

**ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA**

**DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING**

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Natural Language Processing

**DESIGN AND IMPLEMENTATION OF A
REAL-WORLD SEARCH ENGINE BASED ON
OKAPI BM25 AND SENTENCEBERT**

CANDIDATE

Lorenzo Bonetti

SUPERVISOR

Prof. Paolo Torroni

Academic year 2020-2021

Session 2nd

Contents

1	Introduction	1
2	Background	3
2.1	Ranking	3
2.1.1	Query expansion	4
2.1.2	Document expansion	5
2.2	Semantic search	7
2.2.1	The advent of BERT	8
3	Problems and approaches	10
3.1	Limits of keyword matching	10
3.2	Limits of semantic search	11
3.3	The challenge of evaluating commercial application systems	12
4	Technologies	14
4.1	OkapiBM25	14
4.2	SentenceBERT	16
4.3	FAISS	19
4.4	T5 and docTTTTTquery	20
4.4.1	docTTTTTquery	22
5	The hybrid model	23
5.1	BM25	23
5.1.1	Data preprocessing	24

5.1.2	BM25 score	25
5.2	SentenceBERT	26
5.3	The question generator	26
5.3.1	Filtering questions	27
5.4	The final score	29
5.5	Speeding up search	30
6	Experiments and validation	31
6.1	Experiments	31
6.2	Validation	33
6.3	Results	36
7	Conclusion	39
	Bibliography	41

List of Figures

2.1	Nogueira et al. document expansion [8]	6
2.2	Soyeong et al. unsupervised document expansion [9]	6
4.1	TF-IDF's term frequency vs. BM25's term frequency [15]	16
4.2	SBERT architecture with classification objective function [1]	17
4.3	SBERT architecture with regression objective function [1]	18
4.4	SBERT architecture with triplet objective function [16]	19
4.5	T5 baseline model architecture [21]	21
4.6	T5 workflow [21]	22
5.1	Data preprocessing example	25
5.2	Document expansion with generated question	27
5.3	Question generation pipeline	28
6.1	A search result example	36
6.2	Starting paragraph	37
6.3	All the questions	37
6.4	Example of a group of similar questions	38
6.5	Most meaningful questions	38

List of Tables

6.1	Results on 50 questions from a client banking dataset	34
6.2	BeIR benchmark datasets	35
6.3	Results comparison between stand-alone approaches and hybrid model on BeIR benchmark datasets	36

Chapter 1

Introduction

The work conducted in this thesis aims to present an hybrid model for a real-world application search engine. The project presented was part of an internship work carried out in a start-up which deals with Knowledge Management and Artificial Intelligence. The aim of the internship work was to improve the current search engine system to build a new system for a future web application use case. An in-depth study on the limitations of keyword search alone, and on semantic search, revealed the need of a transition from a pure keyword-based information retrieval system to an hybrid model, making use of both keyword search and semantic search. In particular the old system relied on a tfidf-based algorithm, while the final model tries to overcome the limits of keyword search by joining the abilities of OkapiBM25, a probabilistic information retrieval approach, with newer semantic search models based on SentenceBERT [1]. The models, and the algorithm implemented, exploit deeply recent techniques in Information Retrieval such as lexical search, similarity search, query expansion, document expansion and automatic question generation. The data used to test the models came from a banking dataset, belonging to one of the company clients, previously created for an Information Retrieval chat-bot. Different experiments led to a final model able to improve the search performances showing great advantages with respect to keyword search and pure semantic search.

Below the structure of the thesis is presented:

- In the second chapter an introductory background is presented varying between multiple topics strictly related to Information Retrieval systems and NLP
- In the third chapter the problems arised, and the approaches taken to try to tackle them, are shown
- In the fourth chapter the technologies involved in the model and in the algorithm are explained in detail
- The fifth chapter shows the actual implementation of the system
- The last chapter is left for presenting the experiments and showing the results obtained

Chapter 2

Background

2.1 Ranking

Ranking is one of the most important tasks in the Information Retrieval field. It is the process of sorting documents according to the relevance with a particular search query. The goal is to assign a score to each document and successfully retrieve the most relevant one that should contain the answer to the user query. In order to do so many NLP techniques are used nowadays starting from basic keyword matching to complex algorithms and models which exploit deeply the world of semantics and keyword search. We can date the birth of the ranking task back in 1958, when Luhn et al. proposed to consider “statistical information derived from word frequency and distribution... to compute a relative measure of significance” [2] anticipating the technique of TF-IDF which is now the basis of many ranking systems. Later on Maron and Kuhns depicted the problem as receiving requests and try “to provide as an output an ordered list of those documents which most probably satisfy the information needs of the user” [3]. In their work they suggested to weight terms in documents according to the probability that a user desiring information contained in a particular document would use that term in a query, introducing what nowadays is likelihood. In 1975 finally, after gradually accumulating notions and techniques in the past decades, Salton [4] proposed

the use of vector space model, where documents and queries are represented by vectors and their similarity is computed in terms of cosine similarity. In the 1980s and 1990s many studies were done in the direction of finding alternative term weighting schemes starting from the TF-IDF function. One of the most promising and most used in many Information Retrieval systems nowadays is the Okapi BM25 [5] which introduces some free parameters in the classic TF-IDF formula to change how terms occurring extra times in a document add extra score. BM25 takes also into account the length of the document in the computation of the score to assign, penalizing indeed the documents that contain the user query keywords if their length is much greater than the average length in the dataset. The mechanics and the main difference with respect to TF-IDF will be further explained in the next chapters.

2.1.1 Query expansion

Query expansion is the process of reformulating a user query in order to better understand the underlying context and improve search performance in information retrieval. The goal is to minimize the query-document mismatch selecting and adding terms to the user query. Query expansion approaches contain two major classes: global approaches and local approaches. The first class of approaches refers to using global available thesaurus in order to expand the query with synonyms and related words. Wordnet [6] is the most famous English-based lessical database which can provide general similar words. Local approaches instead tend to reformulate the user query adding terms related to the top retrieved documents. This approach was first presented by Attar and Frankael in 1977 [7]. The effectiveness of the technique is obviously highly influenced by the proportion of relevant documents in the high ranks. Let a user query consist of n terms

$$Q = \{t_1, t_2, \dots, t_n\}$$

the reformulated query becomes

$$Q_{ref} = (Q - T'') \cup T'$$

where $T' = \{t'_1, t'_2, \dots, t'_n\}$ is the set of additional terms and $T'' = \{t''_1, t''_2, \dots, t''_n\}$ are stopwords to be removed.

2.1.2 Document expansion

Document expansion is another technique used to reduce the vocabulary mismatch between query and documents that are lexically different but semantically similar. Document expansion works by explicitly generate lexically richer documents, introducing two main advantages with respect to query expansion: the first one is that expanding documents can generate much more relevant terms given the longer text, the second one is that document expansion is a process that can be entirely done at indexing time, avoiding delays at search time. In “Document Expansion by Query Prediction” [8] Nogueira et al. propose a document expansion technique based on predicting queries starting from a document and expanding the document by concatenating the predicted questions.

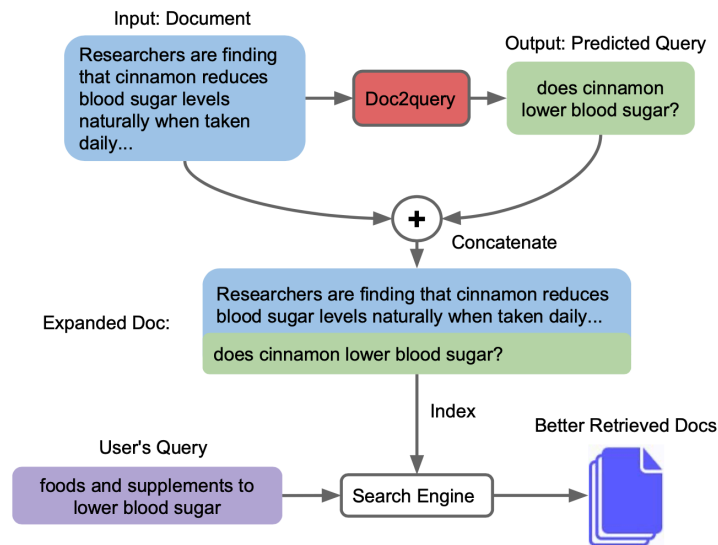


Figure 2.1: Nogueira et al. document expansion [8]

Soyeong Jeong et al. [9] try to address the problem using an unsupervised process in which they generate document-related sentences using a pre-trained language model for summarization. The supplementary sentences are then appended to the input document before indexing them.

