Trabajo Fin de Máster:

Master Dissertation : Information Retrieval for Question Answering based on Distributed Representations

Ana Sagrado Sala

Trabajo de investigación para el

Máster en Lenguajes y Sistemas Informáticos

Universidad Nacional de Educación a Distancia

Supervisors

Dr. Álvaro Rodrigo Yuste y Dr. Fernando Lopez Ostenero

February 2022

Abstract

Commonly used methods for information retrieval such as TF-IDF do not capture the semantics of the query or the document. This is a problem, especially in cases where the words used in the queries are not contained in the documents. Therefore more research needs to be done to investigate how text semantics can be applied to information retrieval, especially in cases where the corpus of documents is big and the queries and documents representations need to be compared fast and without the need of re-indexing. In this work, we conduct an exploratory study to investigate different embeddings and deep learning techniques and how this can be applied to the information retrieval task. We show that although existing methods based on word overlapping perform better in general, in particular cases where the word overlap between queries and documents is low, the use of semantic embedding outperforms other methods based on bag of words.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years we have seen an explosion on the amount of information available on the web and other digital sources. As the size of information increases the access to that information becomes more difficult and that poses new challenges.

One of the challenges to address is how we can make that information available to everyone in a way that is easy to use and do not require domain expertise. This is the main purpose of question answering systems (QA). The QA task consists of, given a question posed in natural language, find the correct answer and present it to the user of the system dismissing the irrelevant information (González, José, 2003). The type of QA will also depend on the input data. For this work, we will focus on question answering over a collection of unstructured text documents.

To access the information correctly the first step is to analyse and understand the question. This raises other subproblems as question classification, ambiguity resolution, identification of temporal relationships, etc. A system that extracts the answer from a document is called shallow, while a system that can infer information from the text is called a deep QA system (Dwivedi & Singh, 2013).

A complete QA system is composed of different submodules. The first step is the information retrieval system where the query is analysed and the system tries to retrieve the most relevant documents. Traditionally, systems keyword-based were used for this approach (Haav & Lubi, 2001). The main problem with these systems is that they lack semantic knowledge and a deeper understanding of the query and context (Deerwester, Dumais, Landauer, Furnas, & Harshman, 1990)(Dumais, Landauer, & Littman,

1996)(Platt, Toutanova, & Yih, 2010). This is partially due to the fact that the same concept can be expressed using different vocabularies and language styles in documents and queries (Shen, He, Gao, Deng, & Mesnil, 2014).

Corpus-based semantic representations (i.e. embeddings) exploits statistical properties of textual structure to embed words in a vectorial space. In this space, terms with similar meanings tend to be located close to each other (Altszyler, Ribeiro, Sigman, & Fernández Slezak, 2017) which alleviates the vocabulary gap between documents and queries.

In this work, we will study if we can improve the information retrieval phase of QA with the use of semantic embeddings. For this purpose, we will compare a keyword baseline system with different approaches based on semantic embeddings. Specially we will focus on cases where the overlap between the words in the queries and the documents is low and therefore a keyword system would perform poorly.

We will apply semantic embeddings and new deep learning techniques to extract the semantics of the document and the query and go beyond the keywords on the text, trying to understand the meaning of the words and the context in which they appear.

Note that the focus of this work is to analyse systems that work well in large scale data. This means that the system should be able to efficiently create a representation for both queries and documents and then compute the similarity between both representations. The documents representations can be computed offline and therefore we can add more complexity when creating these representations. On the other hand, the way we establish similarity between queries and documents needs to be computed fast and therefore the similarity function will need to be simple enough to guarantee that the comparison of the query representation against all the document representations can be achieved in a reasonable amount of time.

The aim of this study is not just to improve existing metrics at all costs, but to keep in mind the feasibility and applications that these systems will have in real life. This study aims to be the first step into more research over different architectures and embeddings. This is why the code has been published and created in a modularised way that can be extended easily and evolve as more research is being done.

This work is organized as follows: In section 2 we will review the state

of the art of QA systems relevant for this work, starting with a presentation of the task and introducing more conventional models and followed by more recent discoveries, making special emphasis on new embeddings and deep learning architectures. Next, based on this literature review, we will describe the experiments. For this we will start by introducing the baseline used, the metrics used to evaluate performance and the dataset used for the experiments. In section 4 we start by evaluating the baseline model and then describe two different experiments. The first experiment will be to make use of existing embeddings and perform different aggregation functions over those embeddings for both the query and the document and the second experiment will be to train a neural network using CNNs. Finally, in section 5 we will draw some conclusions out of the experiments carried out in section 4.

Chapter 2

State of the art

In this chapter, we will review the state of the art for QA systems relevant for this work. First, we will introduce the QA task and the different types of systems and phases. Then we will focus specifically on the information retrieval task and finally, we will describe more recent techniques and deep learning architectures specifically designed for the IR task.

2.1 QA systems

In some context, search engines such as Google have become the common way of searching information (Cairns et al., 2011). The main problem with this type of engines is that people have difficulties distinguishing between reliable and unreliable sources. Unlike search engines, which typically returns a list of documents, the main aim of QA systems is to retrieve the exact answer for a particular question (Tsatsaronis et al., 2012).

A true QA is an information retrieval based system that can understand questions posed in natural language and extract the answer from the text (Loni, 2011).

Although the idea of computer-based QA system has been around since 1970, it did not attract attention until the early 2000 (H. Yu et al., 2007). The development of this field has been widely supported by the Text Retrieval Conference (TREC), which provides the infrastructure necessary for evaluation at scale. The first QA answering system named Baseball was developed in 1961 and was able to answer domain-specific natural language questions about baseball (Loni, 2011). Other systems were developed during this time but were mainly domain-specific systems with limited understanding of the text. It was with the explosion of the information available on the

web that the field has witnessed an increasing interest and more research has been done towards the end goal of having open-domain QA systems.

The annotation of large corpus for QA systems and the creation of curated datasets like WorldTree, OpenBookQA, QED, eQASC, eOBQA, Free917, WebQuestions, WikiMovies, QALD, etc, (Wiegreffe & Marasović, 2021)(Deng & Liu, 2018) has been significant in the development and evaluation of new QA systems. But human-annotated datasets are strictly related to plausibility and not faithfulness (Wiegreffe & Marasović, 2021). This design ignores some of the problems required to answer open-domain questions, like searching for passages that contain the candidate answer, open-domain questions, solving conflicts and aggregating the information from different passages (Dhingra, Mazaitis, & Cohen, 2017). For example, SQUAD questions were made after reading the text, this is called 'back-written questions', which means that lexical overlap between questions and answers is greater than if the questions were made genuinely (Qu et al., 2020).

