documents in response to queries. To achieve this, it was necessary to have a method for evaluating the quality of the retrieved documents.

Thus, secondly, annotated collections were required. These collections contained documents that were pre-annotated based on relevance and served as a reference for evaluating the performance of the IR system.

Lastly, to gain insights on the IR system, it was necessary to test the system on task specific datasets and compare its performance with existing baseline systems or state-of-the-art approaches.

Subsequently, to facilitate a comprehensive evaluation of our IR system, we selected two distinct benchmark datasets and constructed a domain-specific dataset utilizing data sourced from Telavox, encompassing all three key components.

The sections of this chapter are structured as follows: Section 2.1 offers an overview of the chosen benchmark datasets, providing a detailed analysis of their inherent characteristics. Furthermore, it gives an overview of the company dataset, outlining the methodology employed for its creation and the subsequent annotation process. Finally, Section 2.4 presents the evaluation methodology used, including selecting appropriate performance metrics tailored to each dataset.

2.1 Choosing the collections

To gain insights into the relative strengths and weaknesses of the different IR systems, it was necessary to test the system on task specific datasets to see how well they could understand language. To do this we will evaluate the systems on the widely-used sentence pair datasets in the General Language Understanding Evaluation (GLUE) benchmark that capture various

linguistic properties (Wang et al., 2019). The GLUE datasets encompass a diverse range of NLP tasks, each designed to address specific challenges in language understanding (Wang et al., 2019). Taking inspiration from Tien et al. (2019) researching semantic textual similarity, we used a similar setup for evaluating our models. We used this approach in order to see how well the systems understand language and to be able to compare our results not only against our own findings, but also against available baselines and state-of-the-art models.

To evaluate the models based on this task, we carefully selected five datasets from the GLUE benchmark that aligned with our research objectives since they related to tasks similar to semantic similarity. The tasks included semantic textual similarity and equivalence, question-answering, and predicting textual entailment.

For assessing semantic textual similarity, we aimed to determine if the systems could comprehend that different texts can convey similar meanings, even when using different words. For example, the sentence 'A man is tying his shoe' is semantically similar to 'A man ties his sneaker'. Additionally, we examined how well the systems could identify answers to questions within texts, testing their question-answering capabilities. Lastly, we evaluated the systems' understanding of textual entailment, assessing their capacity to recognize logical relationships between given sentences. For instance, from the sentence 'Google files for its long awaited IPO', the conclusion 'Google goes public' can be made.

As part of evaluating the models' performance as IR systems, we selected the Centre for Inventions and Scientific Information (CISI) dataset, created by the University of Glasgow (2023) that suited the intended use of the search function. The datset focuses on improving the effectiveness of IR systems and is comprised of a substantial collection of documents and queries with annotated relevance judgments by domain experts. Nevertheless, it is worth noting that the CISI dataset may not fully represent all types of IR and NLP tasks, and is not a perfect representation of the search function's intended use. Furthermore, there are no known existing baseline systems that we can directly compare our results to.

To specifically evaluate the IR system within the context of a chat application in a company domain, we incorporated a dataset comprising chat messages from Telavox's chat application. As the goal is to find a model for this exact data, it was important to see how the models performed with real data imitating the search function's intended use. However, due to the absence of assessments and labels for this dataset, as well as the limitations of this thesis, conducting a comprehensive evaluation solely based on this dataset was not feasible. Therefore, we combined the Telavox dataset with the GLUE and CISI datasets to ensure a robust and rigorous assessment of our system's performance in both language understanding and document retrieval tasks.

2.2 Format of the benchmark datasets

The GLUE datasets consist of entries organized into four columns, representing either sentence-to-sentence, question-to-question, or question-to-answer pairs. Furthermore, depending on the dataset, these datasets contain a class label indicating the relationship between the pairs, such as entailment/non-entailment or equivalence/non-equivalence. Alongside the GLUE datasets, we also chose to evaluate our models on the CISI dataset. The CISI dataset is comprised of a collection of documents, queries, and mappings that associate query IDs with document IDs, facilitating the linking of queries to relevant documents.

Following is detailed information about each of the chosen datasets, elucidating their characteristics and relevance to our evaluation:

- **STS-B** The Semantic Textual Similarity Benchmark (STS-B) is a collection from GLUE that consists of sentence pairs drawn from news headlines, video and image captions, and natural language inference data (Wang et al., 2019). Each pair has been annotated with a similarity score from 0 to 5, determining the semantic textual similarity (AL-Smadi et al., 2017). This task was chosen as the user sometimes does not have a specific document in mind when searching a query, but instead, a topic or certain keywords, and the search function should still retrieve relevant documents with similar meaning as the query.

- **MRPC** The Microsoft Research Paraphrase Corpus (MRPC) is a corpus from GLUE that consists of sentence pairs extracted from online news sources, with human annotations for whether the sentences in the pair are semantically equivalent or not (Wang et al., 2019). It tests whether the sentences are paraphrases of each other, and not only similar meaning as the semantic textual similarity task will determine (AL-Smadi et al., 2017). This was important to evaluate as the user should not need to know the exact words of any document to make a query in the search function that will retrieve relevant documents.

- **QQP** The Quora Question Pairs dataset (QQP) is a collection from GLUE that consists of question pairs from the community question-answering website Quora (Wang et al., 2019). The dataset is annotated on whether the questions in the pair are semantically equivalent with the same intent or not. We used this dataset in addition to the MRPC dataset to test the search function on queries and documents formulated as questions.

- **RTE** The Recognizing Textual Entailment (RTE) dataset is a GLUE dataset that is constructed based on news and Wikipedia texts (Wang et al., 2019). This task tests textual entailment by comparing two texts and recognizes if the truth of one text fragment follows from another, by predicting a directional relation between them (Korman et al., 2018). We chose to evaluate this as the user should be able to query a summary or main points of a document and the search function should retrieve relevant documents that could support that query.

- **QNLI** The Stanford Question Answering Dataset (QNLI) is a question-answering test collection from GLUE, consisting of question-paragraph pairs, where one of the sentences in the paragraph contains the answer to the corresponding question written by an annotator (Wang et al., 2019). We used it to test whether the search system is able to locate an answer to a question. This was important to evaluate in a question-answering system as the user should be able to form the query as a question and the search function should recognize and retrieve the documents answering that question. To evaluate this dataset we looked to see if the question and corresponding answer would be labeled as a match by the classifier or not.

- **CISI** The Centre for Inventions and Scientific Information (CISI) dataset is an IR dataset that consists of text data about documents and associated queries (University of Glasgow, 2023). We used it to evaluate relevancy based on a query given all the relevant documents in a corpus (Clough and Sanderson, 2013).

Each of the datasets were divided into training and validation sets. The metrics for the training set are shown in Table 2.1, and the validation set is shown in Table 2.2.

Table 2.1: The metrics for the used **training** datasets in the thesis. The table contains the number of rows, the minimum phrase length, the maximum phrase length, the median phrase length, and the average phrase length for each dataset. Length refers to the amount of words.

Dataset	Rows	Min. len	Max. len	Median len	Avg. len
MRPC	3668	7	42	14	21.9
RTE	2490	3	239	7	26.2
STS-B	5749	2	56	10	9.9
QNLI	104743	1	40	7	9.9
QQP	363846	1	237	13	11.1
CISI docs	1460	15	566	124	133.0
CISI queries	77	4	226	46	56.3
Telavox docs	10040	1	1441	13	13.8
Telavox queries	100	1	10	4	4.1

Table 2.2: The metrics for the used **validation** datasets in the thesis. The table contains the number of rows, the minimum phrase length, the maximum phrase length, the median phrase length, and the average phrase length for each dataset. Length refers to the amount of words.

Dataset	Rows	Min. len	Max. len	Median len	Avg. len
MRPC	408	9	35	15	22.0
RTE	277	3	164	7	25.4
STS-B	1500	3	29	13	11.4
QNLI	5463	1	153	31	18.8
QQP	40430	1	134	8	11.1
CISI docs	1460	15	566	124	133.0
CISI queries	35	55	345	111	126.8
Telavox docs	10040	1	1441	13	13.8
Telavox queries	100	1	10	5	4.9

2.3 Format of the Telavox chat data

The dataset we created from the Telavox database contained chat messages from their communication platform. The languages used at Telavox are mainly English and Swedish. However, Telavox also has offices in Spain and Denmark, among other places, so there are some messages that also exist in other languages. We decided not to take them into consideration in our research since they only take up a small share of the messages we retrieved.

The data we received from the company was stored in a MySQL database. A message was represented by a number of attributes a few of which are listed below:

1. **chatmessageid:** Identifies a sent chat message. It is the primary key of the table.
2. **fromuserid:** Identifies the user that sent the message.
3. **message:** Contains the chat message content, stored as a text type.
4. **sent:** Represents the time the message was sent. It is stored as a Datetime type.
5. **chatsessionid:** Identifies the chat the message was sent in.
6. **alsoasms:** A true or false value holding the information if the chat message was also sent as a text message outside of the messaging application.
7. **attachmentid:** Identifies possible attached files sent in the chat message.
8. **title:** Representing the possible title of the chat message.

We chose to create a dataset containing 200 rows of search queries that could possibly be used by the engineering team at Telavox. Then we split it into a validation and training set of 100 queries each. Looking at the metrics in Tables 2.1 and 2.2, there was a wide range in phrase length of the documents. Based on the median for the document length which was 13 words, the queries were chosen to range in length from 1 to 10.

As the scope of this thesis is a search function in a company domain, we focused on technical terms in both English and Swedish when creating the search queries. Telavox is a tech and engineering company so we believed that this type of language for the queries would be able to retrieve relevant results.

The validation query set was used to evaluate the models. We simulated users using the search function by letting the different search queries run through three chosen models. The models matched the queries with a collection of 10.040 chat messages. The models retrieved the ten most relevant matches. These matches were later annotated by us and the Telavox employees as relevant or not. The format of the resulting labeled Telavox dataset was, therefore, 3000 rows of data annotated by experts at Telavox, and 3000 rows of data annotated by us authors to use to finetune the models. Finally, each row in the dataset consists of a query, a chat message, and a label. The label represented if the message was relevant to the query or not.

For privacy and integrity reasons we decided together with Telavox to only collect chat messages from group sessions i.e. not private conversations on the messaging application.

2.4 Evaluating the datasets

In this section, we present the evaluation metrics used to assess the performance of various models, compare their effectiveness, and identify potential areas for improvement. By utilizing standardized evaluation metrics and well-defined datasets, we can ensure that our results are comparable with previous studies and provide a solid foundation for future research and advancements. Additionally, we aim to shed light on the strengths and weaknesses of the different approaches, highlight challenges and limitations, and provide recommendations for further advancements.

2.4.1 Metrics for the GLUE datasets

Models can be evaluated using various metrics to assess their performance. The following chosen metrics provided insights into different aspects of the model's performance, allowing for a comprehensive evaluation of its effectiveness in language understanding.

Binary classification terms

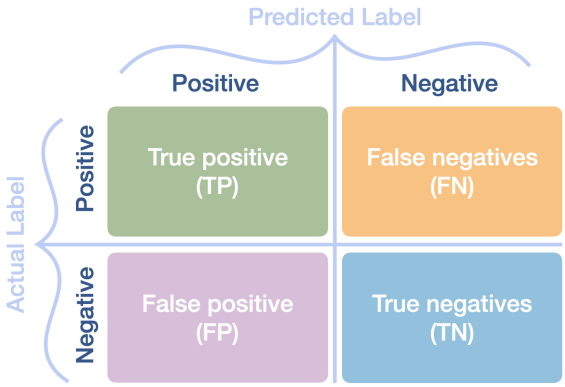


Figure 2.1: Visualization of the concepts true positives, false positives, false negatives, and true negatives.

In the context of binary classification, true positives (TP) are the instances of correctly classified data where the label was 1. These are the cases where the model's prediction aligns with the predicted positive labels. True negatives (TN) represent the amount of correctly classified data instances labeled 0, accurately identifying the absence of the positive class. On the other hand, false positives (FP) refer to the amount of data instances where the label was 0 but predicted 1 by the model. Finally, false negatives (FN) arise when the model incorrectly predicts a 0 for instances that are actually labeled 1. These concepts are visualized in Figure 2.1. These terms will be used in Eqs. 2.1, 2.2 and 2.3.

Accuracy

The accuracy of a model represents the number of correctly classified data instances over the total number of data instances. It is calculated using the number of true positives and true negatives predicted by the model, and is defined as:

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

F1 Score

The precision of the model measures the positive predictive value in classifying the data instances and is defined as:

$$Precision = \frac{TP}{TP + FP}. \tag{2.2}$$

The recall of the model measures the sensitivity or true positive rate and is defined as the following:

$$Recall = \frac{FP}{FP + FN}. \tag{2.3}$$

F1-score is a metric that gives a harmonic mean value depending on the FP and FN values, which is done using both recall and precision (Indurkhya and Damerau, 2010). It is defined as:

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{2.4}$$

There are different categories of F1 scores that can be used for different measurements, these are macro and micro (Kundu, 2022). The macro F1 score counts the average of all F1 scores and is defined as follows:

$$MacroF1Score = \frac{\sum_{i=1}^{n} F1score_i}{n}. \tag{2.5}$$

The micro F1 score when used with a binary dataset is the same thing as the accuracy defined in Eq. 2.1.