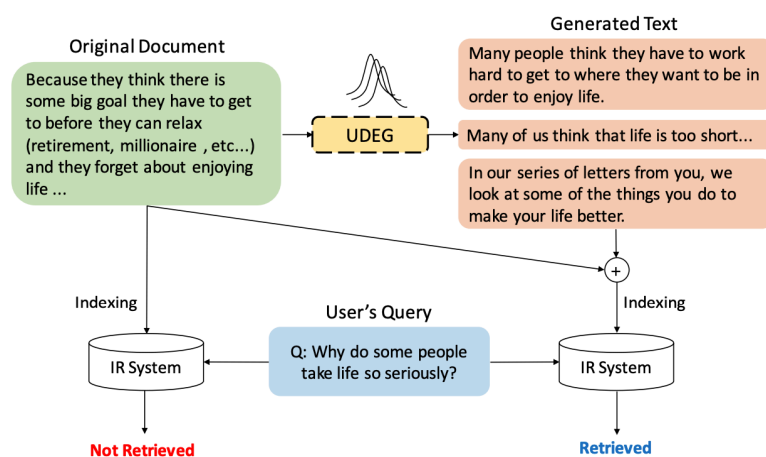


Figure 2.2: Soyeong et al. unsupervised document expansion [9]

2.2 Semantic search

While keyword search tends to find exact match of queries in the documents without actually understanding the meaning of them, semantic search tries another approach to tackle this task in a more “human” way. The idea behind semantic search is to embed all entries in a corpus, whether they be sentences, paragraphs, or documents, into a vector space. At search time, the query is embedded into the same vector space and the closest embeddings from the corpus are found. These entries should have a high semantic overlap with the query. Until 2013 Google searches didn’t take into account the actual meaning of the query but the search was performed to return the more accurate exact matching of the user query. Then Google introduced a Knowledge Graph with the Hummingbird update [10] making a smooth transition from keyword search to actually derive semantics in the search process making a huge advancement in their search technology. Hummingbird sought to solve the problem of finding information when your knowledge on the topic was lacking by focusing on synonyms and theme-related topics. Moreover, by being able to use natural language processing, search results would be able to retrieve nice results for queries both at the head and long-tail level. Understanding the question in a semantic manner, Hummingbird sought to allow users the ability to confidently search for topics and sub-topics rather than having to engineer queries. In the latest years many researches were made in the field of semantic search, and the recent discoveries made in the Deep Learning field boosted even more the hype for these new technologies. The 2017 publication of Vaswani et al. “Attention is all you need” [11] started a real revolution in many AI fields, bringing to the NLP world the innovation of Transformers. The capabilities of the novel architecture proposed by Vaswani revealed themselves right away and not many years later a new model architecture was presented, becoming the base architecture for SOTA results in many NLP tasks: BERT.

2.2.1 The advent of BERT

BERT [12] is a trained Transformer Encoder stack (12 layers for Base version, 24 in the large one) trained on Wikipedia and Book Corpus, a dataset containing more than ten thousand books of different genres. Bert uses a Masked Language Model (MLM) which randomly masks some of the tokens from the input, and predicts the masked word based on its surroundings (left and right of the word). As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the MLM objective enables the representation to use both the left and the right context, which allows to pre-train a deep bidirectional Transformer. Additionally, BERT also uses a Next Sentence Prediction (NSP) task during the pre-training phase. This makes BERT a contextual model, capable of handling relationships between multiple sentences by representing each word differently based on the rest of the sentence in a bidirectional way. BERT works similarly to the Transformer encoder stack, by taking a sequence of words as input which keep flowing up the stack from one encoder to the next, while new sequences are coming in. The final output for each sequence is a vector of 768 numbers in Base or 1024 in Large version.

BERT arrived on the scene in October 2018 and it immediately became clear how powerful it was and how it would change the world of NLP and IR. The first application of BERT to text ranking was reported by Nogueira and Cho [13] in January 2019 on the MS MARCO passage ranking test collection [14]. Within less than a week, effectiveness shot up by around ~30% in relative gain. BERT-based architecture are still on top of many leaderboards for various NLP tasks, from question answering to semantic similarity search, proving once again the State-Of-The-Art results this language representation model achieved. An important distinction to make, when dealing with semantic similarity search, is related to the relative size of queries and documents.

Knowing in advance the shapes of corpus and queries can make a huge difference on the model selection and on search performances. We can distinguish semantic search into two sub-fields: symmetric semantic search and asymmetric semantic search.

- **symmetric semantic search:** the query and the entries in the corpus are of about the same length and have the same amount of content. An example of symmetric search is looking for similar questions. For symmetric tasks, one could potentially flip the query and the entries in the corpus
- **asymmetric semantic search:** usually there is a short query (like a question or some keywords) and the task is to find a longer paragraph answering the query. For asymmetric tasks, flipping the query and the entries in the corpus usually does not make sense.

The pretraining task, and in particular the dataset used, influence the ability of a certain model to better perform in one kind of search instead of the other. BERT models pretrained on the MS MARCO dataset for example are more suited for asymmetric semantic search given that the dataset consists in question answering taken from real Bing questions and passages. Models pretrained on Natural Language Inference tasks instead, tend to better perform when queries are more or less of the same length of the documents, thus belonging in symmetric semantic search.

Chapter 3

Problems and approaches

3.1 Limits of keyword matching

Many information retrieval systems retrieve relevant documents based on exact matching of keywords between a query and documents. This method degrades precision rate, ending in the “vocabulary mismatch problem”. One approach to bridge the gap between query terms and document terms is **enriching query representations**. Query expansion techniques help to add relevant terms that were not presented initially to improve the effectiveness of the search. A simple way to do that is expanding query with alias terms. Alias are not only synonyms but also words that are strictly related, in a semantic way, to the original query keyword. In a keyword matching system, using alias may help finding results when the user’s knowledge doesn’t reflect the documents’ one. Furthermore, other techniques such as document expansion make a step towards trying to solve the “vocabulary mismatch” problem between query and documents. Document expansion, similarly to query expansion, aims at generating lexically richer documents acting directly on the corpus. The main problem of keyword matching however is not entirely solved with query and document expansion means but more powerful semantic search techniques are needed to improve the accuracy of the system. This is where semantic space

steps in to create a rich dense representation where terms and sentences are located in a specific place, and their position assumes an important value. Unlike keyword search, semantic search takes into consideration the researcher's intent to get at the contextual meaning of terms. Semantic search pushes beyond the boundaries of the organization's collective base of understanding to get at information and concepts that haven't been explicitly written into the query. Semantic technology deciphers concepts and meaning by associating search inputs with clarifying terms such as related synonyms that have been built into the system.

3.2 Limits of semantic search

We have seen how semantic search can solve some of the problems related to keyword search, but new limits related to this field arise. Semantic search improves the understanding of the meaning of the query but, in order to do that, the query needs to be good. The result of a semantic search can only be as good as the question asked. The more information users communicate in their search request, the greater the chances of the search engine to deliver good results. Short head queries, for example, provide very limited information about what the user is really looking for and may behave even worse than a simple keyword matching search. This is the reason why in this thesis work, it was chosen to maintain both the techniques and try to aggregate the different results taking the benefits of the two approaches. The use-case taken into consideration can't take advantage in knowing in advance the users and their content. This poses an additional difficulty in the setup of the development environment. Many factors may influence the quality of the search engine results, such as:

- **Volume of data:** the number of documents uploaded may differ from user to user. With the growth of the knowledge base, the performance may decrease in terms of response time and general accuracy

- **Specific domain:** even if the latest semantic search models reached SOTA results in many NLP tasks these were obtained on general-purpose dataset. Domain specific documentation and user queries may lead to unexpected poorer results

3.3 The challenge of evaluating commercial application systems

The performance evaluation of an information retrieval system is a decisive aspect for the measure of the improvements in search technology. However, when dealing with commercial application, common evaluation metrics may not be enough. Quite often, state of the art models, are not so reliable when applied to commercial use cases. Common benchmarks tend to evaluate performance of models on the broadest and as general as possible domains in order to keep a standard and uniform baseline. This however does not perfectly reflect the real world use cases and advanced models need to be carefully refined to obtain good results. A simple example concerns the language of the documents we are evaluating. Recently, pre-trained language models have achieved remarkable success in a broad range of natural language processing tasks. The BERT model we used, for example, was trained on the MS MARCO passage re-ranking task. However, in multilingual setting, it is extremely resource-consuming to pre-train a deep language model over large-scale corpora for each language. Instead of exhaustively pre-training monolingual language models independently, an alternative solution is to pre-train a powerful multilingual deep language model over large-scale corpora in hundreds of languages. However, the vocabulary size for each language in such a model is relatively small, especially for low-resource languages. This limitation inevitably hinders the performance of these multilingual models on tasks

such as sequence labeling, wherein in-depth token-level or sentence-level understanding is essential. This problem brought us to the choice of working with only English contents, at least in an initial setup, to avoid language-related loss of accuracy.