Among the type of questions that we can find in these annotated corpora there are:

- Factoid questions: "How long is a marathon?"
- List questions: "Name the presidents of the US"
- or other question that requires longer answers and more reasoning across the document: "How do clouds form?"

QA systems were initially focusing on factoid questions but they have been progressively moving towards more complicated questions such as definitions (Tsatsaronis et al., 2012).

2.1.1 Types of systems

There are different types of systems that can be used. In this subsection we will review some of them. Firstly we will make a claisfication based on the approach (linguistics approach vs statistical approach). And later we will introduce another classification based on type of data sources.

Approaches

Different techniques from information retrieval, natural language processing and machine learning can be used to achieve better results on the QA task (Loni, 2011). At the beginning some studies focused on querying structured data while others focused on pattern matching techniques to answer the questions.

There are different approaches that can be used to extract relevant answers for a particular query. One approach is the **linguistic approach** that relies on artificial intelligence and different natural language techniques to build QA systems. One technique to obtain responses to a certain question can be querying a knowledge bases. This particular area is called "question answering over knowledge base" and use rich semantics to deeply understand questions posed in natural language and find answers from knowledge bases (B. Fu et al., 2020). In this approach, the question is parsed into a logical form in order to query the knowledge base. However the knowledge bases are often incomplete and for this reason, the focus is now shifting towards unstructured sources as Wikipedia articles (Dhingra et al., 2017). Other linguistic methods include syntax based pattern extraction for questions and answers (Kolomiyets & Moens, 2011).

Another approach is the **statistical approach** in which the words are extracted as features and a statistical model learns to characterise the relation between question and answer from the training data (Berger, Caruana, Cohn, Freitag, & Mittal, 2000). This approach has higher portability than the linguistic approach as it does not need for specific grammars but instead the system learns from the data. This advantage in term of portability comes with other disadvantages compare to the linguistic approach. One of them being the difficulty of adding expertise knowledge to these systems and the requirements of having a good dataset to train the model. Several techniques can be used for this purpose highlighting SVM, Naive Bayes and Maximum entropy models (Dwivedi & Singh, 2013). Although this method presents some disadvantages it produces better results than other competing approaches.

In this work we will focus on the statistical approach, using large corpus in conjunction with statistical models to learn patterns from the data.

Type of data sources

Apart from the approaches and techniques used in QA there are other classifications based on the type of text data that the QA systems has to deal with. According to (Mervin, 2013) we can have the following types of systems:

- Factoid QA: focuses on questions in which the answers are syntactic or semantic entities such as a person, organization, date, etc.
- Web based QA: aim to retrieve webpages that are potentially relevant to a certain query posed by the user.
- IE based QA systems uses natural language processing techniques to parse questions a documents.
- Restricted domain QA are QA systems that focus on a specific domain and requires linguistic knowledge to support the understanding of the text. This type of QA have better accuracy than open domain QA and less redundancy.
- Rule based QA: extends the information extraction QA systems by generating heuristics alongside with other lexical and semantic features.

All these approaches have some pros and cons and there is not any consensus on which approach is better. The choose is purely dependant on the specific problem and the resources available (Dwivedi & Singh, 2013).

2.1.2 Phases of a QA system

Traditionally, a QA system was composed of three steps (Dwivedi & Singh, 2013) (Loni, 2011).

- The question processing task to analyze the question (parsing, question classification and query reformulation (Dwivedi & Singh, 2013)) and create the query for the IR system. Most of the state-of-the-art question classifiers use machine learning techniques for this purpose (Loni, 2011).
- The passage or document retrieval task to return the most relevant passages that are more likely to contain the answers to the query.
- The last step is the answer processing. The answering processing task aim is to retrieve the words or phrases from the passage that contains the answer to the question.

This landmark has changed significantly during past years and common systems are now composed of just two modules. The first module is an information retrieval module and the second is the reading comprehension module that extracts the answer from the retrieved text (Zhu et al., 2021).

Most of QA systems first use a document retrieval system to identify relevant documents and then in the second phase they perform a deeper analysis to extract the answer from a more fine-grained and reduced set of documents (Monz, 2003). Although these two modules can be developed separately, how the information retrieval module works directly impacts the reading comprehension module. We can have higher accuracy in the information retrieval module by retrieving more documents, but this will make the comprehension task more difficult (Dhingra et al., 2017).

(Mervin, 2013) Current QA systems can now accurately find precise answers from a text passage or an entire document (Kaddari, Mellah, Berrich, Bouchentouf, & Belkasmi, 2020). Unfortunately these systems are not able yet to adapt to specific domains such as the Biomedical domain. A lot of work has been done recently around improving the reading comprehension module, but this one is bounded by the precision of the information retrieval module. In this work, we will focus on improving the information retrieval module.

2.2 Models for Information Retrieval

There are several information retrieval strategies. We will review here some of the more conventional approaches according to (Berger et al., 2000). The first is **adaptive tf-idf** and the idea is to adjusts the weights of the tf-idf matrix to maximise the retrieval of the correct answer. Some other term weighting models include BM25, LM Dirichlet and Divergence from independence (Marwah & Beel, 2020).

BM25 is a bag-of-words retrieval function that ranks documents according to their relevant results using the following formula (Tinega, Mwangi, & Rimiru, 2018):

$$\sum_{t \in q} log(\frac{N}{df_t}) \cdot \frac{(k_1 + 1) tf_{td}}{k_1((1 - b) + b(\frac{L_d}{L_{ave}}) + tf_{td}} \tag{2.1}$$

where

N :represents the document for the collection

df_t :the frequency of a query term in a document

t :is a term of the query q

tf_{td} : represents the frequency of a term in document d

L_d :used to calculate the average document length in the collection

k_1 *and* b :tuning parameters

(2.2)

The most critical language issue for retrieval effectiveness is the term mismatch problem between queries and documents. One of the ways to solve this issue is to apply **automatic query expansion**. This is a simple approach where the user's original query is augmented by new features with a similar meaning (Carpineto & Romano, 2012), this allows expanding the terms in the query to better match with relevant documents.