Correlation Coefficients

- **Pearson correlation coefficient** The Pearson correlation coefficient describes the linear correlation between two sets of data. It is calculated using the ratio between the covariance of two variables and the product of their standard deviations. The result ranges between -1 and 1 where 1 is a perfect positive correlation, -1 is a perfect negative correlation, and 0 no correlation. The correlation is defined in Eq. 2.6.

  $$\rho_{X,Y} = \frac{cov(X, Y)}{\sigma_X \sigma_Y}. \tag{2.6}$$

  In Eq. 2.6 and Eq. 2.7, cov is the covariance, and σ_X and σ_Y are the standard deviations of X and Y.

- **Spearman correlation coefficient** Spearman's correlation instead assesses monotonic relationships and is defined as the Pearson correlation coefficient between the rank variables. If there are no repeated data values, a perfect Spearman positive correlation of 1 or negative of -1 occurs when each of the variables is a perfect monotone function of the other. The correlation is defined in Eq. 2.7.

  $$r_s = \rho_{R(X),R(Y)} = \frac{cov(R(X), R(Y))}{\sigma_{R(X)} \sigma_{R(Y)}}. \tag{2.7}$$

  Where $R(X)$ and $R(Y)$ are the rank variables in Eq. 2.7.

2.4.2 Metrics for the IR datasets

When evaluating the models on the IR datasets, we used different metrics than when evaluating the language understanding datasets. In this section, we present two different accuracy functions and a measurement of how highly ranked relevant documents are. We also measured how the annotators' labeling compared to each other, using measurements on the inter-annotator agreement.

Precision and Recall @ n

In an IR system that retrieves a ranked list, the top-n documents are the n documents with the highest similarity scores together with the query. Precision at n is the proportion of the top-n documents that are labeled as relevant. The precision scores were calculated using the following formula:

$$P@n = \frac{r}{n}, \tag{2.8}$$

where r represents the number of relevant documents in a retrieved list and n is the number of returned documents (Craswell, 2009).

We also looked at the recall scores. This metric shows how many actual relevant results were shown out of all actual relevant results for the query. In a dataset where all the relevant documents are known,, the recall is calculated using the formula:

$$R@n = \frac{r}{R}, \tag{2.9}$$

where r is defined as the number of relevant documents, and R is the total number of relevant documents in the dataset.

Mean reciprocal rank

The reciprocal rank gives a value of how high up the most relevant item is, in an ordered result list. For example, if the most relevant item is the first item in the result, the reciprocal rank is 1, and otherwise less than one. It is calculated using the following formula:

$$\text{Reciprocal rank} = \frac{1}{Rank}. \tag{2.10}$$

The mean reciprocal rank looks at the mean rank across all queries, using the formula:

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}. \tag{2.11}$$

Inter-annotator agreement

Cohen's kappa is a measure used for evaluating the agreement between two annotators. A high kappa value shows high reliability meaning that the annotators are likely to come to the same conclusion. However, the measure does not say anything about the validity of the results themselves. It is defined as:

$$\kappa_c = \frac{p_0 - p_e}{1 - p_e}, \tag{2.12}$$

where p_0 and p_e are defined in Eqs. 2.13 and 2.14. p refers to the probability in this equation. The nominator gives the value of what agreement is actually obtained and the denominator in the function gives a value of what agreement can possibly be obtained.

$$p_0 = \frac{\#\text{Agreed samples}}{\#\text{Total samples}}, \tag{2.13}$$

$$p_e = p_{\text{Both would say a statement is true}} + p_{\text{Both would say a statement is false}}. \tag{2.14}$$

Fleiss' kappa is used instead of Cohen's kappa if there are more than two annotators, and is defined as:

$$\kappa_f = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}. \tag{2.15}$$

2.4.3 Benchmark datasets

We utilized diverse evaluation methods for the CISI and GLUE datasets. This was due to variations in the dataset structures, as discussed in the previous section.

Labeling the benchmark datasets

In order to compare the documents in the GLUE and CISI datasets, we adopted a systematic approach. We focused on the document/document and query/document pairs along with their corresponding labels. For each pairing, we computed a similarity score, which served as an indication of the degree of similarity between the elements being compared. To facilitate comparison with the labels, we transformed the similarity score into the same format as the label, leveraging the assistance of a classifier. This enabled us to generate predictions of the labels, which could then be compared to the actual labels to assess the performance of the models on the respective tasks defined by the datasets.

The evaluation of the models on the IR datasets followed a different methodology. Here, our objective was to examine the retrieved results when conducting a search query, taking into account the relevance of the retrieved documents. Instead of comparing the results to a predefined label, we sought to determine the number of relevant matches obtained and the placement of the relevant results within the retrieved list.

To achieve this, we calculated similarity scores between all the queries and all the documents within the CISI and Telavox datasets. This process involved systematically measuring the similarity between each query and every document in order to identify the documents with the highest similarity scores for each query. By generating a list of the most relevant documents based on their similarity scores, we were able to evaluate the effectiveness of the models in retrieving relevant information for a given query.

Evaluating the benchmark datasets

We calculated the performance of each of the models for each of the GLUE tasks using the chosen evaluation methods from the Hugging Face website, shown in Table 2.3. Using the

formula presented in Eq. 2.1, we calculated the accuracy score for MRPC, QQP, QNLI, and RTE. For computing the F1-score for MRPC and QQP, we used the formula shown in Eq. 2.4, and lastly we determined the correlation coefficients for STS-B by applying the formulas in Eqs. 2.6 and 2.7, as outlined in Table 2.3. Using the pre-established evaluation methods allowed us to compare all our results to other existing models and the current state-of-the-art models using the published leaderboard on the Hugging Face website.

For CISI, there was no established evaluation method, so we chose the most common metric accuracy score together with the F1 score as the data is not balanced. The label represented as a binary number gives information regarding whether a document is relevant to the query. 1 defining relevance and 0 non-relevance. The precision and recall at 10, 3, and 1 were also used as measurements for the IR system.

Table 2.3: Table of the evaluation methods used on the data collections.

Data collection	Evaluation method
MRPC	F1 score/Accuracy
STS-B	Pearson/Spearman correlation
QQP	F1 score/Accuracy
QNLI	Accuracy
RTE	Accuracy
CISI	F1 score/Accuracy

2.4.4 Telavox data

When creating a dataset, annotators are needed, as well as an evaluation method. Below we describe how we annotated and evaluated the models on the Telavox dataset.

Labeling the Telavox data

Three employees from Telavox annotated our dataset. The data annotation was conducted using custom Google Forms designed by us for this purpose. Initially, we selected 100 queries to generate the top ten results with the highest similarity scores from the different models under consideration. Subsequently, these Google Forms were distributed to the annotators, featuring two questions for each query. The first question asked the annotators to indicate the relevant items based on the given query, allowing them to select multiple relevant results. The second question required the annotators to identify the most relevant item among the search results for the given query or specify that ‘None of these results are relevant‘. An example from one page of the Google Form is depicted in Appendix A.1.

The employees took two weeks to answer the questionnaire, and it took approximately 60 minutes to complete the form. After receiving the data, we calculated the inter-annotator agreement and kappa values, and created a union of the results to get a collection of labeled data.

Evaluating the Telavox data

Due to time constraints, labeling all available test data from Telavox was not feasible. In this case where not all relevant documents have been identified, it is not straightforward to estimate recall, F1 or accuracy scores, which depends on knowledge of all the data in the data collection (Craswell, 2009). Instead, the evaluation is based on the number of relevant messages retrieved within the first n results, employing precision at rank n (P@n), and the mean reciprocal rank as the evaluation metrics (Clough and Sanderson, 2013). As in the case of CISI we can evaluate it using both precision and recall at rank n.

The search engine provided by Google typically displays the top 10 relevant documents on the first page of search results. Research indicates that a mere 0.44% of searchers proceed to the second page of results, exploring documents beyond the tenth position (Dean, 2020). Hence, we focused on the P@10 score, recognizing that most users do not venture beyond the initial set of results. Moreover, the same study revealed that merely 9% of Google searchers reach the bottom of the first page of search results, underscoring the importance of having the most relevant documents positioned at the outset of the search results. Therefore, we also assessed the P@3 and P@1 scores to gauge the placement of the most relevant documents within the retrieved results.

Chapter 3
Transformer Architectures

Transformers obtain the best results in most NLP tasks, so we predominantly evaluated models based on the transformer architecture in this thesis. The transformer model functions as a neural network that replaces traditional recurrent neural networks (RNNs) for sequence processing. With the help of a self-attention mechanism, the transformer model can attend to all input tokens in parallel rather than sequentially, allowing for the context of a sentence to be understood based on surrounding words. While RNNs process input sequentially, transformers allow for direct connections between any two positions in the input sequence.

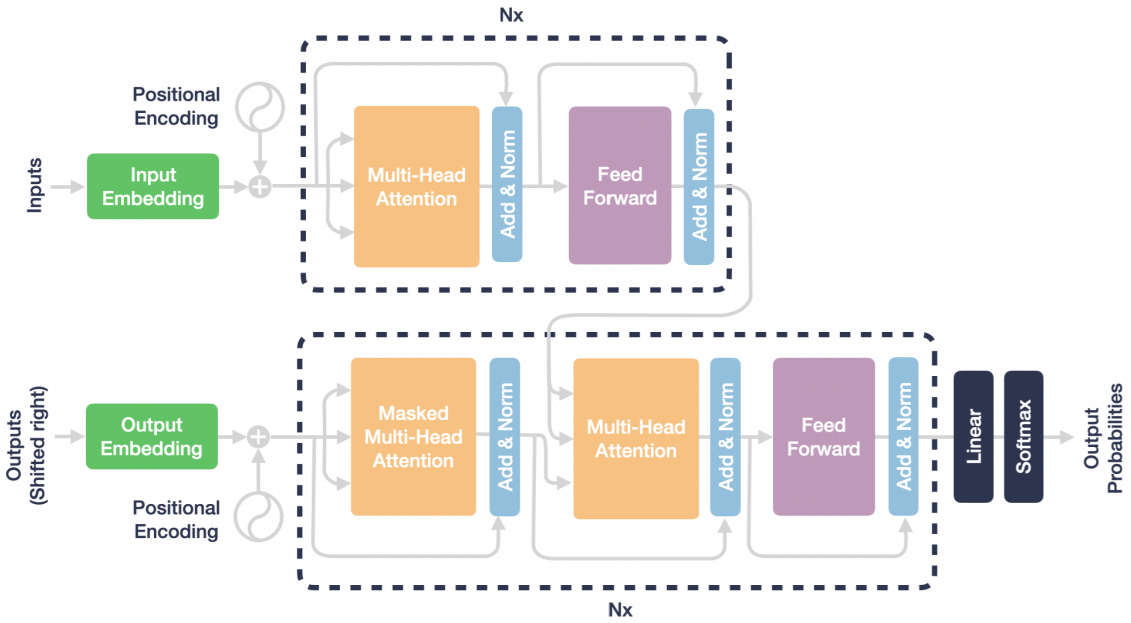


Figure 3.1: The transformer architecture after Vaswani et al. (2017)

.

This means that information can flow more easily across the entire sequence, capturing relationships between words or phrases that are distant from each other in the input. The architecture includes encoder and decoder layers with multi-head self-attention mechanisms and feed-forward neural network layers, as shown in Figure 3.1.

This architecture allows the transformer model to process input sequences of variable length without requiring complex recurrence or convolution operations, making it a popular choice for NLP tasks such as machine translation, language modeling, and text classification. The effectiveness of Transformers in capturing long-range dependencies in the input has further contributed to its widespread use in the field of NLP. It is currently used in many state-of-the-art search functions, particularly for tasks such as question-answering and IR.

3.1 Metrics

The models use different comparison methods when calculating the similarity between sentences and their embeddings. In this section, we present the two different similarity functions that the models use.

3.1.1 Cosine similarity

The cosine similarity is a valuable metric in developing search algorithms. It enables the assessment of the degree of similarity between query and document vectors. The cosine similarity is computed using the following formula:

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}||||\mathbf{v}||}. \tag{3.1}$$

3.1.2 Dot product

The dot product is the sum of the products of the corresponding entries of the two vectors. As cosine similarity, this metric can tell us how similar two vectors are, but the dot product also considers magnitude. The dot product is defined as:

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{n} u_i v_i. \tag{3.2}$$

3.2 Models using Transformers

Transformers can be finetuned for specific NLP tasks. This is one of the main benefits of using pre-existing Transformer-based models like BERT, SBERT, and DPR. These models are pre-trained on large amounts of text data, meaning before usage, the model is fed with large quantities of data before being finetuned further on more specific data. This is done since pre-training can be time-consuming and require significant memory usage. The models can be finetuned on smaller datasets for a particular NLP task, such as sentiment analysis, text classification, or question-answering. Through finetuning, the model adapts its parameters

to the specific task and improves its performance on that task. This approach of pre-training and finetuning has shown to be very effective and has led to state-of-the-art results on many NLP benchmarks.