Another important metric to keep in consideration when evaluating commercial application systems is time. Both training and inference time can play a very important role in assessing the quality of a search engine. The use of deep learning models helps finding more relevant content, but on the other side it adds a substantial overhead in terms of processing time. For a simple comparison, for BM25, the time complexity is $O(m \cdot avgdl)$, where m is the number of documents and $avgdl$ is the average document length. BERT, on the other hand, needs to compute self-attention which has a complexity per layer of $O(d * n^2)$ where d is the embedding dimension. n^2 because it computes attention weights for each word with respect to every other word: $O(n)$ operations for each word and therefore $O(n^2)$ for all the words. To reduce as much as possible the waiting time, we took different choices for training and inference:

- **Training time:** to optimize the embeddings time at training time, the model is deployed with a simple API. The server uses different workers to handle more requests concurrently, speeding up the process at the expenses of additional computational power
- **Inference time:** the main problem at inference time, is the time spent in computing similarity between query and document embeddings. As we will see more in detail in the next sections, this problem has been addressed thanks to an external library to optimize search between vectors and reduce computational time.

Chapter 4

Technologies

4.1 OkapiBM25

The BM25 is a ranking function used by search engines to estimate the relevance of documents to a given search query and is often referred to as “Okapi BM25” since the Okapi information retrieval system was the first system implementing this function. The BM25 retrieval formula belongs to the BM family of retrieval models (BM stands for Best Match). It is a variance of the more famous TF-IDF algorithm which, given a query and a set of documents, assigns a score to each document based on the sum of the term frequencies of the words appearing in the query multiplied by the inverse document frequency, namely:

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

In particular, the term frequency of a word is defined as the number of occurrences of that word in a certain document divided by the length of the document

$$tf_{i,j} = \frac{n_{i,j}}{|d_j|}$$

while the Inverse Document Frequency is a number representing the rarity

or commonness of a certain word across all the documents

$$idf(t, D) = \log \frac{|D|}{1 + |\{d \in D : t \in d\}|}$$

The OkapiBM25 version brings some innovation to the TF-IDF formula starting from the Inverse Document Frequency. The main difference is that the IDF in the OkapiBM25 takes into account frequency of terms and penalizes terms that are common:

$$IDF(q_i) = \ln\left(1 + \frac{docCount - f(q_i) + 0.5}{f(q_i) + 0.5}\right)$$

where docCount is the total number of documents and $f(q_i)$ is the number of documents containing term q_i . The Term Frequency part of the formula becomes:

$$TF(q, d) = \frac{f(q_i, d) \cdot (k_1 + 1)}{f(q_i, d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{avgdl})}$$

where $f(q_i, d)$ is the term frequency of the term q_i in document d , avgdl is the average document length in the collection and k_1 and b are free parameters, for this project the default values have been chosen as $k_1 = 1.2$ and $b = 0.75$. The effect of this modified term frequency is clearly visible in figure 3.1 where the impact of really common words inside a document tend to saturate faster the value of the term frequency so that terms occurring extra times do not add too much extra score.

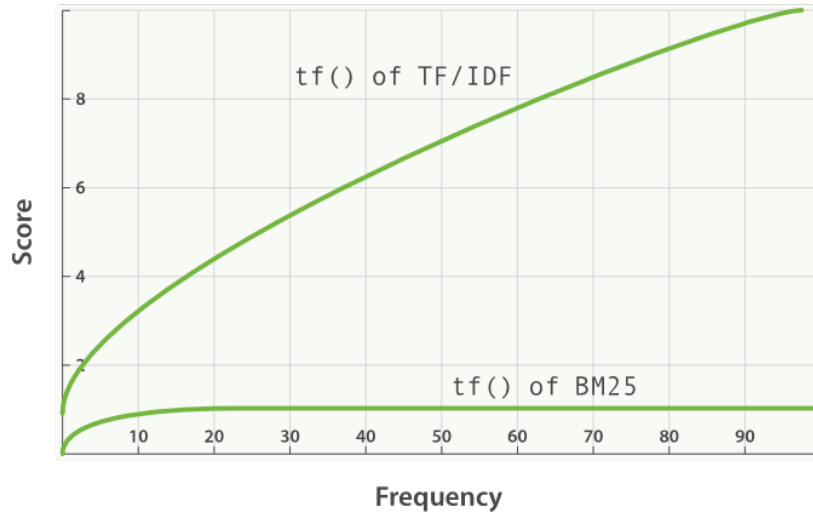


Figure 4.1: TF-IDF's term frequency vs. BM25's term frequency [15]

4.2 SentenceBERT

SentenceBERT is a modification of the BERT network presented by Nils Reimers and Iryna Gurevych in 2019. A large disadvantage of the BERT network structure is that no independent sentence embeddings are computed, which makes it difficult to derive sentence embeddings from BERT. To bypass these limitations, researchers passed single sentences through BERT and then derived a fixed sized vector by either averaging the outputs (similar to average word embeddings) or by using the output of the special CLS token. SentenceBERT uses siamese and triplet networks to derive semantically meaningful sentence embeddings. A siamese neural network is a class of neural network architectures that contain two or more identical subnetworks. Not only they have the same configurations but they also share the same weights. This guarantees that two semantically similar sentences are mapped by each network to very close locations in the feature space. It comes in handy when dealing with one-shot classification problems and finding similar elements in a large amount of

data (the complexity for finding the most similar sentence pair in a collection of 10,000 sentences is reduced from 65 hours with BERT to 5 seconds with SBERT) [1]. The model adds a pooling operation to the output of BERT to derive a fixed-size sentence embedding. In the paper 3 different pooling strategies have been experimented: using the output of the CLS-token, computing the mean of all output vectors (MEAN-strategy), and computing a max-over-time of the output vectors (MAX-strategy). The default configuration, and the one used in this work, is the MEAN-strategy. Also for the training of the model 3 different objective functions have been exploited:

- **Classification Objective Function:** the sentence embeddings \mathbf{u} and \mathbf{v} are concatenated with the element-wise difference $|\mathbf{u}-\mathbf{v}|$ and multiplied with the trainable weight $W_t \in R^{3n \times k}$

$$o = \text{softmax}(W_t(u, v, |u - v|))$$

where n is the dimension of the sentence embeddings and k the number of labels. The loss used is the cross-entropy loss.

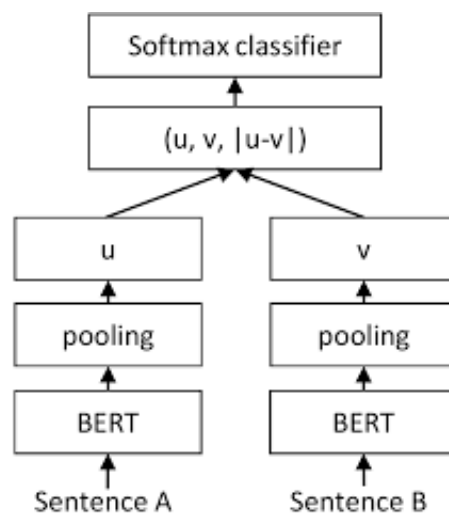


Figure 4.2: SBERT architecture with classification objective function [1]

- **Regression Objective Function:** The objective function is the **meansquared-error loss** with respect to the cosine similarity between the two sentence embeddings **u** and **v**

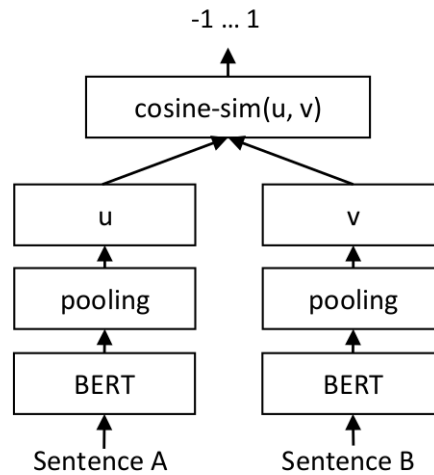


Figure 4.3: SBERT architecture with regression objective function [1]

- **Triplet Objective Function:** Given an anchor sentence **a**, a positive sentence **p**, and a negative sentence **n**, triplet loss tunes the network such that the distance between **a** and **p** is smaller than the distance between **a** and **n**. It minimizes the following function:

$$\max(\|s_a - s_b\| - \|s_a - s_n\| + \epsilon, 0)$$

with s_x the sentence embedding for a/n/p, $\|\cdot\|$ a distance metric and margin ϵ . Margin ϵ ensures that s_p is at least ϵ closer to s_a than s_n . In the experiment the Euclidean distance is used as metric and ϵ is set to 1.