Following a similar approach based on term expansion and relatedness, **statistical translation models** implicitly expands queries with the use of pre-constructed translation models, which lets you generate query words not in a question by translation to alternate words that are related (Lee, Kim, Song, & Rim, 2008). These models aim to characterise the co-occurrence between words in the queries and words in the documents. They also alleviate the lexical gap issue between queries and documents and have been adapted recently and applied in conjunction with more sophisticated deep learning models (C. Xiong, Dai, Callan, Liu, & Power, 2017).

Another relevant approach is the **latent variable models** that makes the assumption that each question and answer comes from a mix of unseen topics that can be modelled in the latent space (Séaghdha & Korhonen, 2014).

Although these approaches are valid and still widely used, the research community has adapted and created new ways to approach the solution. Current retrieval models can be categorised into two classes (C. Xiong, Callan, & Liu, 2017):

- Representation based models: the goal is to learn good continuous representations of the queries and the documents and then rank the relevance based on the similarity of the representations. The way to establish relevance is by similarity of the two embedded representations

and this can range from the simplistic cosine similarity of two vectors, or more complex approaches that make use of deep neural networks to learn deeper interactions between the two representations. One of the main challenges of the representation based approach is that by nature the query and document are different. The query is typically short and keyword-based while the documents can be large and with diversity of vocabulary (Guo, Fan, Ai, & Croft, 2016) and their representations must reflect this fact.

- Interaction based models match the query and document at the word level. These models include the above mentioned translation models. The main challenge of these models is that the word pairs representations are too sparse to learn.

More conventional models, while effective, still lack semantic knowledge. They also suffer from a phenomenon known as the lexical gap. Although introducing semantic knowledge into an IR system can lead to improvements, the way a query is posed is mainly keyword-based, which explains why some term matching algorithms like BM25 work so well for this set up (Guo et al., 2016). Another point that traditional models lack is the awareness of context. For example, a user asking for "Jaguar SUV prices" might get documents related to the animal "Jaguar" in more traditional term matching algorithms where the context is not taken into account. (Hui, Yates, Berberich, & de Melo, 2017)

New techniques mainly based on deep learning are bringing this gap together by creating vector representations in the semantic space. These vectors are usually word-level embeddings and how to match a question with an answer by their words still remains an open problem. Despite all the interest from the research community in applying deep learning techniques and distributed representations for information retrieval, the success on ad-hoc retrieval tasks has been quite limited (Mitra, Diaz, & Craswell, 2016). Another point of concern is the reproducibility of these experiments. While many papers suggest improvements over the baseline, which is usually BM25, the parameters of the BM25 model are rarely specified. This raises some questions on whether or not a small improvement in performance over the baseline is significant, or if it is just a case of the baseline being set up poorly (Yang, Fang, & Lin, 2018).

2.3 Embeddings - state of the art

Word embeddings are fixed-length vector representations for words that represent meaning via geometry (Almeida & Xexéo, 2019) . A good embedding provides vector representations of words such that the relationship between two vectors mirrors the linguistic relationship between the two words (Schnabel, Labutov, Mimno, & Joachims, 2015).

One of the main advantages of the pre-trained models or embeddings is that they reduce the need for feature engineering, which is a time consuming and usually costly task (Qiu et al., 2020). In traditional informational retrieval, terms have a discrete or local representation whereas distributed representations of the text match the query against the document in the semantic latent space. But these two approaches are not mutually exclusive, a combination of local and distributed representation can improve the performance of an IR system (Mitra et al., 2016).

The first generation of pre-trained models were models aimed to learn good word representations, such as Skip-Gram (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) and Glove (Pennington, Socher, & Manning, 2014). Although these embeddings can capture rich semantics, they are unable to capture the context in which the word or token appears. The other main disadvantage is the out of vocabulary issue, in which words that did not appear in the training phase, will not have a representation embedding. To tackle the out of vocabulary issue other representations have appeared based on character level such as CharCNN (Kim, Jernite, Sontag, & Rush, 2015), FastText (Bojanowski, Grave, Joulin, & Mikolov, 2017) and Byte-Pair Encoding (Sennrich, Haddow, & Birch, 2016) .

The second generation of pre-trained models include models such as BERT (Devlin, Chang, Lee, & Toutanova, 2019) , CoVe (McCann, Bradbury, Xiong, & Socher, 2018), OpenAI GPT (Radford & Narasimhan, 2018) and ELMo (Peters et al., 2018) and are models pre-trained in different downstream tasks that represent words as contextualised embeddings.

Pre-trained models offer several advantages (Qiu et al., 2020):

- Models trained on huge corpus can learn universal representations
- Pre-training models offer better initialisation of the network than random weight and this can quickly speed up the training process and improve the results

- Pre-training can also be seen as a regularization technique in the absence of a big corpus

Some of these pre-trained embeddings have been trained over millions of documents and therefore are rich in semantics. Despite this, it remains unclear how these vectors can be used to train accurate IR systems. In the next section, we will review some of the progress that has been made in the use of word embedding applied to IR.

2.4 Deep learning for information retrieval

For years the BM25 model has been the best information retrieval model, but now we are seeing some other models that make use of the state of the art pre-trained embeddings, like BERT, to feed into a deep learning model and create embeddings to represent the text (Karpukhin et al., 2020). Other works, that follow a term matching approach, make use of non contextualized word embeddings like Word2Vec, stating that the semantics of these embeddings trained on large scale data contain more information than the information that can be learnt by training embeddings from the ground truth data for ad-hoc retrieval (Guo et al., 2016).

The improvements made in (Karpukhin et al., 2020) are a combination of the well-known BM25 and embeddings learned from the text. While these improvements are encouraging there are several precautions that we need to take when we evaluate these models. They require a lot of computational resources for both training and vector similarity comparison, and these resources might not be available in all cases. Also, we do not know if the results would work equally on a different dataset that has not been used during training. For example, in (Mitra et al., 2016) they train a neural network for the ad-hoc retrieval task using the data from the search engine Bing, but the pre-trained embeddings they used were previously trained on a corpus of Bing queries. We do not know if using embeddings pre-trained on a different dataset will work equally well.