3.2.1 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a pre-trained language model proposed by Devlin et al. (2019). The main innovation of BERT is its bidirectional training approach, which enables it to capture context from both the left and right directions of the input sequence. The pre-training phase involves using a masked language model and a next-word prediction objective (Tunstall et al., 2022). The model learns to predict missing and possible following words in sentences by randomly masking some of the input tokens and then using the surrounding context to make predictions. The aim is to develop a deeper understanding of contextual relationships between words. A significant advantage of this training is that it uses unlabeled data, making it possible to train on massive amounts of data.

By pre-training on a diverse range of text data, BERT thoroughly learns general language representations, which then can be finetuned on task-specific labeled data for various downstream NLP tasks. Because of the large amounts of data needed to pre-train the model, the pre-training process is the most time-consuming, computationally heavy, and data-demanding stage. Therefore, using pre-existing, pre-trained models and incorporating task-specific finetuning to adapt the model to a specific downstream task is a huge advantage.

Finetuning allows the model to adapt to the specific task's nuances and requirements, improving its performance on the target task. BERT has achieved state-of-the-art results on various NLP tasks, including question-answering, sentiment analysis, language inference, and classification. Figure 3.2 shows the architecture of using a pre-trained BERT model finetuned on single-sentence classification tasks. The output of this model is a class label, but it can be configured to complete other tasks.

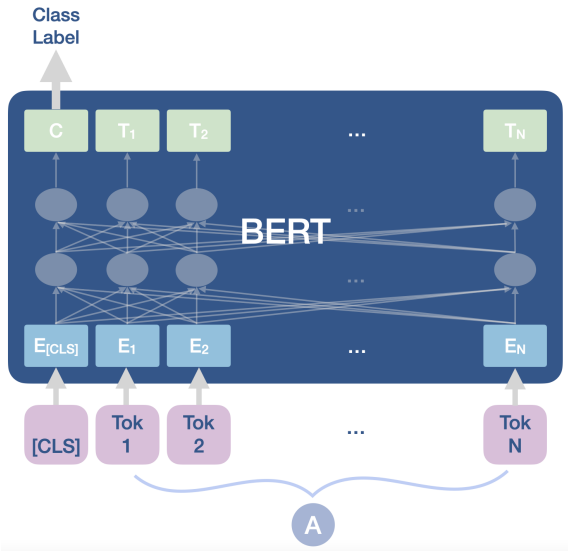


Figure 3.2: BERT architecture of a BERT model finetuned on single sentence classification, inspired by Devlin et al. (2019).

3.2.2 SBERT

Sentence Bidirectional Encoder Representations from Transformers (SBERT) is a language model proposed by Reimers and Gurevych (2019). SBERT is a type of bi-encoder architecture based on the pre-trained BERT model, which is trained using a Siamese network approach to generate fixed-length sentence embeddings. The SBERT model consists of two identical BERT models, as shown in Figure 3.3. The model takes a sentence pair A and B as input and learns to map them into a common embedding space in a way that semantically similar sentences are close together in the space, while dissimilar sentences are far apart. Thus, cosine similarity on the u and v embedding vectors yields a high similarity score if the sentences are semantically similar. The key innovation of SBERT is its ability to generate semantically meaningful sentence embeddings that can be used for various NLP tasks, including text classification, semantic search, and clustering. One important aspect of the architecture is the effectiveness of the Siamese network approach for learning semantically meaningful sentence representations. Other important aspects are the ability to finetune the pre-trained BERT model for specific downstream tasks and, most importantly, the usefulness of fixed-length sentence embeddings for entire documents.

This feature makes SBERT faster than BERT in comparing many sentences and finding matches in a corpus, as it allows SBERT to quickly compute the similarity between two sentences by comparing their embeddings directly, without the need for expensive token-level computations. As a result, SBERT can perform semantic similarity comparisons between many sentences much faster than BERT, making it suitable for applications such as semantic search, where large numbers of comparisons need to be made in real time.

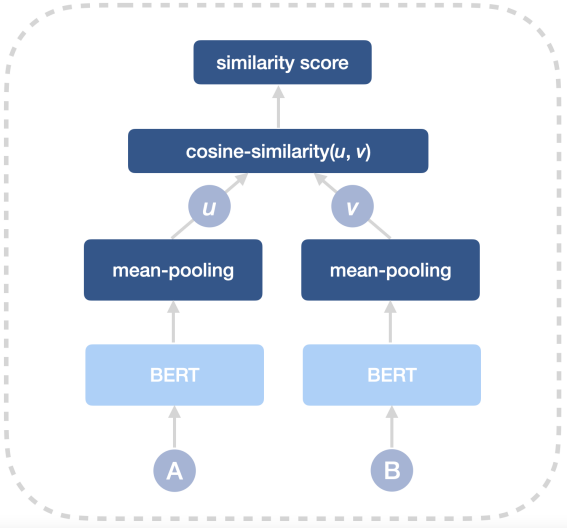


Figure 3.3: The high-level architecture of an SBERT model.

3.2.3 DPR

Karpukhin et al. (2020) introduced Dense Passage Retrieval (DPR), which is a novel neural network architecture. The objective of DPR is to improve the efficiency and effectiveness of open-domain question-answering, where the task is to find the most relevant answer to a

natural language question from a vast corpus of documents. The primary innovation of DPR is the use of dense embeddings to separately represent the queries and passages in the corpus. These dense embeddings are generated using two BERT models, as illustrated in Figure 3.4. One BERT model is trained and utilized for encoding contexts, while the other is trained to encode only questions. The embeddings are then used to compute the dot product between each query embedding and all the document embeddings in the corpus, allowing the system to quickly retrieve the most relevant passages for a given query. The key takeaways from the architecture include its ability to efficiently retrieve relevant passages from extensive corpora, its use of dense embeddings to capture the semantic meaning of queries and documents distinctly, and its effectiveness in enhancing the accuracy of open-domain question-answering systems.

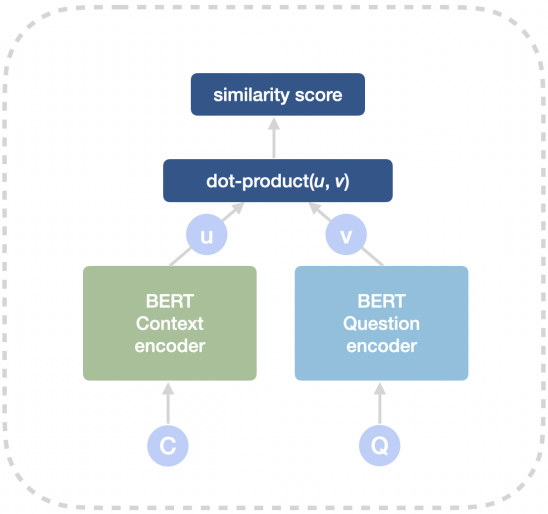


Figure 3.4: The high-level architecture of a DPR model.

3.2.4 Cross-Encoder

The cross-encoder is a neural network-based model that encodes both the query and document simultaneously. Unlike other encoder models, such as the BERT model in Figure 3.2, which are designed to encode individual sentences or passages, a cross-encoder is specifically designed to encode pairs of input texts. The purpose is to capture the relationship and interaction between two texts, such as a question and an answer or a query and a document. It takes both texts as input and processes them together to generate a joint representation that encompasses the contextual information and semantic relationship between the texts. The current state-of-the-art cross-encoders are finetuned BERT-based models. For example, Muennighoff (2022) utilized BERT as a cross-encoder by separating the query and document using a '[SEP]' token and passing them through the finetuned model together. Figure 3.5 shows the architecture of using a pre-trained BERT model finetuned on sentence-pair classification tasks, that can be used as a cross-encoder. The output of this model is a class label which could represent either a classification or similarity score.

However, for each new query, given a corpus of k documents, k forward passes are required. So, despite achieving higher accuracy than SBERT models when trained on a repre-

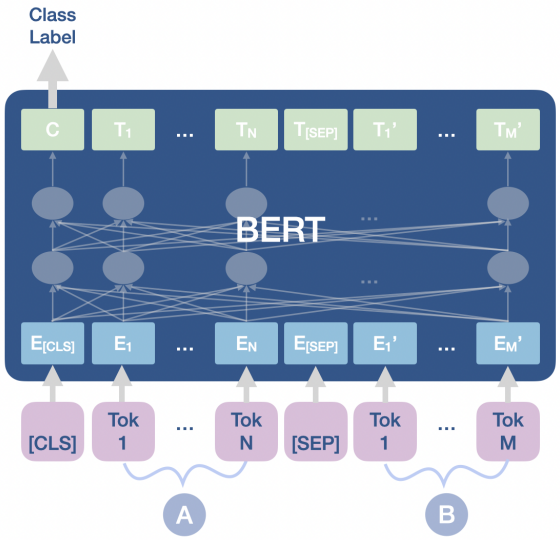


Figure 3.5: Architecture of a BERT model finetuned on sentence-pair classification tasks, inspired by Devlin et al. (2019).

sentative training set, the inefficiency of the cross-encoder architecture makes it not scalable for real-life semantic search applications (Reimers and Gurevych, 2019).

Reimers and Gurevych (2019) gave an example of this, stating that given a clustering task of a dataset with 10,000 sentences, using a BERT cross-encoder would require computing similarity scores for approximately 50 million sentence pairs, taking around 65 hours. Conversely, using a bi-encoder such as SBERT, enabled them to generate embeddings for each sentence individually, taking only 5 seconds.

Jesus-German et al. (2022) overcame the limitations of the cross-encoder architecture by using a bi-encoder combined with a cross-encoder in one single architecture. This combination is known as a cross-encoder with a bi-encoder structure and has proven to be highly effective for tasks such as semantic search and IR where the goal is to find the most similar sentences or documents given a query. The main advantage of this architecture is its ability to generate highly effective sentence embeddings while being computationally efficient.

The architecture works by using the bi-encoder network to generate embeddings based on the context of each sentence independently, capturing local information and retrieving a subset of the corpora that are labeled relevant. In contrast, the cross-encoder network captures global information by considering the entire context of both input sentences when generating the final similarity score, but only on the relevant subset of the corpora. The combination of these two architectures allows for the capture of both local and global contextual information when generating embeddings, making it a promising approach for real-world applications.

Chapter 4

Selection of Models

The first part of creating a search function was to develop an IR system that could rank all the documents depending on relevance to a search query. To do this, it was essential to have a good architecture that computes a high semantic similarity score to relevant documents. There are many different models and methods to do this, and limited time for evaluating them on the Telavox data set. Therefore, we experimented with different setups and compared their performance when classifying only the GLUE datasets. This was done so that we could evaluate the system on its natural language understanding and choose the most suitable encoder model to finetune and evaluate on the retrieval part.

We used the experimental set-up shown in Figure 4.1 to get the correct evaluation format to evaluate the datasets. The input is two sentences A and B, which are then converted to the embeddings u and v, from which a similarity score is calculated and used as input for the classifier that transforms them into the specified evaluation format. In this chapter, we explored the different options for the encoder model.

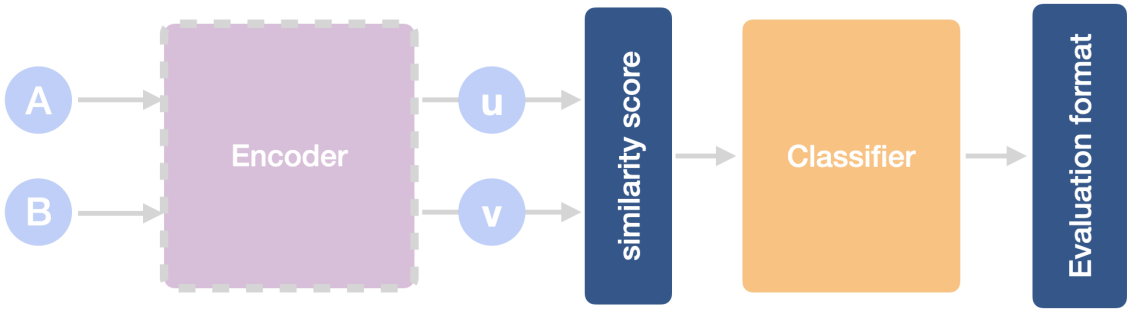


Figure 4.1: The architecture of the system for evaluating the encoder models on the GLUE datasets.

4.1 Encoder model

The first step is converting the collections of raw chat messages to embeddings with an encoder, as shown in pink in Figure 4.1. We created the embeddings in different ways by using tf-idf and BM25 to create word embeddings and pre-trained SBERT, DPR, and InferSent models to derive semantically meaningful sentence embeddings. Below we describe the implementation process in more detail for all the models.