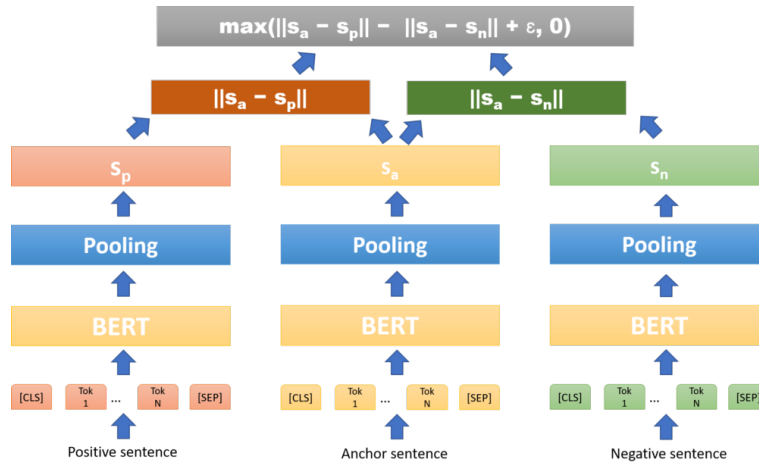


Figure 4.4: SBERT architecture with triplet objective function [16]

4.3 FAISS

FAISS [17] is a library for efficient similarity search and clustering of dense vectors developed by Facebook AI Research [18]. Given a set of vectors x_i in dimension d , Faiss builds a data structure in RAM from it. After the structure is constructed, when given a new vector x in dimension d it performs efficiently the operation:

$$j = \operatorname{argmin}_i \|x - x_i\|$$

where $\|\cdot\|$ is the Euclidean distance (L^2). In our case the x_i vectors are the embeddings produced by SentenceBERT of the documents and d is their fixed size of 768. The new vector x is the query embedded again by the SentenceBERT model. Faiss is built around an index type that stores a set of vectors, and provides a function to search in them with L2 and/or dot product vector comparison. To actually compute the cosine similarity we first need to normalize our embedding vectors and then proceed with the dot product. Some index types are simple baselines, such as exact search. Most of the available indexing structures correspond to various trade-offs with respect to

- search time

- search quality
- memory used per index vector
- training time
- need for external data for unsupervised training

In our case we decided to opt for the *IndexFlatIP* which provides exact search for inner product performing an exhaustive search. FAISS has been used to optimize the time of response of our system by speeding up the search and computation of cosine similarity between query embedding and documents embeddings.

4.4 T5 and docTTTTTquery

Encoder-only models like BERT are designed to produce a single prediction per input token or a single prediction for an entire input sequence. This makes them applicable for classification or span prediction tasks but not for generative tasks like translation or abstractive summarization.

T5, or Text-To-Text Transfer Transformer, is a novel neural network model that uses a text-to-text approach, presented by a Google research team in the paper “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer” in April 2020. Any NLP task can be reframed into a unified text-to-text-format where the input and output are always text strings, in contrast to BERT-style models that can only output either a class label or a span of the input. To train T5, the team of researcher introduced a new open-source pre-training dataset, called the Colossal Clean Crawled Corpus (C4) [19] based on Common Crawl dataset [20]. The Common Crawl corpus contains petabytes of data collected since 2008. It contains raw web page data, extracted metadata and text extractions, ending up with 750 gigabytes of clean-ish English text. The model architecture is a standard encoder-decoder transformer, as proposed

by Vaswani et al [11]. Specifically the encoder and decoder consist of 12 blocks that comprehend self-attention, optional encoder-decoder attention and a feed-forward network. In particular the feed-forward network is constituted by a dense layer followed by a ReLU activation layer and lastly another dense layer. Totally, the number of parameters of the model is around 220 millions, approximately twice as much as a BERT base model (having 2 stacks instead of 1 as in the BERT base model).

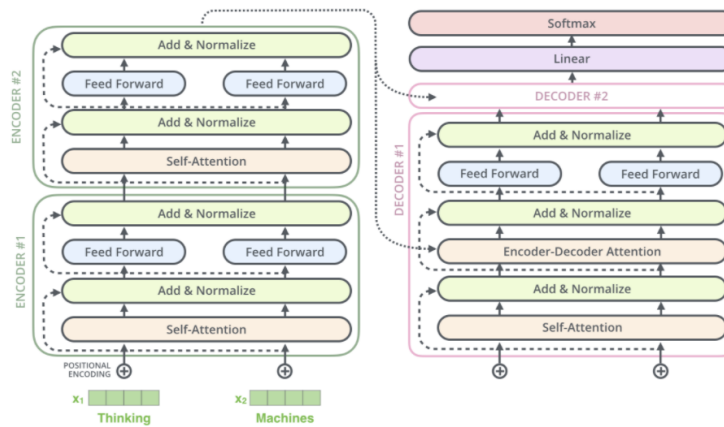


Figure 4.5: T5 baseline model architecture [21]

In the pretraining phase, the model has been trained on text-to-text only tasks, allowing to use a maximum likelihood objective (teacher forcing [22]) and cross-entropy loss. The pretraining was performed for $2^{19} = 524,288$ steps on C4 before fine-tuning. Packing multiple sequences in batches led to having batches containing 2^{16} tokens, resulting in pre-training on $\sim 2^{35}$ tokens. T5 is then finetuned on a variety of downstream task such as GLUE [23] , SQuAD [24] for question answering and CNN/DM [25] dataset for summarization.

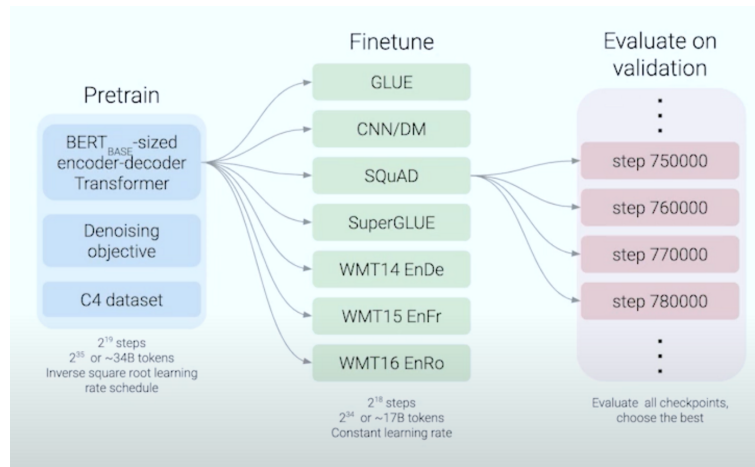


Figure 4.6: T5 workflow [21]

4.4.1 docTTTTTquery

DocTTTTTquery or docT5 query is the latest version of the doc2query family of document expansion models presented by Rodrigo Nogueira and Jimmy Lin in 2019 [26]. The basic idea is to train a model that, when given an input document, generates questions that the document might answer (or more broadly, queries for which the document might be relevant). The model is an extension to the author’s previous work doc2query [8]. Essentially in the new version they substitute the vanilla transformer with the T5 base model resulting in large effectiveness gains. The model has been trained on the MS MARCO passage re-ranking dataset obtained from the top-10 results retrieved by the Bing search engine (from 1M queries). The authors found out that using top_k sampling helped produce more efficient queries, using k=10. Similarly to our intended use, in their work the expansion model is used to predict questions or queries and append them back to the documents before indexing them in a ranking task. In our case the model has been used to predict questions for the documents but instead of appending them to the documents they have been used separately to compute an additional score and to guide the search for users.