Another drawback is that the encoded representation of the paragraph tends to be question dependant and that make these systems more inefficient and difficult to implement for a real use case (W. Xiong, Wang, & Wang, 2021). On the other hand, methods based on tf-idf are insufficient to capture deep semantics beyond lexical overlapping (W. Xiong et al., 2021).

To tackle these issues some authors have opted for a mix of BM25 index model for the first retrieval phase and a second more complex deep learning model that re-ranks the results from the BM25 output (Nogueira & Cho, 2020). Other authors claim that the use of a model based on local representations and a model based on distributed representations can outperform any of the two separately (Mitra et al., 2016). They create two models, one based on an interaction matrix between the terms in the query and the terms in the document, and another deep learning model that aim to generate good representations of the query and document and then learns the relevance between the two in the interaction layers. The final ranking score is then computed as the sum of the scores for both models.

2.4.1 Neural encoders and architectures

To create embeddings from words we need a way to encode the words information. Initially, there are two types of architectures widely used to encode the semantics of the words. The first one is sequential based models. These models capture the context of the word in sequential order. One example of these models are the LSTM or GRU recurrent models or the CNN models. The main problem with LSTM or GRU is that given their recurrent nature they tend to be slow for both training and inference (A. W. Yu et al., 2018). CNN models have proven successful in capturing the context of the words (C. Xiong, Callan, & Liu, 2017)(Hui et al., 2017) and are much faster than their counterparts RNN (A. W. Yu et al., 2018).

Another alternative that has seen a lot of success recently, is the fully connected self-attention model. These models use a fully connected graph to represent the relation between every pair of words and the model learns the structure (Qiu et al., 2020). In common architectures for QA the self-attention layer learns the interactions between context and question (A. W. Yu et al., 2018). While these models have proven powerful they tend to overfit on small or medium-size datasets.

2.4.2 Applications of deep learning

There are different IR tasks in which neural ranking models can be applied (Guo et al., 2019):

- **Ad-hoc retrieval** refers to the IR in which documents remain relatively static

- **Question answering** refers to the answers of question posed in natural language. Question could be close or open domain, and the sources of information could come from a knowledge base (or other source of structured data) or unstructured data as web pages. There have been a variety of task formats for QA:
  - multiple-choice selection
  - answer passage/sentence retrieval
  - answer span locating
  - answer synthesizing from multiple sources

  Altough some of this task are not considered IR tasks (Guo et al., 2019): "*multiple-choice selection is typically formulated as a classification problem while answer span locating is usually studied under the machine reading comprehension topic*". Hence, when referring to QA, it means, particularly for the answer passage/sentence retrieval task.

- **Community QA** aims to answer questions based on previous questions already answered by the community. Questions and answers usually come from web pages such as Quora, Yahoo!, Stack Overflow, etc. In this type of IR models, the question can be answered directly by the pool of answers retrieved by the community or assuming that similar questions can be answered by the same answer.

- **Automatic conversation** systems can be divided into two categories. Retrieval based systems that search for the answers to a query in a large conversational repository and return the best match. Or generative based systems that synthesize new replies (Song et al., 2018).

Deep learning can be applied to many of these tasks. Deep learning has seen great success over recent years and it is attracting more research. These models are getting more popularity than the more traditional models based on bag of words mainly due to the fact that they have better capabilities in terms of understanding the text and they can encode information not just about the context but also about the semantics and syntaxis of the text. Specifically, when applied to areas such as QA systems, it can reduce considerably, or even eliminate, the feature engineering step.

2.4.3 Model Architectures

In this subsection, we will formulate the problem and review the different deep learning architectures commonly used for ad-hoc retrieval. For this

task, the main goal is to learn a ranking function that assigns a higher rank to documents more relevant to the query posed.

The ranking method can be described as a function f of s and t, where s and t are the query representation and document representation respectively. We can formulate f as

$$f(s,t) = g(\psi(s), \phi(t), \eta(s,t))$$

where ψ, ϕ are representation functions that extract features from s and t respectively and η is an interaction function that extracts features from the (s,t) pairs. And the final step consists of a function g that computes the relevance score or rank based on these representations.

As mentioned in previous sections we can differentiate between representation based and interaction-based systems. In interaction-based systems, the interaction function η is a complex function while for representation based systems this function can be a simple function like cosine similarity.

Symmetric vs. Asymmetric Architectures

The representation functions ψ and ϕ define the type of symmetry architecture. If ψ and ϕ are the same, then we have a symmetric network, and if ψ and ϕ are different functions then the architecture is asymmetric.

Symmetric Architecture Symmetric architectures refer to architectures in which question and answer can be exchanged before training without affecting the final output. Among the symmetric architectures, we have siamese networks and symmetric interaction networks. Siamese networks refers to the network in which weights are the same between the question and answer representation, i.e, ψ is equal to ϕ. On the other hand, symmetric interaction networks employ a symmetric interaction function, η, to represent the inputs.

These symmetric architectures are well suited for problems in which the question and answers, s and t, are similar in length and form. This can be the case for phrase retrieval systems but if the document is much longer than the query this type of architecture can perform badly mainly because of the padding needed to fit the query and document in the same initial embedding.

Asymmetric Architecture These architectures fit well on ad-hoc retrieval but can also be used on QA tasks where passages and questions need

to be ranked. In these architectures, if we exchange s and t into the network we will get completely different outputs.

There are mainly three strategies used in this architecture, as depicted in Figure 2.1.

- Query split: based on the assumption that most queries in ad-hoc retrieval are keyword based. In this strategy, we split the query into words terms to match against the document. A typical model is DRMM (deep relevance matching model) (Guo et al., 2016).

- Document split: based on the assumption that a long document could be partially relevant to a query under the scope hypothesis. We can split the document to capture fine-grained interaction signals rather than treat them as a whole. A representative model based on this strategy is HiNT (Romero et al., 2015).