4.1.1 Tf-idf

In order to find an appropriate evaluation method we started with creating a baseline using a tf-idf model. Tf-idf stands for term frequency-inverse document frequency and is a measurement of how important a word is within a document and corpora. This can prove useful for a search algorithm since the larger the corpus, the harder it is to find what data is relevant (Indurkhya and Damerau, 2010). The difference between these measurements is that the inverse term frequency looks at the occurrence of a word in an entire corpus while the term frequency looks at the frequency of a word in a document. Calculating just the term frequency will not give context on what words are special for a certain corpus. The inverse term frequency is thus needed to get a value of what is a more significant word in a certain context. To calculate the term frequency, the following formula is used:

$$\text{tf(t, d)} = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}. \tag{4.1}$$

Where t in Eq. 4.1 is the frequency of a specific term within a document d, $f_{t,d}$ is the number of times that term t occurs in document d, and the denominator is the sum of all terms in document d. To calculate the inverse document frequency the following formula is used:

$$\text{idf(t, D)} = \log(\frac{N}{|d \in D : t \in d|}). \tag{4.2}$$

Where t in Eq. 4.2 is the frequency of a specific term in a collection of documents D, N is the number of documents, and the denominator is the number of documents where the term t exists. The Tf and Idf scores are then multiplied in the following equation to get the tf-idf score for a term in a document, given a corpus of documents:

$$\text{tf-idf(t, D)} = \text{tf(t, d)} \cdot \text{idf(t, D)}. \tag{4.3}$$

We created a tf-idf model using the existing Python framework *sklearn* (scikit learn, 2023). For the GLUE datasets, we fit the model on all the words in the training datasets and then created the embeddings for both the training and validation set.

4.1.2 BM25

BM25 is a model that stems from tf-idf but uses something called Okapi weighting (Briggs, 2023). This type of weighting takes into consideration the number of times a term is mentioned. In tf-idf the score increases linearly if a term is mentioned multiple times, in BM25

the result is normalized based on how long the document is. To calculate the BM25 term frequency the following formula was used:

$$\text{tf(t, d)} = \frac{f_{q_i,d} \cdot (k_1 + 1)}{f_{q_i,d} + k_1 \cdot (1 - b + b \cdot \frac{|d|}{avg(d)})}. \tag{4.4}$$

In Eq. 4.4 BM25 introduces two new variables k_1 and b. Normally k_1 is around 1.25 and b is around 0.25, otherwise, the equation is similar to Eq. 4.1.

The idf Eq. 4.5 takes the total number of documents N into consideration in order to normalize the scoring so that the score does not increase linearly with the number of times the term is mentioned. The formula is defined as:

$$\text{idf(t, D)} = \log(\frac{N - |d \in D : t \in d| + 0.5}{|d \in D : t \in d| + 0.5} + 1). \tag{4.5}$$

We created a BM25 model using the existing Python framework *gensim*. The resulting setup was very similar to the implementation of the tf-idf model.

4.1.3 SBERT

SBERT is a transformer-based model specifically designed to generate high-quality embeddings for sentences, enabling more accurate semantic similarity comparisons and natural language understanding tasks. Therefore, we chose this model next to test on the GLUE datasets.

Implementation

For the pre-trained models, we used the Python *SentenceTransformers* framework which is one of the frameworks for state-of-the-art sentence and text embeddings.

Selection of pre-trained model

There are several models suitable for this task that have been extensively evaluated for their quality of embedded sentences and to embed search queries and paragraphs. For this thesis, there are several aspects and advantages of each of the models that we took into consideration. One of the aspects is overall performance and quality. Another aspect is the handling of different languages. Telavox is a multi-national company where the chat application is used in many different languages so the search function should preferably work with several languages. For this reason, we are also looking at the lower-performing multi-lingual models that generate aligned vector spaces, so that similar inputs in different languages are mapped close in vector space. The following selection describes the models from Huggingface (2023) that fit the requirements of our program:

all-mpnet-base-v2 The all-mpnet-base-v2 model currently provides the best quality on sentence embeddings and semantic search for a general purpose. The model is trained on several English datasets of over 1 billion training pairs and maps sentences to a 384-dimensional dense vector space.

all-distilroberta-v1 The all-distilroberta-v1 model currently produces some of the best-performing sentence embeddings. The model is also trained on several English datasets of over 1 billion training pairs and maps sentences to a 768-dimensional dense vector space.

all-MiniLM-L6-v2 The all-MiniLM-L6-v2 is one of the fastest, best-performing models with 5 times faster performance than the best-performing all-mpnet-base-v2 model. Similarly to the all-mpnet-base-v2 model it is trained on several English datasets of over 1 billion training pairs and maps sentences to a 384-dimensional dense vector space.

distiluse-base-multilingual-cased-v2 The distiluse-base-multilingual-cased-v2 model is a version of the multilingual Universal Sentence Encoder that supports over 50 languages including Swedish and English. It maps sentences to a 512-dimensional dense vector space.

paraphrase-multilingual-MiniLM-L12-v2 The paraphrase-multilingual-MiniLM-L12-v2 is the multilingual version of the paraphrase-MiniLM-L12-v2 model, trained on parallel data for over 50 languages including Swedish and English. The model maps sentences to a 384-dimensional dense vector space.

paraphrase-multilingual-mpnet-base-v2 The paraphrase-multilingual-mpnet-base-v2 is the multilingual version of the paraphrase-mpnet-base-v2 model, trained on parallel data for over 50 languages including Swedish and English. The model maps sentences to a 768-dimensional dense vector space.

To find the pre-trained model with the best performance for our program, we evaluated each of the model embeddings on the chosen GLUE tasks. The models and their performance scores can be seen in Table 4.1. The *all-mpnet-base-v2* model had the highest performance on the majority of the five GLUE tasks, the advantages of using a multilingual model are considered far more important than the small margins of difference in the performance scores

Table 4.1: Table of the performance of the SBERT models on the GLUE datasets. The datasets were evaluated using the methods described in Table 2.3. All values are scaled by 100.

Models trained on data in English					
Model	**MRPC**	**STS-B**	**QQP**	**QNLI**	**RTE**
all-mpnet-base-v2	**83.7/75.7**	88.1/88.1	**74.7/80.8**	**72.0**	65.0
all-distilroberta-v1	82.5/73.5	**88.3/88.3**	73.0/79.6	71.7	**66.8**
all-MiniLM-L6-v2	82.8/74.3	87.0/86.7	71.3/78.4	71.1	62.8
Models trained on data in several different languages					
Model	**MRPC**	**STS-B**	**QQP**	**QNLI**	**RTE**
paraphrase-multilingual-mpnet-base-v2	83.4/75.2	**88.2/89.1**	**71.8/79.3**	68.9	**62.4**
paraphrase-multilingual-MiniLM-L12-v2	**84.0/75.7**	87.0/87.54	71.2/78.3	68.5	60.6
distiluse-base-multilingual-cased-v2	81.5/72.3	81.9/81.9	67.4/75.7	**70.0**	54.9

in Table 4.1. Therefore, the chosen SBERT model for comparison was the multilingual model *paraphrase-multilingual-mpnet-base-v2*.

4.1.4 InferSent

InferSent by Conneau et al. (2017), developed by Facebook AI Research, is a sentence embedding model trained on natural language inference data that provides semantic sentence representations. The InferSent model utilizes a bi-directional LSTM architecture to encode the sentences. It is trained on a large corpus of labeled task-specific data. The model focuses on capturing the meaning and context of the sentences and has been widely used for various tasks, including sentence classification, semantic similarity, and textual entailment. Compared to the transformer models, it offers a simpler and more straightforward architecture that performs well on general semantic tasks.

Figure 4.2 shows the architecture of the InferSent model. As seen in the aforementioned figure, the vector input consists of three parts; a concatenation of the two representations $(\mathbf{u}, \mathbf{v})$, element-wise product $u * v$, and the absolute element-wise difference $|\mathbf{u} - \mathbf{v}|$. The resulting vector, which captures information from both sentences, is fed into a regression model which determines the similarity score.

The implementation of InferSent was done with the help of the GitHub repository provided by Conneau et al. (2017) available at `https://github.com/facebookresearch/InferSent`. The repository was downloaded and then the code used for the previous implementations was reused.

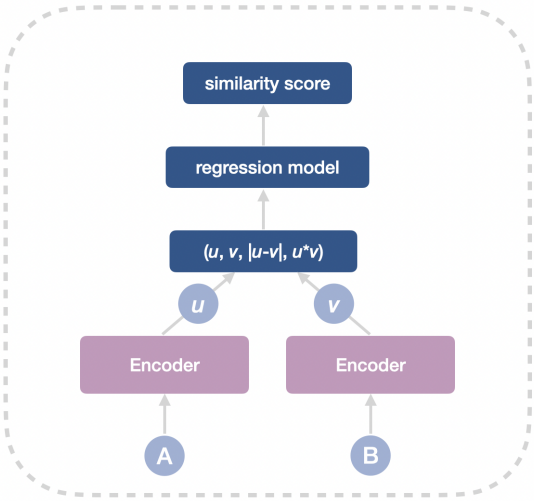


Figure 4.2: The architecture of the InferSent model.

4.1.5 DPR

As mentioned previously, DPR is a model where contexts and questions are represented as dense vectors whose representation is obtained by using two separate BERT models. One of the BERT models is trained and used for encoding contexts and the other is used to encode only questions. This makes DPR competitive in question-answering tasks. Therefore, we

chose to test this model next to test to see how it compared to the other encoder models in areas other than question-answering.

Implementation

For the pre-trained BERT-based DPR model, we used the Python Hugging Face library, which creates both document and query embeddings using the pre-trained BERT model *bert-base-cased*. For comparison, we also use the pre-existing DPR Facebook models proposed by (Karpukhin et al., 2020). The models proposed were the *facebook/dpr-ctx_encoder-single-nq-base* document encoder, and the *facebook/dpr-question_encoder-single-nq-base* query encoder.

Selection of pre-trained models

Similarly to SBERT, there are many pre-trained models available. In order to find the pre-trained model with the best performance for our program, we evaluated two of the model embeddings on the chosen GLUE tasks. Table 4.2 shows the models' performance scores on the datasets. The Facebook DPR model outperformed the more simple BERT-based DPR model in four out of five tasks. The BERT-based DPR model only outperformed the Facebook one with very small margins on the MRPC task. Therefore, the chosen DPR model for semantic-similarity comparison was the Facebook DPR model.

Table 4.2: Table of the performance of the DPR models on the GLUE datasets. The datasets were evaluated using the methods described in Table 2.3. All values are scaled by 100.

Query encoder	Context encoder	MRPC	STS-B	QQP	QNLI	RTE
bert-base-cased 1	bert-base-cased 2	**81.2/68.4**	2.7/3.0	21.3/64.0	49.5	53.1
facebook/dpr-question_encoder-single-nq-base	facebook/dpr-ctx_encoder-single-nq-base	81.0/68.1	**63.6/64.0**	**33.5/66.0**	**65.7**	**54.9**

4.2 Classifier

As shown in Figure 4.1, the next step after the encoding is classifying the data. The similarity score needs to be binary, i.e. 0 or 1 to represent entailment or equivalence. To this, we use the logistic regression classifier model from the *sklearn* library (scikit learn, 2023). For the STS-B task, we used a linear regression model from the same library instead as we wanted a float value between 0 and 5 as the predicted format. An important note is that when we used the model as a message retriever, the classifier was not used. We only added the model to the architecture to be able to evaluate the GLUE datasets. The model was given the cosine similarity scores of the training set and their existing label and then predicted the label for the validation set using only their similarity scores from the model.

4.3 Comparison of chosen models

By utilizing the most optimal model for each encoder approach, a comprehensive comparison was conducted to determine the superior method among the different techniques on the GLUE datasets. The outcome of this evaluation is outlined below.

4.3.1 Results

Table 4.3 shows the results of the evaluation of the chosen models on the benchmark datasets.

Table 4.3: Table of the performance of the chosen models on the GLUE datasets. The datasets were evaluated using the methods described in Table 2.3. All values are scaled by 100.