Chapter 5

The hybrid model

In order to rank a set of documents given a search query we need to assign a score to each document reflecting the importance and relation of it with respect to the user search. In this chapter we'll see the architecture and the steps of the algorithm used in this project. As already stated in the previous section, it is a hybrid model based both on the BM25 ranking algorithm and on SentenceBERT. The following sections will explain in details how the two different parts work by assigning a score to each document autonomously and how the results are then aggregated.

5.1 BM25

The first score assigned to a document is solely based on syntactic features and doesn't explore the semantic world. It is based on a statistical information retrieval technique which is the BM25 algorithm. To address some of the issues arising from a mismatch between different word forms used in the queries and the relevant documents, a preliminary step needed is the data preprocessing.

5.1.1 Data preprocessing

Researchers have long proposed the use of various stemming algorithms to reduce terms to a common base form. Before doing that all the text contained in the documents has been lowercased. Lowercasing is quite a drastic option since it takes away all the information related to entities, names, acronyms ecc. but, since the average user will use lowercase words to make its search, it is better to lowercase everything to avoid mismatches. Then the Porter Stemmer algorithm [27] is performed to reduce words to their stem and thus have a common form. When working with TF-IDF like algorithms, common words are already penalized by their great occurrences but, removing all the stopwords is a cleaner and more efficient way to improve the quality of the search. The list of stopwords has been taken from the Natural Language Toolkit library [28]. The final step in the Preprocessing phase is to take the list of words lowered, stemmed and without stopwords and create a second list of ngrams containing both unigrams and bigrams. The use of bigrams is needed to create more unique and complex keywords that will guide the search towards more precise results. All the steps of data preprocessing are applied to the user query as well, so to have a set of keywords comparable to the now processed documents. While in a previous setup a further step of query enrichment was performed, to add synonyms and relative words from a common thesaurus, in the final evaluation of the model it was preferred to only keep original query terms. This choice was taken after weeks of human evaluation, the main reasons after this come from the specific use cases of the model and the related dataset. Using synonyms and relative tended to stray results from what the user was looking for.

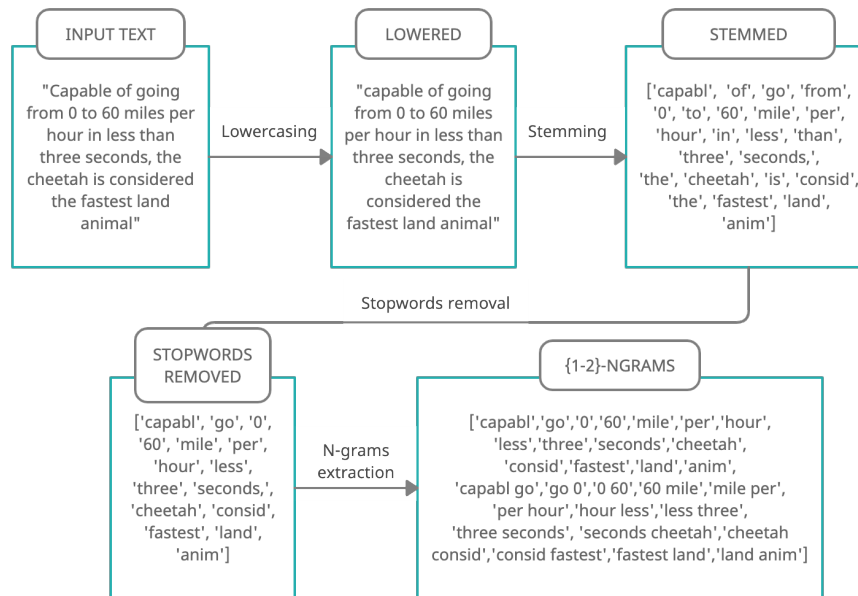


Figure 5.1: Data preprocessing example

5.1.2 BM25 score

Once the data has been preprocessed, the application of BM25 can take place. For each document the BM25 score is computed according to the following formula:

$$\text{BM25}(d) = \sum_{i=1}^N \text{IDF}(q_i) \cdot \frac{f(q_i, d) \cdot (k_1 + 1)}{f(q_i, d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{\text{avgdl}})}$$

where q_i are the keywords in the user query, namely the unigrams and bigrams remaining after the cleaning and preprocessing phase. $\text{IDF}(q_i)$ is the Inverse Document Frequency of BM25 already explained in chapter 3.1. The BM25 score is a value which is not upperly bounded and can be greater than 1. Since in our scenario this score will have to be weighted with the one obtained with cosine similarity (thus between 0 and 1) the following step to perform is a normalization in order to bring the values between 0 and 1.

5.2 SentenceBERT

To work on the semantic features of the sentences it has been chosen to go with SentenceBERT. SentenceBERT is a modification of the more famous BERT architecture that uses a siamese network to derive semantically meaningful sentence embeddings. Sentence embeddings can be used for similarity comparison tasks such as clustering and information retrieval via semantic search. In our case the SentenceBERT model was used to compute embeddings both for the user query and for each document, such that looking for similar sentences becomes looking for close embeddings vectors. The metric used to assess the proximity of vectors was the Cosine Similarity which has proven to be really effective when dealing with vector distances. The cosine similarity measures the angle between two vectors, the closer they are in the semantic field the smaller the angle is and thus the higher the cosine (at most 1):

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

Computing the cosine similarity between the vector embedding of the user query and the vector embeddings of the documents gives as a result a vector of values between 0 and 1 that, once sorted descendingly, is a ranked list of relevant documents according to the user query.

5.3 The question generator

To improve the quality of the results of the search engine the use of an automatic question generator has been adopted in order to try to emulate human-askable questions. The questions' goal is not only to help guide the search to the engine, but also to be displayed to the user to help him find the information he was looking inside the document. The model used to generate question is a pretrained model, based on Google's T5, namely the docT5query. These predicted questions will then be used to compute an additional score that will

be integrated in the final score of the document. The idea is to generate a fair amount of questions per document, embed them as usual with the SentenceBERT model, and compare their embeddings with the user query one at inference. The comparison is computed via the cosine similarity which will again assign a score between 0 and 1 to the question. Since more than one question is generated per document, only the highest question score is considered regarding the relevance to the user query.

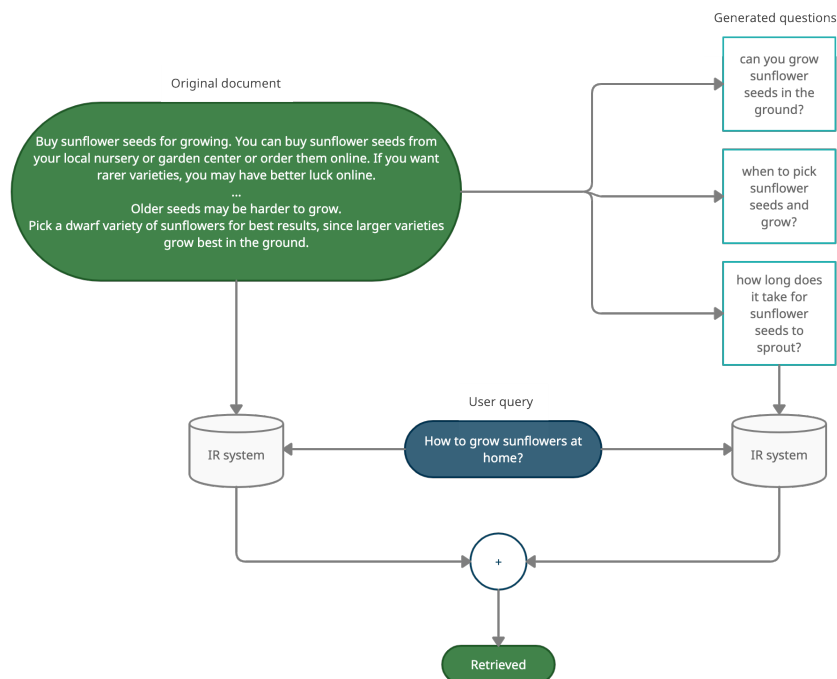


Figure 5.2: Document expansion with generated question

5.3.1 Filtering questions

The output of the question generator is very often too vague, resulting in many unnecessary questions. Since the questions will be displayed directly to the user they need to be as worthwhile as possible. To filter out some of the questions a simple approach has been taken to assign a score to each question and filter the one that does not comply with a particular threshold. Due to some

stochasticity in the generation process, running the question generator multiple times, without fixing the random seed, actually produces different questions. In the first step the questions generator is run 5 times to output a wide list of questions to choose from. Then a score is assigned to each question based on the relevance with respect to the input text. This score is the cosine similarity between the embedding vectors of the question and the input text. The embeddings are computed using the same SentenceBERT model which is used later to compute the ranking score. In order to avoid having too similar questions, a further step is computed to filter them out again. The questions are embedded with the **distilbert-base-nli-stsb-quora-ranking** [29] SentenceBERT model, a version trained on the Quora Duplicate questions dataset [30]. The questions are finally grouped by similarity (computing cosine similarity with the new embeddings) and only the best question per group (according to the relevance with the paragraph) is kept.