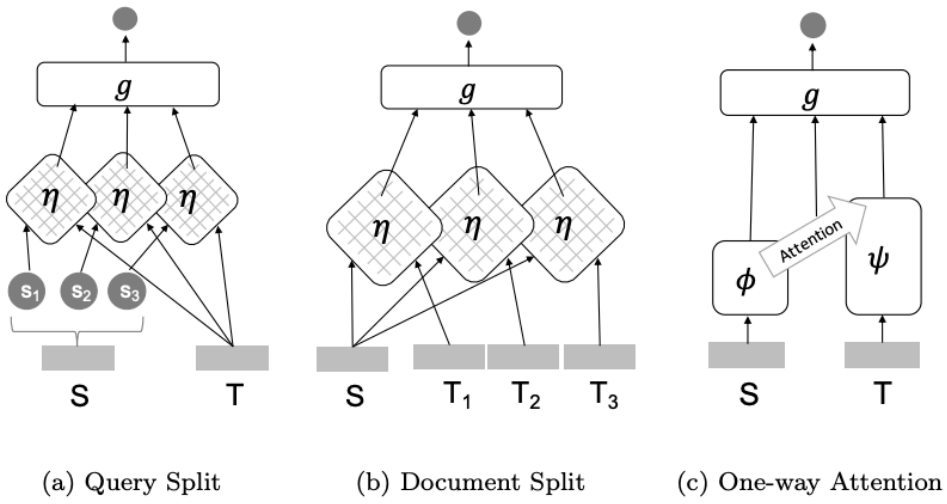
- Join split: Joint split, by its name, uses both assumptions of query split and document split. A typical model based on this strategy is DeepRank (Pang et al., 2017).

DeepRank is a model that uses deep learning techniques to rank for the IR task (Pang et al., 2017). The aim is to generate a score that can be used to rank and order the phrases based on the relatedness to the query. This model consists of three different steps that try to imitate the way the humans will score the responses, this is done by answering the three questions: Where does the relevance occur? How to measure the local relevance? How to aggregate such local relevances to determine the global relevance score?

DeepRank is structured into three steps: a Detection Strategy (according to the query-centric assumption, the relevance usually occurs at the locations where the query terms appear in the documents), a Measure Network (the goal of this step is to determine the local relevance, i.e relevance between query and each query-centric context), and an Aggregation Network (after the measurement step, we obtain a vector $\mathbf{h}$ to represent the local relevance between query and each query-centric context. Therefore, we need a further aggregation step to output a global relevance score).

In addition, in neural ranking models applied for QA there is another popular strategy leading the asymmetric architecture. We name it one-way attention mechanism which typically leverages the question representation to obtain the attention over candidate answer words in order the enhance

the answer representation. For example, IARNN and CompAgg get the attentive answer representation sequence weighted by the question sentence representation.



(a) Query Split (b) Document Split (c) One-way Attention

Figure 2.1: Three types of Assymetric Architecture

Representation-focused vs. Interaction-focused Architectures

We can distinguish another two types of architectures, representation focused (mainly focused on getting two good representation functions ϕ and ψ) or interaction focused (more focused on the interaction function η). Besides these two categories, some neural rankings used a hybrid approach, see Figure 2.2.

Representation focused architectures These architectures focus on getting good deep representations functions ϕ and ψ, but no interaction function η is computed, instead, the interaction g between question and answer is computed as a simple evaluation function as cosine similarity or MLP.

Different deep network structures have been applied for ϕ and ψ, including fully-connected networks, convolutional networks and recurrent networks.

This architecture is also more suitable for tasks with short input texts since it is often difficult to obtain good high-level representations of long

texts (Guo et al., 2019). The main advantage of these models is that they are efficient for online computation. The embeddings of the document are computed offline and although the query needs to be embedded online this comes with a low cost given the short size of the query compared to the size of the documents. This last point is relevant to the problem we are focusing on how to create IR systems that can work efficiently on large scale data.

Interaction focused architectures The assumption of this architecture is that relevance should be represented as the interaction between question and answer. These models focus on the interaction function η instead of the representation functions ϕ and ψ.

The use of an interaction-based model could be better suited for the ad-hoc retrieval task given that the representation based approach will lose the detailed matching signals that are more relevant for the ad-hoc retrieval task than for other NLP tasks (Guo et al., 2016).

Hybrid architectures This architecture uses both, the representations of s and t, and the interaction between them. There are two major hybrid strategies:

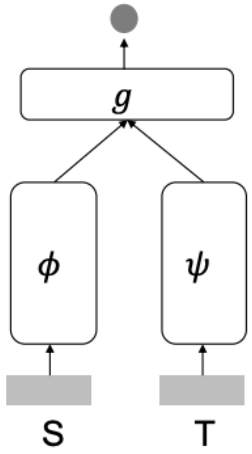
- Combined strategy: generates the representation based and interaction-based sub-models and combines their outputs to generate the relevance score.
- Coupled strategy: integrate the interaction functions, ϕ and ψ, with the interaction function η in the same compact representation.

Single-granularity vs. Multi-granularity Architecture

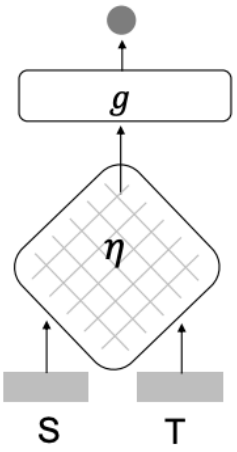
The final relevance score is produced by the evaluation function g. This function combines the different representations to create the ranking score. This can be done at different granularities (word, sentence, phrase, document) as illustrated in figure 2.3.

Single-granularity architecture assumes that the function g can be directly applied to the feature representation ϕ, ψ and η.

Multi-granularity architecture assumes that the function g uses features with different granularities based on different language units. We can identify two types of multi-granularity



(a) Representation-focused



(b) Interaction-focused

Figure 2.2: Representation-focused and Interaction-focused Architectures

- Vertical multi-granularity: takes advantage of the hierarchical nature of deep networks so that the evaluation function g could leverage different level abstraction of features for relevance estimation.
- Horizontal multi-granularity: is based on the assumption that language has intrinsic structures (e.g., phrases or sentences), and we shall consider different types of language units, rather than simple words, as inputs for better relevance estimation.