GLUE Baselines						
Rank Name	**Model**	**MRPC**	**STS-B**	**QQP**	**QNLI**	**RTE**
GLUE Baseline Highest	BiLSTM +ELMo +Attn	**84.4/78.0**	**74.2/72.3**	**63.1/84.3**	**79.8**	**58.9**
GLUE Baseline Lowest	CBOW	81.5/73.4	61.2/58.7	51.4/79.1	72.1	54.1
Our findings						
Rank Name	**Model**	**MRPC**	**STS-B**	**QQP**	**QNLI**	**RTE**
Baseline	tf-idf	81.0/71.1	72.0/72.0	47.6/65.5	71.5	57.8
	BM25	77.5/67.6	62.8/71.2	22.7/63.3	**72.3**	51.3
InferSent	Facebook model	80.6/69.1	61.8/62.4	51.6/67.6	59.7	50.2
DPR	Facebook model	80.5/67.6	63.6/64.0	33.5/66.0	65.7	54.9
SBERT	paraphrase-multilingual-mpnet-base-v2	**83.4/75.2**	**88.2/89.1**	**71.8/79.3**	68.9	**62.4**

The tf-idf model achieved above-average results in all of the GLUE tasks. Comparing the findings with the GLUE baselines, it was just slightly under the GLUE baseline, which is usually used to rank state-of-the-art models. The best performance of tf-idf was on the MRPC, STS-B, QNLI, RTE datasets. We suspect this could be because the sentence pairs in those datasets use many of the same words, for example, this sentence in STS-B: 'A girl is styling her hair' and 'A girl is brushing her hair.'.

The result for BM25 was worse than the baseline on all GLUE tasks apart from QNLI where it performed better than the lowest GLUE baseline. The bad performance in the other tasks might be the chosen implementation and values of $k1$ and b, or that these data sets worked better without weighting. InferSent performed worse than the baseline in all tasks except QQP. The chosen DPR model performed similarly to InferSent, except in QQP where it performed the second worst. The chosen SBERT model performed best out of all the tested

models in four out of the five GLUE tasks. We suspect the performance is because many of the SBERT models are pre-trained on similar tasks, so in this context, they work very well.

4.4 Finetuning the chosen model

In the next step, we finetuned the chosen model *paraphrase-multilingual-mpnet-base-v2* on the available datasets to improve the model further. As we had deemed the five GLUE and CISI datasets as appropriate for evaluation, we also decided to use that data for finetuning.

4.4.1 Losses

The loss function plays a critical role when finetuning the models on training data. For the SBERT models, we used the module *sentence-transformers.losses* to calculate the losses. For all the datasets with binary labels, we used ContrastiveLoss, and the CosineSimilarityLoss as the loss function for the datasets with float range labels.

ContrastiveLoss

The Contrastive loss function in this module expects two texts and a binary label as input. If the label is 1, then the distance between the two embeddings is reduced, and if the label is 0, then the distance between the embeddings is increased. The loss is calculated using the following equation:

$$\frac{1}{2}yd^2 + (1-y)\max(\alpha - d, 0), \tag{4.6}$$

where y is the input label, d is the distance between the document vector embeddings, and α is the margin. If the two documents are similar, their distance should be less than the margin.

CosineSimilarityLoss

Cosine similarity loss expects two texts and a float label as input. By default, it minimizes the following loss:

$$\|y - \cos(\mathbf{u}, \mathbf{v})\|_2, \tag{4.7}$$

where y is the input label, and $\mathbf{u}$ and $\mathbf{v}$ the two document embedding vectors.

4.4.2 Method

As shown in Table 2.1 the GLUE training datasets greatly differ in size. For instance, the MRPC dataset is 100 times smaller than the QQP dataset. Therefore, we used a subset of maximum 10,000 rows of each of the GLUE datasets for finetuning the models on. This was done to not over-train the data on the bigger datasets and for the finetuning to not be too time-consuming. The resulting model trained on this subset is named *GLUE*.

To see how this affected the results, we created an additional GLUE training set using larger subsets of the larger datasets, i.e. QQP and QNLI. The resulting model trained on this subset is named *GLUE + extra QQP & QNLI*.

For the CISI and Telavox dataset we used all available labeled data. We chose to separate the different finetuned models to see the impact the finetuning of each benchmark had on the model.

For each model, we used a batch size of 16 for a total of 3 epochs, this format suited the environment best, as we uploaded and executed the finetuning in Google Colab using a GPU hardware accelerator as the chosen runtime.

4.4.3 Result

Table 4.4 shows the results of the different finetuned variants of the SBERT model. In the top row ´Turing ULR v6´ is shown as it was the highest-scoring model on the GLUE leaderboard. However, this model is not implemented as there are no public libraries of pre-trained models or public repositories to use. Therefore, it is used for comparison only.

In Table 4.4, we can see that the result for the models does not vary that much for the different finetuned models. What we could see is that the addition of more QQP and QNLI data only slightly increased the performance on the QQP dataset while performing worse on all the others, so we used the balanced GLUE training data for all the other models.

The best-performing model is different for all the different datasets, and there is no obvious best-performing model. However, the best result for the average performance score of all data sets is the model finetuned with all the datasets. Therefore, we will also evaluate the model finetuned on all the datasets on the IR systems.

Table 4.4: Table of the performance of finetuned variants of the SBERT model 'paraphrase-multilingual-mpnet-base-v2' on the CISI and GLUE datasets. The datasets were evaluated using the evaluation methods described in Table 2.3. All values are scaled by 100.

Current state-of-the-art model							
Model	**MRPC**	**STS-B**	**QQP**	**QNLI**	**RTE**	**CISI**	**Avg.**
Turing ULR v6	**94.2/92.3**	**93.5/93.1**	**76.4/90.9**	**96.7**	**93.6**	–	**91.34**
Our findings							
Finetune data	**MRPC**	**STS-B**	**QQP**	**QNLI**	**RTE**	**CISI**	**Avg.**
No data	83.4/75.2	88.2/89.1	71.8/79.3	68.9	62.4	**22.2**/97.5	72.28
TELAVOX	84.0/77.0	88.2/89.1	72.7/79.3	50.5	52.3	20.3/97.0	67.77
CISI	84.4/76.2	86.4/88.1	70.8/78.1	**68.4**	61.0	20.6/**97.7**	71.76
GLUE	86.8/80.9	**89.3/89.3**	77.2/82.6	61.5	62.8	14.7/97.5	72.24
GLUE + extra QQP & QNLI	84.0/76.2	69.0/72.1	**79.0/83.9**	59.1	60.0	6.1/92.8	66.77
CISI + TELAVOX	84.1/75.7	85.8/87.7	70.8/78.2	68.3	**63.2**	20.3/97.4	71.92
GLUE + TELAVOX	**87.1/81.4**	88.9/89.2	77.8/83.1	49.7	61.7	14.5/97.5	70.19
GLUE + CISI	85.2/77.9	87.8/88.5	75.2/81.0	66.4	**63.2**	17.5/95.6	72.32
GLUE + CISI + TELAVOX	85.9/78.9	87.2/88.2	75.9/82.8	66.4	61.7	17.8/95.1	**72.33**

Chapter 5

Models as Information Retrieval systems

In this Chapter, we evaluate three models on the IR datasets. As seen in the last chapter, the model performing best on the semantic similarity datasets was the SBERT model *paraphrase-multilingual-mpnet-base-v2*. From this pre-trained model, we created eight different finetuned versions. The model finetuned on the GLUE, CISI, and Telavox data was chosen as the second model. In this chapter, these two models are compared to the tf-idf baseline model.

In previous chapters, the models have been evaluated on the GLUE benchmark. The GLUE benchmark is used for analyzing natural language understanding systems. However, the GLUE benchmark is not made for analyzing IR systems that have a similar structure as the Telavox dataset with search queries and a document collection that we want to rank on relevancy. Figure 5.1 show the architecture of an IR system. The CISI and Telavox datasets simulates the real-world use as an IR system more accurately. Therefore, it was also important to evaluate the model on the Telavox and CISI datasets.

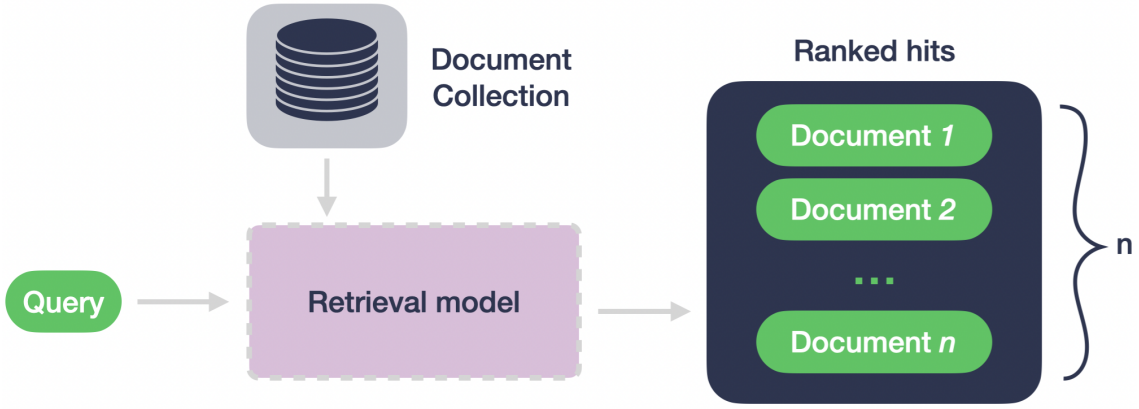


Figure 5.1: The architecture of the system for evaluating the encoder models on the IR datasets.

5.1 Telavox Dataset

Evaluation of the models involved evaluating the characteristics of the IR system itself and also assessing the satisfaction with the system by the employees of Telavox. The three annotators from Telavox are referred to as Annotator 1, Annotator 2, and Annotator 3. To ensure that we had explained the annotation process correctly, we also annotated the data for tf-idf ourselves to see if we got similar results. These results are referred to as Author 1 and Author 2.

5.1.1 Evaluation of the Telavox Dataset

We evaluate the models on the precision and mean reciprical rank at 1, 3, and 10 to see how many documents and where the most relevant documents are located in the results, using Eq. 2.8 and Eq. 2.11. The results for the calculated precision can be found in Table 5.1. In the table, the average percentage of irrelevant results and MRR of the 10 retrieved messages is also shown. As we do not know the total number of relevant documents in the collection of 10,040 documents, we do not use the recall at n method instead and used the position of the *most* relevant document in the retrieved result to calculate the MRR.

Table 5.1: Table of the calculated Precision at n, the average percentage of irrelevant results, and mean reciprocal rank for the different annotators and models on the Telavox dataset.

Model	Annotator	P@1	P@3	P@10	Irrelevant	MRR
tf-idf	Author 1	35%	27%	22%	34%	0.40
tf-idf	Author 2	23%	17%	16%	28%	0.36
tf-idf	Annotator 1	21%	17%	14%	41%	0.38
tf-idf	Annotator 2	18%	12%	9%	65%	0.54
tf-idf	Annotator 3	50%	42%	34%	14%	0.39
SBERT	Annotator 1	21%	17%	15%	36%	0.32
SBERT	Annotator 2	8%	8%	8%	69%	0.38
SBERT	Annotator 3	30%	29%	27%	26%	0.31
SBERT finetuned	Annotator 1	7%	7%	7%	72%	0.28
SBERT finetuned	Annotator 2	13%	9%	9%	64%	0.37
SBERT finetuned	Annotator 3	8%	8%	7%	62%	0.28

The average performance of each model are presented in Table 5.2. This analysis aimed to determine the model that demonstrated the highest overall performance based on the data provided in Table 5.1. By aggregating the individual assessments of the annotators, we were able to obtain a comprehensive evaluation of each model's effectiveness. The average scores provided valuable insights into the relative performance of the models and allowed for a comparison of their respective strengths and weaknesses. These findings played a crucial role in identifying the top-performing model among the ones evaluated, providing valuable guidance for further refinement and optimization.

Table 5.2: Table of the average Precision at *n*, percentage of irrelevant results, and mean reciprocal rank for the different models on the Telavox dataset.

Model	P@1	P@3	P@10	Irrelevant	MRR
tf-idf	**29.7%**	**23.7%**	**19.0%**	**40.0%**	**0.44**
SBERT	19.7%	18.0%	16.7%	43.7%	0.34
SBERT finetuned	9.3%	8.0%	7.7%	66.0%	0.31

5.1.2 The annotators

After we received the data, we calculated the inter-annotator agreement to see the validity of the results. Table 5.3 shows the inter-annotator agreement between the different annotators using Cohen's kappa values. To get an overview of the overall agreement, Table 5.4 shows the inter-annotator agreement between all three annotators using Fleiss' kappa values. The kappa values range from 0 to 1, where 1 would mean a perfect agreement and 0 no agreement. The highest Cohen's kappa value for each of the models and the overall highest Fleiss' kappa value is marked in bold.

Table 5.3: Inter-annotator agreement between the different Telavox annotators.

Model	Metric	Annotator	Result
tf-idf	Cohen's kappa	Annotator 1 and Annotator 2	**0.34**
tf-idf	Cohen's kappa	Annotator 1 and Annotator 3	0.32
tf-idf	Cohen's kappa	Annotator 2 and Annotator 3	0.25
SBERT	Cohen's kappa	Annotator 1 and Annotator 2	0.30
SBERT	Cohen's kappa	Annotator 1 and Annotator 3	0.26
SBERT	Cohen's kappa	Annotator 2 and Annotator 3	**0.37**
SBERT finetuned	Cohen's kappa	Annotator 1 and Annotator 2	0.20
SBERT finetuned	Cohen's kappa	Annotator 1 and Annotator 3	0.19
SBERT finetuned	Cohen's kappa	Annotator 2 and Annotator 3	**0.39**

Table 5.4: Inter-annotator agreement between all the Telavox annotators.