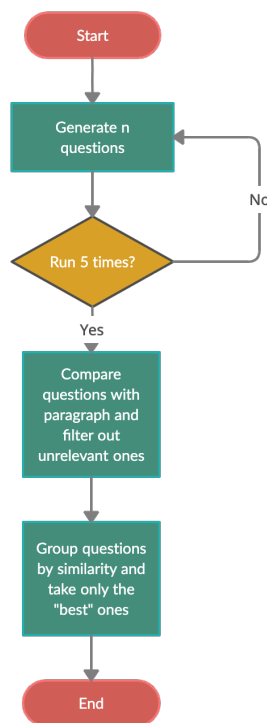


Figure 5.3: Question generation pipeline

5.4 The final score

As already stated, the final score assigned to a document is a weighted sum of the two scores previously described. In particular the BM25 score needs to be normalized in order to be compared and summed with the cosine similarity score. So the first thing to do is divide all the BM25 documents scores by the maximum obtained:

$$\text{BM25 (d)} = \frac{\text{BM25 (d)}}{\max_{d \in D} \text{BM25 (d)}}$$

When dealing with search engines between documents, titles have their importance. Instead of including the title of a document in its content it has been treated separately, having its own BM25 score and its own cosine similarity score. Afterwards the title score has been integrated with the content score with a weighted sum:

$$\text{BM25(d)} = \text{BM25_title(d)} \cdot \alpha_1 + \text{BM25_content(d)} \cdot (1 - \alpha_1)$$

The same thing is performed with the SBERT score:

$$\text{SBERT(d)} = \text{SBERT_title(d)} \cdot \alpha_1 + \text{SBERT_content(d)} \cdot (1 - \alpha_1)$$

where SBERT_title(d) is the cosine similarity between the embedding vector of the title of the document and the user query and SBERT_content(d) is the cosine similarity between the embedding vector of the content of the document and the user query.

While BM25 has not been applied to the generated questions, the SentenceBERT model has been used to compute their cosine similarity with respect to the user query. This value has been taken into consideration while computing the SBERT score only if greater of a certain threshold:

$$\text{SBERT}(d) = \begin{cases} \text{SBERT}(d) \cdot 0.5 + \text{q_score} \cdot 0.5, & \text{if } \text{q_score} \geq \text{q_threshold} \\ \text{SBERT}(d), & \text{otherwise} \end{cases}$$

The final score can then be computed by simply a weighted sum of the two intermediate scores:

$$\text{Score}(d) = \text{BM25}(d) \cdot \alpha_2 + \text{SBERT}(d) \cdot (1 - \alpha_2)$$

5.5 Speeding up search

As previously mentioned, we made use of the FAISS library for storing the vectors of the embeddings and performing search on optimized indexes. In particular we created three different indexes: one for the title, one for the bodies of the contents and one for the questions. The three indexes store the embeddings vector in an optimized data structure which is furtherly saved on disk and retrieved at runtime. Since the index we used (*IndexFlatIP*) belongs to the class of indexes that only performs dot product, we first need to normalize our embeddings vectors. FAISS provides a function that computes the L2 norm of the vectors and normalizes them before adding them to the index. Then at run-time the dot product is computed on the normalized index which result in the same result of computing cosine similarity:

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{A}{\|A\|} \cdot \frac{B}{\|B\|} = \text{dot}\left(\frac{A}{\|A\|}, \frac{B}{\|B\|}\right)$$

Chapter 6

Experiments and validation

6.1 Experiments

In this chapter we will present experiments conducted and the result obtained in the implementation of the information retrieval system that has been described in the previous sections. This section will also examine the validation results and the delicate process of how they have been obtained.

The final use case for the implementation is a web application providing a service of research inside personal knowledge base for the users. The functioning is pretty straightforward: the user first uploads its own documents to the platform, and later can make use of the practical search bar to obtain search results, taken from the previously uploaded documents.

During the whole experimentation period, many different models have been tried and tested. The first implementation was a simple TF-IDF-based search engine. As discussed before, this kind of model could not grasp the meaning of queries thus ending in poor performances quite often (in particular when working with italian dataset which have an expressive richness). In the second approach, the first hybrid model came to life: the score given to a certain document was a weighted sum between the TF-IDF score and the cosine similarity between embeddings. The first BERT model taken into consideration was the '**distiluse-base-multilingual-cased-v2**', a model not exactly

suites for semantic search, and in particular for asymmetric semantic search. The performance of the hybrid model didn't improve that much, which led to further studying and two main modifications for the third model. First the basic TF-IDF implementation was substituted with the Okapi BM25 algorithm, secondly, after a more in depth analysis of the various transformers models, the 'distiluse-base-multilingual-cased-v2' was replaced by the 'msmarco-distilbert-base-v2'. The two changes actually improved the system by benefiting from the better techniques and from a more suited model. The expansion of the document with generated questions had two main reasons to be investigated: the first was to actually see if the use of questions improved the quality of the search engine, the second was a need coming from the UX design of the web application. The first question generator was based on T5-small, in particular it was the **valhalla/t5-base-qg-hl**, a pretrained version for question generation. Due to the poor results obtained, a custom one was trained using t5-large as base model and running a finetuning task following Patil Suraj work [31] (the author of the valhalla model). The finetuning task was performed on SQuAD v1, using the dataset in a text-to-text approach: given the passage the model will try to output the questions. Despite the difference in terms of parameters (from ~60M in T5-small to ~220M in T5-base) the results did not show any improvement in terms of quality of text. The last model taken into consideration, and the one that is actually now in use, was a pretrained model still based on T5: the 'castorini/docTTTTTquery'. The key in the model, we believe, was the dataset used to train it: once again the MS MARCO dataset. Also, as they explain in the paper, the goal was to produce a question generation model able to extract "queries that will be issued for a given document" which reflected more accurately our goal. As a confirmation of what we found out in our tests, Nogueira et al. used the T5 base model and not larger ones since they did not notice any improvement in retrieval effectiveness. The final experiments were done in the direction of improving the quality of the questions, and this was achieved with the filtering operations

applied, as described in the previous section.

6.2 Validation

The main source of validation of the experiments was human evaluation operated by the Quality Assurance (QA) team. The way these tests were performed changed depending on the different use-cases, dataset and task. For the information retrieval system the QA team had a custom dataset containing queries collected both from previous real users (clients mainly) and from the team itself. The system was evaluated assigning a score to each query result ranging from 1 to 5 according to some criteria:

- 5: the most related document is in first position, and it contains the answer to the user query
- 4: the most related document appears in the first four positions, and it contains the answer to the user query
- 3: a document related to the user query appears in the first four positions, it does not contain exactly the answer to the user query but in a more broad sense it still gives enough information about it
- 2: the most related document appears between the fifth and the eight positions
- 1: the most related document does not appear in the first eight positions

The dataset used for this test was a banking documents dataset, obtained by one of the clients of the company. The dataset is composed of 326 heterogeneous documents with an average length of 1681 words. The documents were taken from a section of the client's website, in order to build an IR chat-bot. The queries used to test the retrieval system were collected in two different ways:

- 25 questions have been created by the QA team which built up the documents dataset in the first place

- 25 questions were queries asked by the chat-bot clients, namely common people utilizing their website, accurately filtered and chosen by the QA team

The most successful model obtained an average score of 4,02 out of 5, improving the result of the current system in use of 3,38. In particular these were the results:

	Old model	First model	Second model	Final model
Answers scored 5	15	16	23	29
Answers scored 4	14	13	9	10
Answers scored 3	6	7	7	2
Answers scored 2	5	5	4	1
Answers scored 1	10	9	7	8

Table 6.1: Results on 50 questions from a client banking dataset

About the quality of the generated questions, the tests were defined to divide questions in 4 different categories:

- **Non sense questions:** these were questions either not correct from a grammatical point of view, or totally unrelated to the document
- **True for every content:** these questions were too general and ended up being true for every content, thus not adding any information to the document
- **Good questions:** these were actually good questions that could in some way help the search of the user
- **Brilliant questions:** these were extremely good questions, even if asked by humans, that rarely appeared

Lastly, the final model was tested against some famous benchmarks for information retrieval tasks. The model was evaluated through BeIR [32], a heterogeneous benchmark containing different datasets to use as benchmarks. In particular the datasets used are the following:

- **SciFact**: a collection of 1.4K expert-written scientific claims paired with evidence-containing abstracts, and annotated with labels and rationales [33]
- **SCIDOCS**: an evaluation benchmark consisting of seven document-level tasks ranging from citation prediction, to document classification and recommendation [34]
- **NFCorpus**: a full-text English retrieval data set for Medical Information Retrieval. It contains a total of 3,244 natural language queries (written in non-technical English, harvested from the NutritionFacts.org site) with 169,756 automatically extracted relevance judgments for 9,964 medical documents (written in a complex terminology-heavy language), mostly from PubMed [35]
- **FiQA-2018**: a Question Answering dataset for the financial domain. It contains roughly 6,000 questions and 57,000 answers [36]

The tests were done considering a document retrieval “accurate” if the result appeared among the top 10 results of a query. Due to the high resource requirements, some of the datasets have not been exploited entirely but only a portion of them has been used.

Dataset	Website	Queries size	Corpus size
SciFact	github.com/allenai/scifact	300	5K
SCIDOCS	allenai.org/data/scidocs	1,000	25K
NFCorpus	cl.uni-heidelberg.de/statnlpgroup/nfcorpus/	323	3.6K
FiQA-2018	sites.google.com/view/fiqa/	648	57K

Table 6.2: BeIR benchmark datasets

The tests were performed to highlight the improvement of the hybrid model with respect to the two stand-alone approaches:

Dataset	BM25 accuracy	Sbert accuracy	Hybrid model accuracy
SciFact	0.69	0.613	0.72
SCIDOCS	0.438	0.431	0.502
NFCorpus	0.65	0.613	0.66
FiQA-2018	0.339	0.484	0.487

Table 6.3: Results comparison between stand-alone approaches and hybrid model on BeIR benchmark datasets

6.3 Results

In this last section we will present some of the results both of the search engine and the questions generator process.

The following image shows an example of query and the corresponding results of the search engine with an uploaded knowledge base about cryptocurrencies.

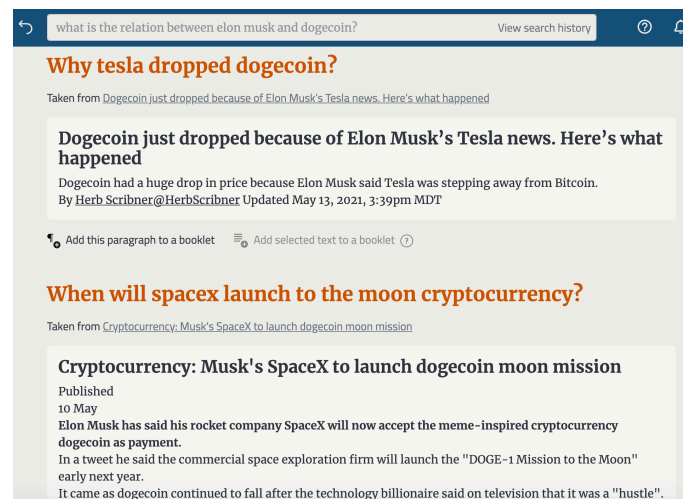
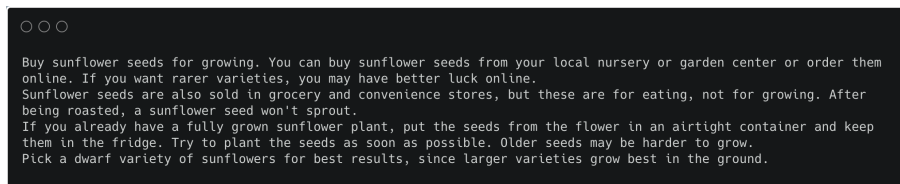


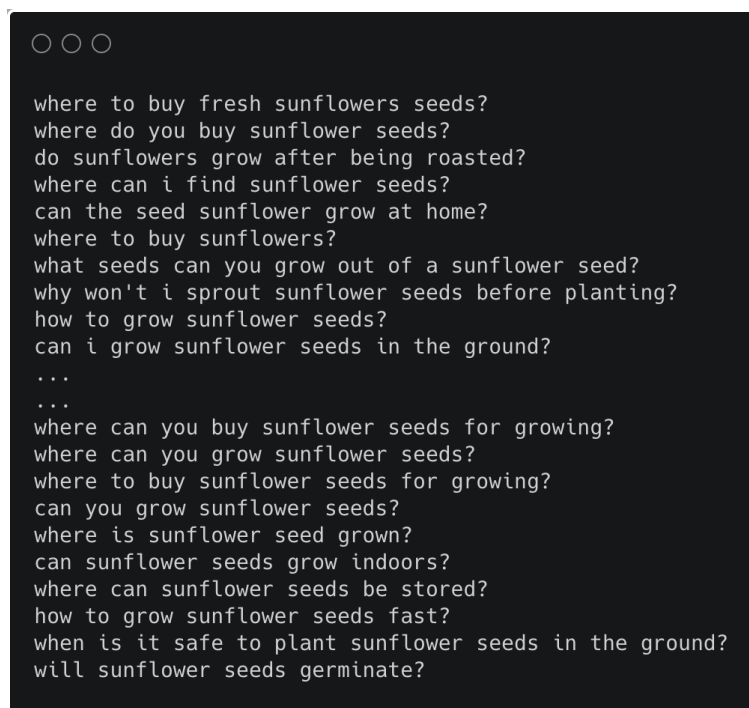
Figure 6.1: A search result example

In the question generation process we first start from a paragraph and generate 50 questions related to it:



Buy sunflower seeds for growing. You can buy sunflower seeds from your local nursery or garden center or order them online. If you want rarer varieties, you may have better luck online. Sunflower seeds are also sold in grocery and convenience stores, but these are for eating, not for growing. After being roasted, a sunflower seed won't sprout. If you already have a fully grown sunflower plant, put the seeds from the flower in an airtight container and keep them in the fridge. Try to plant the seeds as soon as possible. Older seeds may be harder to grow. Pick a dwarf variety of sunflowers for best results, since larger varieties grow best in the ground.

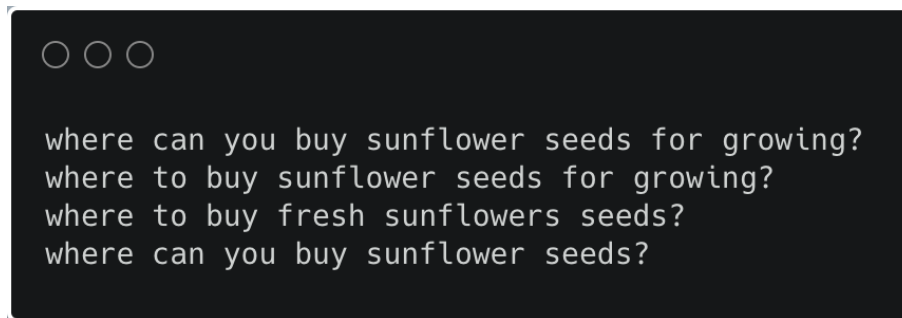
Figure 6.2: Starting paragraph



where to buy fresh sunflowers seeds?
where do you buy sunflower seeds?
do sunflowers grow after being roasted?
where can i find sunflower seeds?
can the seed sunflower grow at home?
where to buy sunflowers?
what seeds can you grow out of a sunflower seed?
why won't i sprout sunflower seeds before planting?
how to grow sunflower seeds?
can i grow sunflower seeds in the ground?
...
...
where can you buy sunflower seeds for growing?
where can you grow sunflower seeds?
where to buy sunflower seeds for growing?
can you grow sunflower seeds?
where is sunflower seed grown?
can sunflower seeds grow indoors?
where can sunflower seeds be stored?
how to grow sunflower seeds fast?
when is it safe to plant sunflower seeds in the ground?
will sunflower seeds germinate?