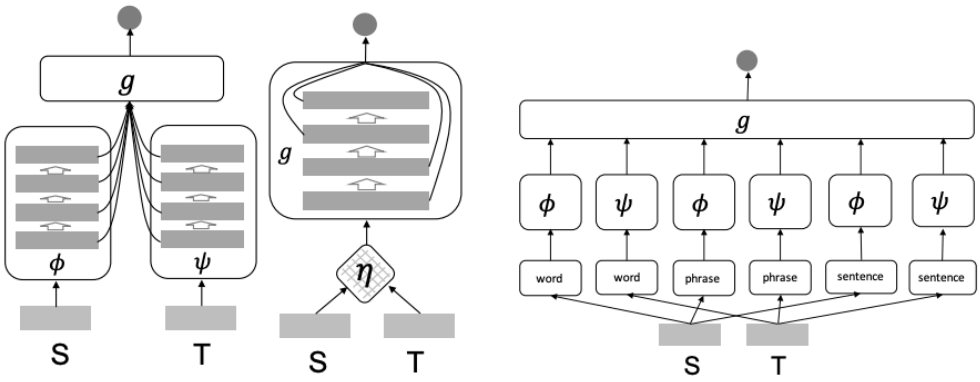
2.4.4 CNN applied to the ad-hoc retrieval task

As introduced in the previous section, there are several architectures that can be used to train a neural network for the information retrieval task. The main purpose of this neural network is to learn a good ranking function

$$f(s,t) = g(\psi(s), \phi(t), \eta(s,t))$$

As we stated at the beginning of this work, we aim to create a system that can work well on large scale data, and for that, we need to focus on representation based architectures. This means that we can describe the ranking as

$$f(s,t) = g(\psi(s), \phi(t))$$



(a) Vertical Multi-granularity (b) Horizontal Multi-granularity

Figure 2.3: Multi-granularity Architectures

The type of neural network chosen to learn good representation functions ($\psi(s)$ and $\phi(t)$), will depend on the use case and requirements. Traditionally RNN works well with text data, given the sequence nature of the text (X. Fu, Liu, Xu, & Cui, 2017). Also, CNNs have proven successful when applied to computer vision (Krizhevsky, Sutskever, & Hinton, 2017), but some recent studies suggest that their applications are not just bound to image processing (Collobert & Weston, 2008). There are several use cases that have proven the success of applying CNNs to text data, and particularly to the information retrieval task (Pang et al., 2017) (Shen et al., 2014). Apart from being powerful, these types of neural networks are faster than other widely used NN like their counterpart RNN (or any of their variations like LSTM or GRU). Here we present some of the most relevant examples that successfully apply CNN's to different language tasks and that will inspire our experiments:

1. (Shen et al., 2014) trains a neural network using CNNs for the retrieval task. They use clickthrough data logged by a commercial search engine. The architecture of the model contains (1) a word-n-gram layer, (2) a letter-trigram layer that transforms each word trigram into a letter-trigram representation vector, (3) a convolutional layer that extracts contextual features for each word (4) a max-pooling layer to form a fixed-length sentence-level feature vector, and (5) a semantic layer that extracts a high-level semantic feature vector for the input word sequence.

After training the neural net, cosine similarity is applied to compute the similarity between the queries and the documents.

2. (Kim, 2014) uses pre-trained word2vectors to train a neural network for sentence classification. The neural network is a 1-layer CNN trained on top of word vectors from an unsupervised language model. In the first experiments, they kept the word vectors static and just learnt the parameters from the network. They state that these results suggest that the word vectors are universal feature extractors. Among the tests done, the most successful is the non-static strategy in which they use one single channel but they update the word embeddings for the specific task during backpropagation.

3. (Reimers & Gurevych, 2019) they use siamese networks to fine-tune BERT to create more meaningful sentence embeddings trained on the Semantic Textual Similarity task. They argue that the BERT embeddings need to be fine-tuned for any downstream task.

CNNs provide certain features like local connectivity, weight sharing, and pooling. They have proven to be effective in mining semantic clues in contextual windows (Young, Hazarika, Poria, & Cambria, 2018). This goes in line with the aim of this work, which is to study the impact of the use of semantic and contextualised embeddings over word overlapping methods. Two of the major drawbacks of CNNs are that they are data-heavy models, and in some instances, it might not be possible to have a big training and test set, and that they struggle to learn long-distanced contextualised information and preserve the sequential order (Young et al., 2018).

2.5 Recap

In this state of the art section, we have provided an overview of the QA systems, the type of systems and their phases. Specifically, we have focused our attention on the information retrieval module.

More semantic capabilities with applications to text data have been introduced in recent years and we have seen some of those applied to the information retrieval tasks.

The existence of open-source embeddings trained over millions of documents and for different tasks have proven effective in adding semantic value when transferred to other downstream tasks different from the ones used for

training. Also, we have introduced some deep learning architectures that aim to learn semantic patterns and create models that go beyond key-word based methods like BM25.

It is important to emphasize that the focus of this work is to study the applications of semantics for information retrieval over large scale data. Working with small scale datasets comes with its own challenges, like small training sets, but also some benefits like the possibility of applying interaction focused architectures that can retrieve the documents fast enough. As we have seen in the literature review, an interaction focused architecture might be more suitable for the information retrieval task, but this approach is incompatible with the creation of systems that can retrieve documents over millions of documents in a reasonable amount of time for the users.

In 2019, 90% of the data produced was produced between 2017 and 2019 alone (Wu, Barker, Kim, & Ross, 2013). And as we expect this trend to increase, more research needs to be done on how the IR systems can evolve and be applied without incrementing the response times exponentially as the number of documents increases. We approach this task knowing that representation focused architectures might not perform as good as interaction focused architectures but that more research has to be done to create systems that can seamlessly cope with the increase of information available.

In the next sections, we will introduce the experiments, the data used for these experiments and the evaluation metrics used to evaluate the experiments. Later we will run the experiments and provide the results and lastly we will provide some conclusions and ideas for future work.

Chapter 3

Experimental setup

As mentioned before, we will make different experiments, one based on word embeddings and another on neural networks. In this chapter, we will make an introduction to these experiments. First, we will describe the dataset that we will use for training and testing, then we will introduce the models that we are going to test and later we will describe the evaluation metrics that we will apply to measure the performance of each individual experiment.

3.1 Dataset

The dataset used for the experiments will be SQUAD2.0 (Rajpurkar, Zhang, Lopyrev, & Liang, 2016). SQUAD is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage . Although this dataset was designed to be evaluated on the machine comprehension task we will adapt it to the ad-hoc retrieval task.

We have chosen this dataset because it provides documents of variable length and semantically restricted paragraphs. As we will see in the following sections the use of BERT embeddings is restricted to paragraphs of a maximum of 512 tokens, this means that the way the SQUAD dataset was created will make straightforward the process of creating the embeddings for every paragraph.