Model	Metric	Result
tf-idf	Fleiss' kappa	0.27
SBERT	Fleiss' kappa	**0.30**
SBERT finetuned	Fleiss' kappa	0.26

5.2 CISI

We also evaluated the models on CISI. Identically to the Telavox dataset, we evaluated the models on precision at 1, 3, and 10. The precision is calculated using the method shown in

Eq. 2.8. Table 5.5 shows the results for the precision at 1, 3, and 10 for the CISI dataset.

Table 5.5: Table of the calculated **precision at *n*** for the different models on the CISI dataset.

Model	P@1	P@3	P@10
tf-idf	52.9%	39.2%	21.2%
SBERT	**58.8%**	**49.0%**	**27.6%**
SBERT finetuned	17.6%	25.5%	21.8%

As opposed to the Telavox dataset, for CISI we do know the total number of relevant documents in the collection, so we are able to calculate the recall at 1, 3, and 10 to see where the relevant results were located in the retrieved documents. The recall is calculated using the method described in Eq. 2.9. Table 5.6 shows the results for the calculated recall.

Table 5.6: Table of the calculated **recall at *n*** for the different models on the CISI dataset.

Model	R@1	R@3	R@10
tf-idf	**9.5%**	**14.5%**	18.7%
SBERT	3.8%	10.6%	17.3%
SBERT finetuned	1.7%	7.0%	**18.9%**

5.3 Result

We examined performance of a tf-idf, SBERT and finetuned SBERT model conducted on two different datasets: the Telavox dataset and the CISI dataset. The evaluation encompassed various metrics, providing insights into the models' performance in terms of relevance and retrieval accuracy. Additionally, the inter-annotator agreement was assessed to gauge the consensus among evaluators. These findings shed light on the strengths, weaknesses, and overall effectiveness of the models, serving as a foundation for choosing the best model to test further improvements on.

5.3.1 Telavox dataset

Based on the findings presented in Tables 5.2 and 5.1, several conclusions can be drawn regarding the performance of the evaluated models on the Telavox dataset. The tf-idf model demonstrated the highest precision across all depths, while SBERT followed with slightly lower precision scores. The finetuned version of SBERT exhibited the lowest precision scores.

Additionally, the percentage of irrelevant results provides insights into the models' ability to filter out irrelevant documents. The tf-idf model achieved the lowest percentage of irrelevant results, with 40.0%. SBERT had a slightly higher percentage of irrelevant results, at 43.7%. The finetuned SBERT model had the highest percentage of irrelevant results, with 66.0%.

Furthermore, the MRR provides an overall measure of the models' ranking performance. The tf-idf model achieved the highest MRR score of 0.44, indicating that, on average, the relevant documents were ranked higher compared to the other models. SBERT had a lower MRR of 0.34, while the finetuned SBERT model had the lowest MRR of 0.31.

Overall, these findings suggest that the tf-idf model demonstrated the highest performance in terms of precision, relevance filtering, and ranking accuracy on the Telavox dataset. SBERT showed competitive performance but had slightly lower precision and a higher percentage of irrelevant results. The finetuned SBERT model exhibited the lowest performance across all metrics, indicating the potential limitations of the finetuning approach.

The findings regarding the inter-annotator agreement indicate a low level of agreement among the annotators in evaluating the performance of the models. The calculated median Fleiss' kappa of 0.27 and Cohen's kappa of 0.30 suggest that there is limited consensus among the annotators in their assessments. These values will be discussed in more detail in Chapter 7, where the potential factors contributing to the low agreement will be explored and analyzed.

5.3.2 CISI

On the CISI dataset we observed variations among the models. In terms of precision, SBERT outperformed tf-idf at all ranks. Notably, the finetuned version of SBERT yielded lower precision scores across all ranks.

Moving on to recall, we examined the recall at different retrieval depths. Here, SBERT yielded a lower recall than tf-idf. Surprisingly, the finetuned SBERT model exhibited the lowest recall scores at R@1 and R@3, but interestingly, a higher recall at R@10.

While SBERT generally outperformed tf-idf in terms of precision, it showed lower recall scores. Additionally, the finetuned version of SBERT had lower precision overall, but it achieved a higher recall at R@10.

5.3.3 Chosen model

Based on the comprehensive evaluation of the models on the CISI and Telavox datasets, it is evident that SBERT slightly outperformed tf-idf. These findings support the selection of SBERT as the preferred model for the subsequent chapter. The higher precision values achieved by SBERT at various ranks on the CISI dataset, along with the low annotator agreement on the Telavox dataset, and its superior performance in the last chapter, highlight its suitability for the intended purposes. As a result, SBERT will be the model of choice moving forward, based on its performance in all evaluations.

Chapter 6

Re-ranking

The next step after deciding the best encoder model for our use case was to look at the possibility of re-ranking the fetched search results based on sentence comparison. The re-ranking can be done in different ways. The goal of re-ranking is to optimize search results so that the highest documents in the list are the most relevant for a query. We decided to use a bi-encoder together with a cross-encoder for this. Cross-encoders have high performance, as they perform attention across the query and the document (Reimers, 2022). The reason this is not exclusively used for the re-ranking is that scoring large amounts of query and document pairs would be expensive in terms of time and computing power. Hence, we used the faster bi-encoder model based on the result in Chapter 4, to retrieve a set of 10 possible candidate documents. The 10 documents were then re-ranked by the cross-encoder in order to get the most relevant documents ranked highest. Figure 6.1 depicts an overview of the architecture.

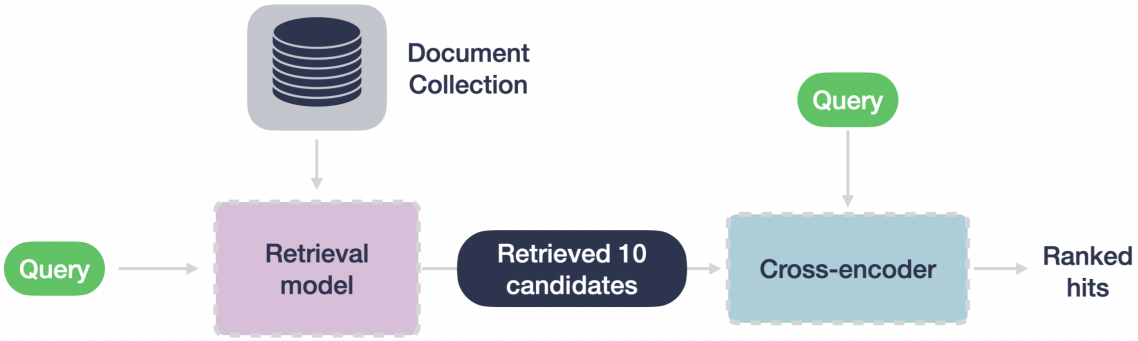


Figure 6.1: The architecture of a re-ranking system.

6.1 Method

The cross-encoder model used was a transformer network that takes a query and a document as input and outputs a similarity score between 0 and 1. The similarity score is then multiplied by the similarity score from the retriever model in order to create a new rank order (Reimers, 2022).

Several cross-encoder models are available online. These models are suitable for search applications with general natural language data and have already been extensively evaluated for their quality to re-rank. Hugging Face has published a selection of cross-encoders fine-tuned on MS MARCO, an IR corpus that was created based on real user search queries using the Bing search engine (Reimers, 2022). However, the MS MARCO dataset is a question-answering dataset, so it is not the optimal fit for our training objective. Therefore, we decided to compare this pre-trained model to a non-finetuned BERT model. We evaluated these to see which model performed best on the Telavox dataset.

6.1.1 Non-finetuned BERT model

For the non-finetuned BERT model, we used a model pre-trained on the English language. We implemented the first model using 'bert-base-cased' and used the auto-tokenizer that was provided by Hugging Face (Devlin et al., 2019).

6.1.2 MSMARCO finetuned model

The other model used was the pre-trained 'ms-marco-MiniLM-L-12-v2' model which was already finetuned for re-ranking. The model was provided by Hugging Face Huggingface (2022).

6.2 Implementation

A simple app was developed using React.js for the purpose of labeling the re-rank results. The interface of the app which is shown in Appendix B.1 was designed to mimic a simple search engine and showed the previous and following message to the relevant search result. The top three search results were shown and then the user had to select which one of the messages was the most relevant. The decision could be based on the query, the context of the message, and the date of the conversation. There was also a possibility to select 'None of the results were relevant'.

The back-end of the application is written in Python and for each query/document pair the application stores the MRR score, and a label representing if the document is relevant to the query or not. This data was stored in a .txt file, which is updated instantly when the value was entered via the app, using an API request to the back end. The purpose of the application is to create a conceptual app that shows how Telavox could implement the search function. Because of the file output, the application also acts as a way to quickly create training data for the cross-encoder, as for each iteration it outputs the data in the format of training input data.

6.3 Evaluation

The first step in the evaluation was to decide which retrieval model to use together with the cross-encoder, and which model to use as the cross-encoder itself. Because of time constraints, it was not possible for the Telavox annotators to annotate the results from all of the models and model combinations. Instead, we did this ourselves.

We use the non-finetuned SBERT model as the retrieval model for the re-ranking. The reason is that the non-finetuned SBERT model performed best in Chapter 4, but had a sub-optimal performance compared to the other models in Chapter 5 on the actual Telavox dataset. This model is used for retrieving the 10 candidates for the cross-encoder to re-rank.

6.3.1 Chosen cross-encoder

For the cross-encoder, we labeled and evaluated three different model setups. Together with the SBERT retriever model, we evaluated the use of no cross-encoder at all, a non-finetuned BERT model, and an MSMARCO finetuned model. The results from this are shown in Table 6.1.

Table 6.1: Table of the MRR of the different re-ranking models.

Model-type	Model-name	MRR
No cross-encoder	-	0.46
Non-finetuned cross-encoder	'bert-base-cased'	0.43
Finetuned cross-encoder	'ms-marco-MiniLM-L-12-v2'	**0.55**

The results show that the finetuned cross-encoder model received the highest MRR of 0.55. Only using the SBERT model without re-ranking received a score of 0.46. Finally, the non-finetuned cross-encoder model had the lowest score of 0.43. Therefore, we chose to continue to evaluate the 'ms-marco-MiniLM-L-12-v2' as the cross-encoder model.

6.3.2 Result

We evaluated the re-ranking results by measuring the MRR with and without the chosen 'ms-marco-MiniLM-L-12-v2' cross-encoder in the system architecture. The resulting MRR scores from the Telavox annotators are shown in Table 6.2. The MRR for the cross-encoder improved but the rank was still low considering that the best document on average was the third and last. However, this could be due to several factors including that no relevant results exist in the database. We further discuss this in Chapter 7.

Table 6.2: The MRR scores for the original and re-ranked data.

Annotator	MRR No re-ranking	MRR Cross-Encoder
Annotator 1	0.31	**0.34**
Annotator 2	0.08	**0.19**
Annotator 3	0.31	**0.36**

Chapter 7
Discussion

In this chapter, we reflect on our research findings and discuss the potential implications and limitations of using different encoding models and cross-encoders for search applications. We also identify areas for future research and discuss the broader implications of our findings for the field of NLP and search function development.

7.1 Benchmark datasets

In this section, we will dive deeper into the implications of the findings in Chapters 4 and 5, explore possible explanations for observed outcomes, and discuss the broader significance of the results.

7.1.1 Evaluation of the encoder model

From Table 4.3, we can see that our tf-idf baseline performed decently compared to the other models. The baseline did not receive the highest nor lowest score on any of the GLUE datasets but instead placed somewhere in the middle. The DPR model did not perform well in any of the tasks except the question-answering dataset QNLI, which is reasonable as it is the type of data it is trained to perform well on. The best-performing model was SBERT in all tasks except QNLI. The vector-based model, BM25, was slightly better for QNLI than SBERT. Perhaps due to the fact that many answers contain many similar words corresponding to a question and thus vector-space models can be effective at finding these similarities as they look at statistical properties of words. A possible reason why SBERT may not perform well on that dataset is that QNLI requires reasoning and inference abilities to answer the questions. In contrast, SBERT is primarily designed for semantic similarity tasks.

The results in Table 4.4 indicate that finetuning the models on the Telavox dataset did not improve the scores for any of the datasets besides the accuracy score for STS-B. The scores had remained mostly the same after the finetuning indicating that the small amount of data

did not impact the models very much. It is also interesting that the most finetuned model barely improved its accuracy scores in many of the tasks compared to the pre-trained one. This might be because the model is already trained with the chosen tasks and our additional dataset was too small for the finetuning to show a significant result in all tasks.