Figure 6.3: All the questions

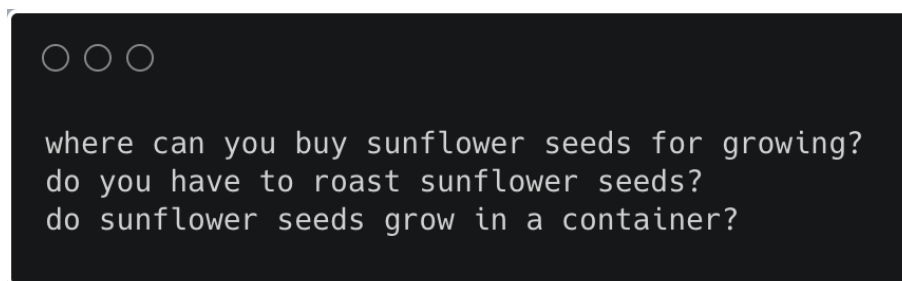
Then, after removing the duplicates, similar questions are aggregated in groups after comparing all of them one by one.



```
○ ○ ○  
where can you buy sunflower seeds for growing?  
where to buy sunflower seeds for growing?  
where to buy fresh sunflowers seeds?  
where can you buy sunflower seeds?
```

Figure 6.4: Example of a group of similar questions

For each group only the question with the highest score is kept, i.e. the one with the highest relevance with respect to the starting paragraph. Lastly, the questions are filtered again, keeping only the ones that have a score above a predefined threshold, leaving us with just a bunch of meaningful and useful queries.



```
○ ○ ○  
where can you buy sunflower seeds for growing?  
do you have to roast sunflower seeds?  
do sunflower seeds grow in a container?
```

Figure 6.5: Most meaningful questions

Chapter 7

Conclusion

The main goal of this thesis work was to present an hybrid model, exploiting keyword search and semantic search for a web application use case. The need of a transition from a pure keyword-based algorithm to an hybrid one, arised from an in depth study conducted on the limitations of the previous approaches. The hybrid model made use of the latest techniques developed in the NLP and IR fields. For the keyword search the model took advantage of the Okapi BM25 algorithm in order to assign meaningful weights to words and terms in documents. For the semantic search, the model used the ability of SentenceBERT to derive semantically meaningful sentence embeddings and use them to encode documents and queries for further vector comparisons. This project showed the limits, both in keyword and semantic search, and the techniques used to try to address them such as query expansion and document expansion. To enrich and expand the documents it was made use of a questions generator, a supplemental model based on T5. The questions generator has been used not only to enrich documents and thus improving search performances but also as part of the search experience from the user point of view. Furthermore, an external library was used to address the problem of speed when dealing with large amount of vectors, and computing distances among them. The model proved to be effective, overcoming the results obtained by the two single stand alone approaches on the BeIR benchmark. Bearing in

mind the challenges of evaluating an information retrieval system for a commercial application, the model has been tested also by human evaluators on a specific domain dataset with a custom set of queries. Through this custom dataset obtained from a client of the company, it was made possible to test the model for the search efficiency. The tests showed not only an improvement both with respect to keyword search and semantic search stand alone approaches, but also with respect to the previous algorithm in use. The thesis work ended demonstrating the improvements brought by the joint approach of keyword and semantic search both in terms of accuracy and user satisfaction.

Bibliography

- [1] N. Reimers and I. Gurevych. Sentence-bert: sentence embeddings using siamese bert-networks, August 2019.
- [2] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165, 1958. DOI: 10.1147/rd.22.0159.
- [3] M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *J. ACM*, 7(3):216–244, July 1960. ISSN: 0004-5411. DOI: 10.1145/321033.321035. URL: <https://doi.org/10.1145/321033.321035>.
- [4] G. Salton, A. Wong, and C.-S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, 1975.
- [5] F. Crestani, M. Lalmas, C. J. Van Rijsbergen, and I. Campbell. “is this document relevant?...probably”: a survey of probabilistic models in information retrieval. *ACM Comput. Surv.*, 30(4):528–552, December 1998. ISSN: 0360-0300. DOI: 10.1145/299917.299920. URL: <https://doi.org/10.1145/299917.299920>.
- [6] C. Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.
- [7] R. Attar and A. S. Fraenkel. Local feedback in full-text retrieval systems. *J. ACM*, 24(3):397–417, July 1977. ISSN: 0004-5411. DOI: 10.1145/322017.322021. URL: <https://doi.org/10.1145/322017.322021>.

- [8] N. Rodrigo, Y. Wei, L. Jimmy, and C. Kyunghun. Document expansion by query prediction, December 2019.
- [9] S. Jeong, J. Baek, C. Park, and J. Park. Unsupervised document expansion for information retrieval with stochastic text generation, May 2021.
- [10] How the google hummingbird update changed search. URL: <https://www.searchenginejournal.com/google-algorithm-history/hummingbird-update/#close>.
- [11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17*, pages 6000–6010, Long Beach, California, USA. Curran Associates Inc., 2017. ISBN: 9781510860964.
- [12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics, June 2019. DOI: 10.18653/v1/N19-1423. URL: <https://aclanthology.org/N19-1423>.
- [13] R. Nogueira and K. Cho. Passage re-ranking with bert. *ArXiv*, abs/1901.04085, 2019.
- [14] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. Ms marco: a human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268, 2016. URL: <http://dblp.uni-trier.de/db/journals/corr/corr1611.html#NguyenRSGTMD16>.
- [15] S. Connelly. The bm25 algorithm. URL: <https://www.elastic.co/blog/practical-bm25-part-2-the-bm25-algorithm-and-its-variables>.

- [16] C. Vu. URL: <https://towardsdatascience.com/a-complete-guide-to-transfer-learning-from-english-to-other-languages-using-sentence-embeddings-8c427f8804a9>.
- [17] Facebook ai similarity search. URL: <https://github.com/facebookresearch/faiss>.
- [18] Facebook ai. URL: <https://ai.facebook.com/>.
- [19] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL: <http://jmlr.org/papers/v21/20-074.html>.
- [20] Common crawl. URL: <https://commoncrawl.org/>.
- [21] Colin raffel: exploring the limits of transfer learning with a unified text-to-text transformer. URL: <https://www.youtube.com/watch?v=eKqWC577WlI&list=UUEqgmyWChwvt6MFGG1mUQCQ&index=7>.
- [22] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989. DOI: 10.1162/neco.1989.1.2.270.
- [23] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. GLUE: a multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics, November 2018. DOI: 10.18653/v1/W18-5446. URL: <https://aclanthology.org/W18-5446>.
- [24] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250,

2016. arXiv: 1606.05250. URL: <http://arxiv.org/abs/1606.05250>.
- [25] R. Nallapati, B. Zhou, C. dos Santos, Ç. Gülçehre, and B. Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics, August 2016. DOI: 10.18653/v1/K16-1028. URL: <https://aclanthology.org/K16-1028>.
- [26] R. Nogueira and J. Lin. From doc2query to docttttquery, December 2019.
- [27] C. van Rijsbergen, S. Robertson, and M. Porter. New models in probabilistic information retrieval, 1980.
- [28] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. ” O’Reilly Media, Inc.”, 2009.
- [29] Distilbert-base-nli-stsb-quora-ranking. URL: <https://huggingface.co/sentence-transformers/distilbert-base-nli-stsb-quora-ranking>.
- [30] First quora dataset release: question pairs. URL: <https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs>.
- [31] Question generation using transformers. URL: https://github.com/patil-suraj/question_generation/.
- [32] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych. BEIR: a heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL: <https://openreview.net/forum?id=wCu6T5xFjeJ>.

-
- [33] D. Wadden, S. Lin, K. Lo, L. L. Wang, M. van Zuylen, A. Cohan, and H. Hajishirzi. Fact or fiction: verifying scientific claims. In *EMNLP*, 2020.
- [34] A. Cohan, S. Feldman, I. Beltagy, D. Downey, and D. S. Weld. Specter: document-level representation learning using citation-informed transformers. In *ACL*, 2020.
- [35] V. Boteva, D. Gholipour, A. Sokolov, and S. Riezler. A full-text learning to rank dataset for medical information retrieval. In 2016. URL: <http://www.cl.uni-heidelberg.de/~riezler/publications/papers/ECIR2016.pdf>.
- [36] Financial opinion mining and question answering. URL: <https://sites.google.com/view/%EF%AC%81qa>.