The dataset contains questions with a short answer and a paragraph containing the answer. There is also some unanswerable questions to help improve the machine comprehension model but for this exercise, we will solely

focus on the questions with answer. The dataset is composed of a training set used to train the models and a development set used to test the performance of the models [1].

- The train set contains 86769 questions with the answer for each of those questions contained in one of the 440 documents.
- The development set contains 5926 questions with the answer contained in one of the 35 documents. The development set will serve as the test set for our experiments.

As we stated above, the initial aim of this dataset was to be used to train machine comprehension models. We will need to adapt the dataset to make it suitable for the ad-hoc retrieval task.

This dataset is composed of many paragraphs corresponding to particular Wikipedia documents. Because we want to retrieve the document that contains the answer for a particular question, rather than the single paragraph containing the answer, we have created the documents from the paragraphs of the document.

Given the length of the documents, we will encode each paragraph separately and then we will flatten the encoders at the document level. For example, if we have a document with two paragraphs and the first paragraph contains 40 tokens, the second paragraph 30 tokens and the encoder length is 768, then the vectors representation for that document will have size [70, 768].

This point will be especially relevant while using contextualised embeddings. The basic idea behind these computations is that for each paragraph the encoder will contain information related to that single paragraph but we will feed all the paragraphs, as a single document, into the neural network and with the convolution layers we will add the context and capture signals from other paragraphs.

3.2 Evaluation metrics

To evaluate the performance of the models, we will use the MAP (Cormack & Lynam, 2006) (Mean Average Precision) with different k levels.

[1] For SQUAD, the test set is not publicly available. This dataset is just used to evaluate the models.

Given a query q_i and a set of documents D, the information retrieval system should return a set of possible documents $s_1, s_2, s_3, ..., s_k$ ranked based on the relevance to the correspondent query, where each $s_i \in D$.

We define:

$$rel(d) \begin{cases} 1, & \text{1 if the document d is relevant to the query } q_i \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

$$P@k = \sum_{i=1}^{k} \frac{rel(s_i)}{k} \tag{3.2}$$

here P@k is the precision for the k most relevant documents retrieved. Next we define the total relevant documents:

$$R = \sum_{d_i \in D} rel(d) \tag{3.3}$$

Using equations 3.3 and 3.2 we can define the average precision as:

$$AP = \sum_{k=1}^{|S|} rel(s_k) * \frac{P@k}{R} \tag{3.4}$$

Where $S = s_1, s_2, ..., s_k$ is the set of documents retrieved.

The same average precision is computed for each question separately. To obtain the final score given a set of queries we calculate the average precision of each query individually and then calculate the mean value of the AP for each question. This final score is called MAP (Mean Average Precision) and it is the metric we will use to evaluate the experiments.

Another commonly used metric, especially when there is just one document containing the answer to the question is the Mean Reciprocal Rank (MRR) (Craswell, 2016). We have opted to use the MAP because there are several k values we can adjust for this metric. However, for this particular case, the MAP value when k=1 is the same as the MRR when there is just one relevant document, which is the case for SQUAD.

We will evaluate all the experiments using k values for $k \in 1, 2, 3, 5, 10, 20, 50$. We have selected many values to have a clearer view of model performance. We also want to study in more detail how the model performs with small k-values and that is the reason why we have selected lower granularity on the small k-values.

3.3 Baseline

Since we are focusing on models that work well on large scale data our baseline will be BM25 (see details of the function in section 2.2). We will index the documents in the SQUAD test set using BM25 and then rank the queries. This will be the baseline that we will use to compare performance with the other two experiments.

We will use BM25 at a document level and apply MAP (Mean Average Precision) to evaluate and compare the results with the embedding approach (see section 3.2).

The first exercise will be to compare how an unsupervised model will perform for this particular task. For this, we will choose both parameters, k1 and b, for the BM25 using the default python values (k1=1.5 and b=0.75) and other values already tuned for another IR retrieval task in (Pérez-Iglesias, Joaquín and Garrido, Guillermo and Rodrigo, Álvaro and Araujo, Lourdes and Peñas, Anselmo, 2009) (k1=0.1 and b=0.6). To make a fair comparison between fully unsupervised models we have to assume that there is no access to any ground truth, and therefore these parameters can not be fine-tuned.

SQUAD	map1	map2	map3	map5	map10	map20	map50
BM25 (k1=1.5, b=0.75)	0.8209	0.8521	0.8610	0.8677	0.8715	0.8732	0.8736
BM25 (k1=0.1, b=0.6)	0.7662	0.8072	0.8192	0.8282	0.8338	0.8356	0.8360

Table 3.1: Results of the BM25 applied to the ad-hoc retrieval task

The results shown in the table 3.1 will serve as the baseline when comparing other approaches in the next section.

Chapter 4

Models based on Embeddings Similarity

In this experiment we are going to evaluate different embeddings and different aggregation functions applied to different embeddings. Once the document and query are aggregated we will apply the cosine similarity to establish the relevance between document and query. We have left for future work the study of other similarity metrics.

4.1 Description

We will generate the word embeddings and aggregate the word embeddings of the documents and queries using different aggregation functions (sum, max and min). We will also test the following embeddings:

- fasttext-wiki-news-subwords-300
- word2vec-google-news-300
- glove-wiki-gigaword-50
- glove-wiki-gigaword-300
- glove-twitter-25
- glove-twitter-200
- BERT (dimension: 768)
- Sentence transformers (dimension: 768) (Reimers & Gurevych, 2019)

This selection of embeddings will allow us to draw some conclusions on how the size of the embeddings and the nature of the embedding, i.e the dataset used to train those embeddings, impacts the quality of the retrieval model. For example, comparing two different embeddings with the same nature but different sizes (glove-wiki-gigaword-50 with dimension 50 vs glove-wiki-gigaword-300 with dimension 300) or embeddings that were trained using different data sources (for example glove-twitter-25 trained on social media data or word2vec-google-news-300 trained on google news). Apart from the different embeddings, we will test three different aggregation functions over these embeddings, these will be: minimum, maximum and sum.

All the embeddings are created using the python gensim library, except for BERT that uses the pytorch pretrained BERT model and the sentence transformers created using the sentence_transformers python library. The motivation to use BERT or other models based on BERT, like sentence transformers, is to include contextualised embeddings in our study.