7.1.2 Implications of using the benchmarks

We used a selection of the existing GLUE benchmark representing different linguistic properties for semantic similarity for sentence pairs for the initial evaluation of the embedding models. One of the primary advantages of using GLUE to test embedding models is that it provides a standardized framework for comparison. This allows us to evaluate the relative strengths and weaknesses of the different models on a range of NLP tasks. Additionally, GLUE provides for a comprehensive evaluation of the models' overall performance rather than just its performance on a single task. However, one of the potential disadvantages of testing embedding models on GLUE is that it may not accurately reflect real-world performance. The tasks included in the benchmark are designed to be relatively simple and straightforward, whereas real-world language understanding tasks are often more complex and nuanced. Furthermore, the benchmark may not capture the full range of variation and diversity in natural language. This could limit its usefulness as a tool for evaluating the generalizability of different embedding models.

We also tested the models on the CISI benchmark, in order to evaluate the performance of the system as an IR system. IR was not covered by GLUE and thus we needed an additional dataset to test this. One of the primary advantages of using the CISI dataset to test embedding models is that this provides a realistic and challenging evaluation of how well a model can retrieve relevant documents given a query, making it more similar to the desired performance of our search function. Also, in comparison to the Telavox dataset, the CISI dataset contains over 100,000 query-document pairs. Thus it is also much more substantial than the Telavox dataset which was an advantage. However, one of the potential disadvantages of using the CISI dataset is that it may not be representative for all types of IR tasks. It was also challenging to find information on the dataset and how it was developed and should be evaluated.

As seen in the performances on the GLUE benchmark in Tables 4.3 and 4.4, and results for the IR retrieval on the CISI dataset in Tables 5.5 and 5.6 compared to the retrieval on the Telavox results in Tables 5.1, there are some discrepancies. Telavox showed the best results for Tf-idf while SBERT performed better on CISI. While the Telavox dataset was annotated by real users at Telavox, the small amount of data was not representative and was hard to use to justify the results based on only that dataset. The low agreement for the Telavox dataset makes us doubtful if these results can be justification for using tf-idf since we could see how well SBERT performed on CISI so it was good to have CISI as a benchmark to compare the Telavox results.

7.2 Telavox dataset

In Chapter 5, we received results from the annotation, which we did with the help of the Telavox employees. The result we retrieved was different from what we were expecting. We

expected the best performance from the finetuned SBERT model, then the non-finetuned SBERT model, and lastly the tf-idf model. However, the tf-idf model outperformed the SBERT models both in terms of R@N, MRR, P@N, and Irrelevant results. Here below we discuss what we believed led to this outcome in our results.

7.2.1 Creating the queries

We knew that the queries we chose to train the data with affected our results greatly. Due to this, we tried to make the queries broad but technical, and with the help of the Telavox employees, we tried to make sure they were relevant to the employees. In hindsight, we would have changed the queries a bit due to the fact that we created the document dataset from only chat messages from one specific group chat. Due to the messages in our dataset being only from a group chat of engineers, some of the queries were not as relevant since they would more likely appear in a private chat session. This would explain the large number of irrelevant fetched documents presented in Table 5.1. Therefore, a more suitable approach to creating the queries for the dataset would be making them more relevant for the group chats. Another approach would be collecting queries from the Telavox employees instead of generating them ourselves.

7.2.2 The annotation of the data

We found that finding a good and efficient method for annotating the data was one of the hardest things when conducting these experiments. We had the goal to get as much annotated data as possible, but since it is a time-consuming task we could not ask the employees at Telavox to annotate thousands of sentence pairs. Instead, we used Google Forms to allow for the annotation of 100 queries with ten results per query as we deemed this was an acceptable amount of data to annotate per person for the Telavox employees.

An issue we discovered with the form shown in Figure A.1 was the variation of the definition of the term 'relevant'. One of the annotators reached out during the first iteration and wanted this definition clarified. However, we decided to let every annotator have their own interpretations to mimic that they were writing the search query and performing the search themselves. Perhaps we could have clarified this or done a pilot study to make sure the annotators were more aligned with the task. However, this occurred after the tf-idf form was sent out and thus we decided to stick with our initial plan.

The Cohen's and Fleiss' kappa values were very similar between the different form iterations, with values ranging somewhere between 0.2-0.4. That the annotator agreement did not have a considerable variation between the forms possibly indicates that none of the models had an overwhelmingly clear result in regard to relevance, neither positively nor negatively. The level of agreement falls within the slight to fair range, indicating that there was room for improvement in terms of consistency and alignment among the annotators. Therefore, while the results of the Telavox dataset can be an indication of what model performs best, the low Cohen's kappa can indicate that these findings might be affected by the random choice of an annotator rather than a well-founded decision.

If we would have had a higher Cohen's kappa value between the annotators it would show that the annotators were more likely to come to the same conclusion regarding if the same

messages are relevant or non-relevant for a specific query, however, it still does not tell us if the annotators found the results to be relevant or not in general.

7.2.3 Best performing model for the Telavox dataset

In Table 5.1, the tf-idf model performs better than SBERT in terms of precision for all annotators. The average percentage of irrelevant results is high across all models and annotators, indicating that there is still room for improvement in the IR system. As previously mentioned, the high amount of irrelevant results could be due to the fact that the queries used did not have a corresponding match in the database.

Looking at Table 5.1, it is evident that the number of results considered irrelevant by the annotators significantly increased across the different forms. This discrepancy could potentially be attributed to the passage of approximately one month between the completion of the initial form and subsequent forms. It is plausible that the annotators' perception of relevance evolved over time, leading to a more stringent evaluation criteria as they encountered a diverse range of results. Another plausible explanation is that the performance of the SBERT and finetuned SBERT models were comparatively inferior to that of the tf-idf model for the Telavox dataset. Consequently, if Telavox were to develop a search engine based on this thesis, utilizing the tf-idf approach might be more advantageous if keyword and sentence matching is prioritized over context matching. This aligns with the priorities of a keyword and vector space-focused search engine, as opposed to context matching offered by the transformer-based models.

Overall, for all three models, the calculated MRR of the most relevant document is generally low across all models and annotators, indicating that the relevant documents are often ranked lower than they should be. This suggests that there is still room for improvement in the IR system, especially in terms of reducing the number of irrelevant results and improving the ranking of the most relevant documents.

If we would conduct the study again, we would have liked to explore alternative approaches for annotating the data. The chosen method possessed certain advantages, notably its simplicity and efficiency in labeling the data. However, it did not permit the annotation of data in the same volume as that achieved by well-established datasets such as the GLUE benchmark.

7.3 Re-ranking

Table 6.1 shows the evaluation of the re-ranking performance with and without a cross-encoder, using the web application. The results indicate that the re-ranking system with a cross-encoder outperforms the system without a cross-encoder for all annotators. Annotator 2 experienced a significant improvement, with the MRR score increasing from 0.08 to 0.19. Annotator 1 and Annotator 3 also experienced an improvement with increasing MRR scores for the cross-encoder from 0.31 to 0.34 and 0.31 to 0.36, respectively. These results suggest that using a cross-encoder in the system architecture can significantly improve the performance of the system, which can lead to more accurate and relevant results for the users.

The reason for the improvement being smaller in some cases could be the very small amount of training data, in relation to what other models usually are trained on. Only 100

queries with 10 messages on three models were annotated by the authors – resulting in 3000 annotated message/query combinations. Our suggestion to improve the result would be to get more annotated data to train the re-ranking model on, preferably annotated by the employees at the company.

7.4 Usage in a company domain

While SBERT can be more powerful than tf-idf in certain applications, it may not be suitable for all scenarios. For example, if the dataset is small or noisy, it can be difficult to train a good SBERT model. In that case, tf-idf may be a more robust and reliable option. Additionally, tf-idf is more interpretable and easier to explain to non-experts, which can be important in many practical applications. Overall, while SBERT has its advantages, tf-idf can be a simple and effective choice for many IR tasks. In order to still utilize the language understanding properties of the transformer architecture, a solution would be to combine a bag-of-words-based model together with a transformer-based cross-encoder. The system would retrieve documents based on textual similarity and re-rank the results based on semantic similarity.

7.5 Future work

There were further steps that we would have liked to take if we had had more time.

Firstly, finetuning the SBERT model on more data. For this, we would both want to use more datasets in general, not only data from the company, and to use more annotators to achieve more accurate results. Another possibility would be to use different queries and or use data from private chats or other group chats in order to increase the amount of data available.

One area for future research is to test the performance of different encoding models and cross-encoders on larger and more diverse datasets, potentially including data from multiple sources and chats. Other types of queries could also possibly be used, covering more subject areas. Additionally, more labeled data could be collected to improve the accuracy of the evaluation.

Another potential avenue for further exploration is to investigate the performance of other types of models and architectures for search applications, such as graph-based models or models that incorporate external knowledge sources.

Furthermore, it would be interesting to explore different evaluation metrics that capture other aspects of search performance, such as click-through rate and user satisfaction, which could provide a more comprehensive picture of the strengths and limitations of different search approaches.

Chapter 8
Conclusions

In this thesis, we conducted a comprehensive evaluation of transformer-based models in the context of search applications within a company domain. Our evaluation encompassed multiple aspects of performance assessment, allowing us to gain a holistic understanding of the models' effectiveness.

Initially, we assessed the models' capabilities using the established GLUE benchmark, which encompasses a wide range of NLP tasks testing semantic similarity. Among the models considered, the SBERT encoder model exhibited superior performance in four out of five GLUE datasets, surpassing the performance of both alternative transformer-based models and traditional bag-of-word approaches. The findings of our study provided compelling evidence supporting the utilization of transformer-based models as encoder models, resulting in enhanced search performance compared to vector-based models on the chosen GLUE datasets.

Subsequently, we finetuned the best-performing SBERT model and evaluated its effectiveness in IR tasks, using both the CISI benchmark and a newly created dataset specific to the company domain. We then compared it to the non-finetuned model and tf-idf baseline. These evaluations allowed us to measure the models' retrieval performance in real-world scenarios and assess their ability to handle complex search tasks within the company context. Notably, we observed that the finetuning the SBERT model did not improve performance on the GLUE datasets significantly. Additionaly, both SBERT models performed sub-optimally on the company dataset while the SBERT model was the best on the CISI benchmark. For the company, the tf-idf model demonstrated the best performance in IR but the annotation agreement showed a bad correlation between annotators and thus we can not confidently justify the findings on this dataset.

Lastly, we investigated the potential improvements in performance achieved by implementing a cross-encoder in conjunction with a bi-encoder architecture. By incorporating a cross-encoder model, we aimed to optimize the overall performance while maintaining computational efficiency. Our results indicate that cross-encoders generally outperform sentence transformers in IR tasks, and can be beneficial to use for improving the performance

on domain-specific datasets.

Nevertheless, it is crucial to acknowledge the limitations inherent in our research. Firstly, our dataset was relatively small and confined to chat messages from a single group chat, potentially limiting its representativeness for all types of user conversations. Secondly, the absence of a robust inter-annotator agreement presents challenges in generalizing our findings for the domain-specific dataset.

In conclusion, our findings highlight the continued need for exploration and advancements in NLP-based search applications. Through our study we have shown how transformers can be implemented to solve a number of tasks and using it in search functions can help users navigate through large amounts of data. By addressing the identified limitations and engaging in further research and development, substantial enhancements in the usability and effectiveness of search functions across diverse contexts can be achieved.

References

AL-Smadi, M., Jaradat, Z., AL-Ayyoub, M., and Jararweh, Y. (2017). Paraphrase identification and semantic text similarity analysis in arabic news tweets using lexical, syntactic, and semantic features. *Information Processing & Management*, 53(3):640–652.

Briggs, J. (Collected 2023). Semantic search: Measuring meaning from jaccard to bert. `https://www.pinecone.io/learn/semantic-search/`.

Clough, P. and Sanderson, M. (2013). Evaluating the performance of information retrieval systems using test collections. *Information Research*, 18(2):1 – 10.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364.

Craswell, N. (2009). *Encyclopedia of Database Systems*, chapter Precision-Oriented Effectiveness Measures, pages 2128–2129. Springer US, Boston, MA.

Dean, B. (2020). How people use google search (new user behavior study). `https://backlinko.com/google-user-behavior`.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ghojogh, B. and Ghodsi, A. (2020). Attention Mechanism, Transformers, BERT, and GPT: Tutorial and Survey. *OSF Preprints*.

Huggingface (2022). Cross-encoder for ms marco. `https://huggingface.co/cross-encoder/ms-marco-TinyBERT-L-2`.

Huggingface (2023). Sentence transformers. `https://huggingface.co/sentence-transformers`.

Hussan, B. K. (2020). Comparative Study of Semantic and Keyword Based Search Engines. *Advances in Science, Technology and Engineering Systems Journal*, 5(1):106–111.

Indurkhya, N. and Damerau, F. (2010). *Handbook of Natural Language Processing, Second Edition*. Taylor & Francis.

Jesus-German, O.-B., Gemma, B.-E., and Helena, G.-A. (2022). Sentence-crobi: A simple cross-bi-encoder-based neural network architecture for paraphrase identification. *Mathematics*, 10(3578):3578.

Karpukhin, V., Oguz, B., Min, S., Wu, L., Edunov, S., Chen, D., and Yih, W. (2020). Dense passage retrieval for open-domain question answering. *CoRR*, abs/2004.04906.

Korman, D. Z., Mack, E., Jett, J., and Renear, A. H. (2018). Defining textual entailment. *Journal of the Association for Information Science and Technology*, 6(6).

Kundu, R. (2022). F1 score in machine learning: Intro & calculation. *V7*. `https://www.v7labs.com/blog/f1-score-guide`.

Lample, G. and Conneau, A. (2019). Cross-lingual language model pretraining. *CoRR*, abs/1901.07291.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

Lukovnikov, D., Fischer, A., and Lehmann, J. (2019). Pretrained transformers for simple question answering over knowledge graphs. In Ghidini, C., Hartig, O., Maleshkova, M., Svátek, V., Cruz, I., Hogan, A., Song, J., Lefrançois, M., and Gandon, F., editors, *The Semantic Web – ISWC 2019*, pages 470–486, Cham. Springer International Publishing.

Muennighoff, N. (2022). SGPT: GPT Sentence Embeddings for Semantic Search. *arXiv e-prints*, page arXiv:2202.08904.

Qaiser, S. and Ali, R. (2018). Text mining: Use of tf-idf to examine the relevance of words to documents. *International Journal of Computer Applications*, 181.

Reimers, N. (2022). Retrieve & re-rank. `https://www.sbert.net/examples/applications/cross-encoder/README.html`,.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084.

Salton, G. and Lesk, M. E. (1965). The smart automatic document retrieval systems—an illustration. *Commun. ACM*, 8(6):391–398.

scikit learn (2023). Machine learning in python. `https://scikit-learn.org/stable/index.html`.

Tien, N. H., Le, N. M., Tomohiro, Y., and Tatsuya, I. (2019). Sentence modeling via multiple word embeddings and multi-level comparison for semantic textual similarity. *Information Processing & Management*, 56(6):102090.

Tunstall, L., von Werra, L., and Wolf, T. (2022). *Natural Language Processing with Transformers: Building Language Applications with Hugging Face*. O'Reilly Media, Incorporated.

University of Glasgow (Collected 2023). School of computing science. `https://www.gla.ac.uk/schools/computing/research/researchsections/ida-section/informationretrieval/#popularresources`.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2019). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*. In the Proceedings of ICLR.

Yamoun, L., Guessoum, Z., and Girard, C. (2022). Transformer RoBERTa vs. TF-IDF for websites content-based classification. In *Deep Learning meets Ontologies and Natural Language Processing, International Workshop in conjunction with ESWC*, Hersonissos, Greece.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Zhang, X., Yates, A., and Lin, J. (2021). Comparing score aggregation approaches for document retrieval with pretrained transformers. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 – April 1, 2021, Proceedings, Part II*, page 150–163, Berlin, Heidelberg. Springer-Verlag.

Appendices

Appendix A
Google Form for Annotating Telavox Dataset.

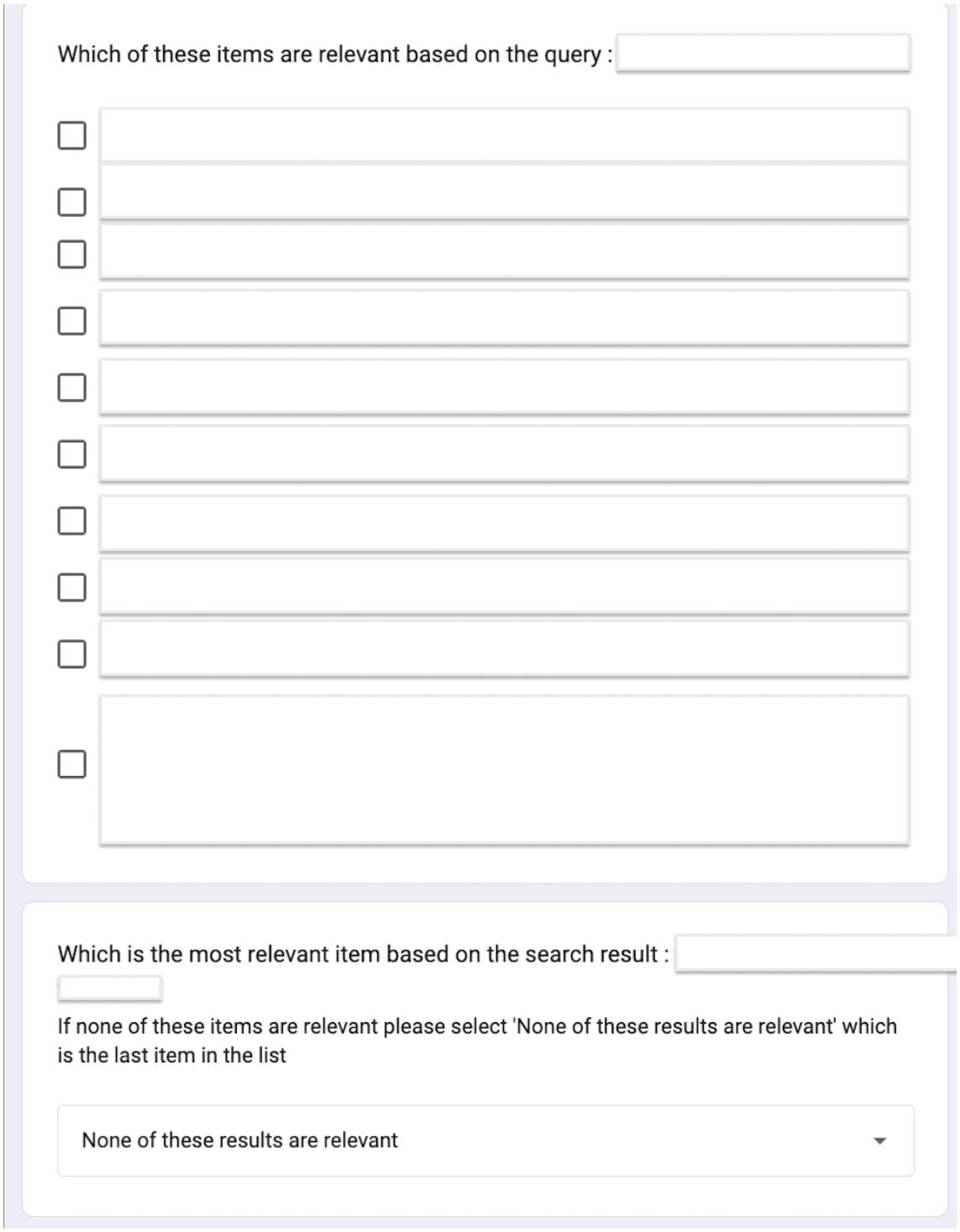


Figure A.1: Image of the Google Form that was used for annotating data. Queries and search results are not shown for data privacy reasons.

Appendix B
Screenshot of Application

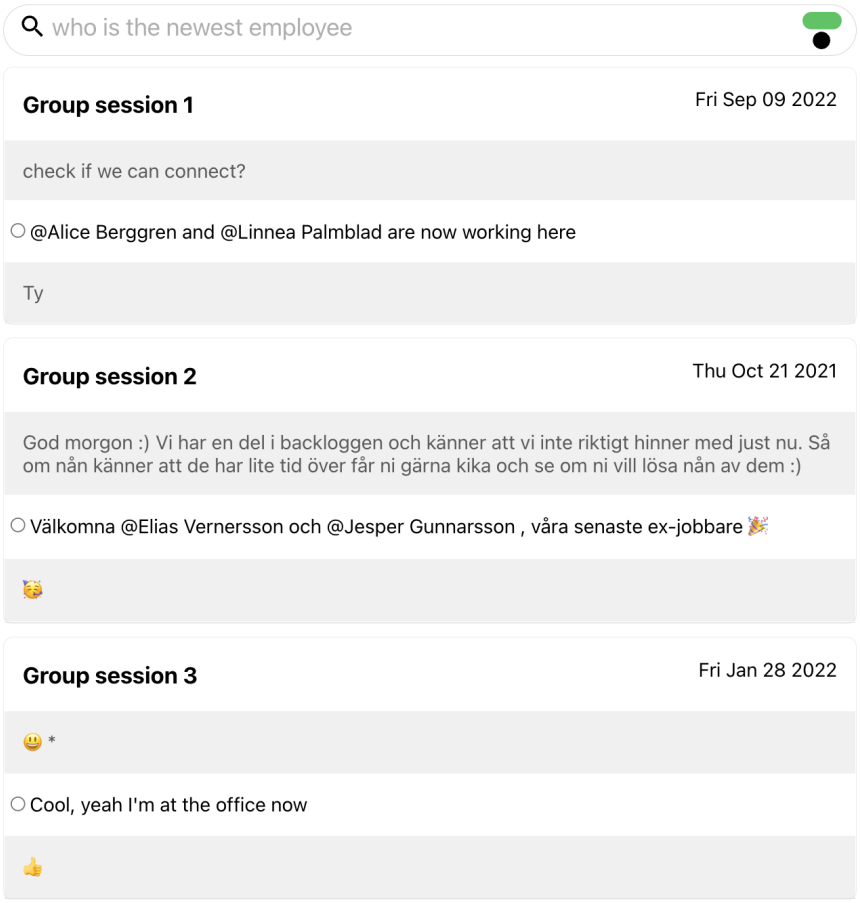


Figure B.1: A screenshot of the re-ranking application searching for the query *'who is the newest employee'* using an example dataset.

EXAMENSARBETE Using Transformers to improve Search functions
A study on a smart messaging system
STUDENTER Alice Berggren, Linnea Palmblad
HANDLEDARE Pierre Nugues - (LTH)
EXAMINATOR Jacek Malec - (LTH)

Tekniken att jämföra äpplen och äpplen

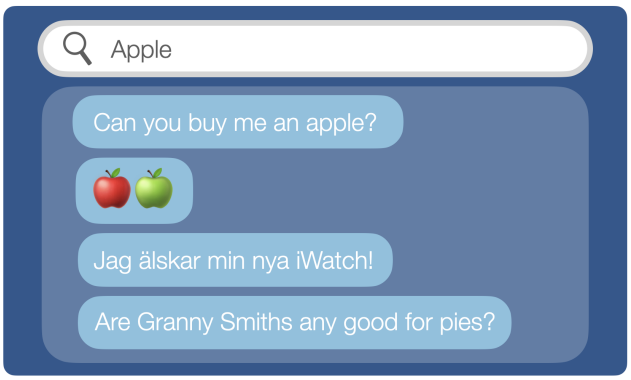
POPULÄRVETENSKAPLIG SAMMANFATTNING **Alice Berggren, Linnea Palmblad**

Sökfunktioner spelar en avgörande roll i dagens digitala landskap genom att erbjuda användarna effektiva sätt att hitta önskad information bland en omfattande mängd data. Examensarbetet undersöker transformersarkitekturen och ser hur den jämför sig med mer traditionella metoder.

Språkteknologi utgör en gren inom maskininlärning som ägnar sig åt att utveckla metoder och modeller för att möjliggöra maskiners förmåga att förstå och tolka mänskligt språk. Med tanke på den ökande kommunikationen via internet har språkteknologin fått en betydligt större inverkan än någonsin tidigare. Inom ramen för informationssökning kan språkteknologin spela en avgörande roll för att säkerställa att rätt innehåll når rätt människor. Transformers är en state-of-the-art modell inom maskininlärning och har visat sig vara revolutionerande inom förståelse av kontext inom språk, vilket är väldigt användbart inom sökmotorer.

I en nyckelordsbaserad sökning där en mening innehåller ordet 'Apple' skulle systemet inte kunna urskilja mellan frukten eller teknikföretaget. En transformer modell hade kollat på meningen och kontexten orden befinner sig i, för att förstå innebörden, vilket gör den så banbrytande. Trots att denna teknik är välkänd, är det inte alla företag som besitter den nödvändiga kompetensen som krävs för att implementera dessa modeller.

I detta examensarbete har vi undersökt olika transformers modeller och jämfört dessa med mer konventionella metoder. Först testade vi fem olika modeller på hur väl dom kan avgöra om två olika meningar är relaterade. Sedan testades modellerna på hur väl de kan ranka dokument i en samling baserat på en sök-term. Detta gjordes på ett etablerat dataset och på ett företags egna dataset som vi fick skapa från grunden genom deras databas. Sedan tillämpades metoden re-



ranking för att förbättra sökresultaten genom att lägga till en ytterligare modell för att placera de mest relevanta sökresultaten högst upp. Den mest framstående modellen identifierades och valdes.

Resultaten visar att transformers är överlägsna traditionella modeller, då användningen av en transformer-baserad modell både var mest effektiv på att bedöma relationen mellan två olika meningar, och i kombination med en annan transformer-baserad modell på att placera de mest relevanta sökresultaten högst upp.