4.2 Results

Here we present the results after using the embeddings introduced above. See tables 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8. Each table contains the results for MAP@k for $k \in [1, 2, 3, 5, 10, 20, 50]$ and using the aggregation functions sum, max and min.

As we can appreciate in the tables the bigger the embeddings are the higher the scores. This could be due to the big size of the documents and the fact that the higher the dimension of the embeddings the more information we can capture.

We also notice that embeddings trained on Wikipedia tend to give better results this could indicate that embeddings trained on a dataset with the same nature of the document give better results.

Across all the experiments there is another common pattern. The aggregation function sum always outperforms the others two (min and max).

Despite the not so bad results, the baseline BM25 still retrieves better results than the best embedding, which is BERT using the sum aggregation.

SQUAD	map1	map2	map3	map5	map10	map20	map50
gensim - sum	0.3920	0.4514	0.4748	0.4960	0.5125	0.5217	0.5245
gensim - max	0.0425	0.0646	0.0741	0.0907	0.1152	0.1347	0.1470
gensim - min	0.0509	0.0696	0.0787	0.0936	0.1099	0.1287	0.1440

Table 4.1: Results using fasttext-wiki-news-subwords-300 with three different aggregation functions

SQUAD	map1	map2	map3	map5	map10	map20	map50
gensim - sum	0.5934	0.6484	0.6697	0.6853	0.6959	0.7001	0.7011
gensim - max	0.2458	0.3035	0.3315	0.3578	0.3788	0.3911	0.3952
gensim - min	0.2529	0.3104	0.3360	0.3604	0.3829	0.3960	0.3997

Table 4.2: Results using word2vec-google-news-300 with three different aggregation functions

SQUAD	map1	map2	map3	map5	map10	map20	map50
gensim - sum	0.5287	0.5992	0.6230	0.6414	0.6532	0.6577	0.6589
gensim - max	0.1460	0.1915	0.2157	0.2414	0.2662	0.2825	0.2883
gensim - min	0.1223	0.1654	0.1895	0.2152	0.2402	0.2564	0.2635

Table 4.3: Results using glove-wiki-gigaword-50 with three different aggregation functions

SQUAD	map1	map2	map3	map5	map10	map20	map50
gensim - sum	0.6304	0.6916	0.7087	0.7217	0.7311	0.7341	0.7351
gensim - max	0.3462	0.4122	0.4395	0.4627	0.4830	0.4918	0.4942
gensim - min	0.2959	0.3597	0.3896	0.4155	0.4376	0.4476	0.4502

Table 4.4: Results using glove-wiki-gigaword-300 with three different aggregation functions

SQUAD	map1	map2	map3	map5	map10	map20	map50
gensim - sum	0.2644	0.3299	0.3573	0.3849	0.4086	0.4192	0.4221
gensim - max	0.0511	0.0726	0.0854	0.1008	0.1207	0.1401	0.1534
gensim - min	0.0226	0.0381	0.0497	0.0655	0.0888	0.1086	0.1222

Table 4.5: Results using glove-twitter-25 with three different aggregation functions

SQUAD	map1	map2	map3	map5	map10	map20	map50
gensim - sum	0.4978	0.5572	0.5797	0.6009	0.6144	0.6201	0.6217
gensim - max	0.0928	0.1281	0.1503	0.1749	0.1993	0.2159	0.2246
gensim - min	0.0426	0.0602	0.0710	0.0857	0.1099	0.1278	0.1412

Table 4.6: Results using glove-twitter-200 with three different aggregation functions

SQUAD	map1	map2	map3	map5	map10	map20	map50
bert - sum	0.6962	0.7510	0.7676	0.7786	0.7857	0.7880	0.7885
bert - max	0.4569	0.5181	0.5386	0.5559	0.5700	0.5769	0.5800
bert - min	0.1888	0.2479	0.2732	0.2983	0.3194	0.3342	0.3394

Table 4.7: Results using BERT contextualised embeddings with three different aggregation functions

SQUAD	map1	map2	map3	map5	map10	map20	map50
ST - sum	0.6297	0.6871	0.7058	0.7194	0.7277	0.7314	0.7324
ST - max	0.5372	0.5985	0.6177	0.6349	0.6472	0.6525	0.6541
ST - min	0.5604	0.6216	0.6426	0.6577	0.6687	0.6736	0.6750

Table 4.8: Results using sentence transformers with three different aggregation functions

4.3 Overlapping Analysis

For the next analyses, we will focus on the embeddings that performed better, which are the BERT embeddings using the sum aggregation. In SQUAD the queries were created after seeing the text, which leads to a higher word overlapping than if the question were made genuinely before seeing the text. We will focus on the results above and perform a more in-depth analysis on how this overlap might be affecting the results.

Given that BM25 is a keyword-based method, i.e it ranks based on the overlapping between document and query, we expect to see low performance on BM25 for queries with a low overlap between the queries and documents. In this particular case of low overlap we expected to see better results using the embeddings approach.

The definition of overlap in this context is quite vague. To define overlap we will first tokenize the query and remove the stop words. With the query list of words, we will see how many of them appear in the text. This is a very simple experiment that aims to check whether the word overlapping between the query and the document is benefiting the results for BM25.

In the following experiment, we will set a maximum overlap threshold and just take the questions where the overlap is less than the particular threshold. For example, if we set the maximum overlap threshold to 9, we will filter out the queries in which the overlap between query and document is more than 9 and then compute MAP@k for the remaining queries. As these results are cumulative, we will see an increase in the MAP@k values as we increase the maximum overlap. We will do the comparisons using MAP@k for every $k \in [1, 5, 20]$. We have choosen this values to be able to draw some conclusions for cases where we require a very precise system (k=1), a more flexible system that allows a small set of documents retrieved (k=5) and a system where we can retrieve many documents (k=20).

We will extract some statistics from the data. For every query overlap we are going to analyse the average length of the queries and also how many queries we have for that particular overlap. If we check the average length for the different query overlaps (see Figure 4.1) we can easily see that there is a tendency of queries with low overlap to be shorter in length.

The fact that the query overlap is almost equal to the average query length suggests that there is a high overlap between words in the queries and

the documents containing the answers. This result reinforces the idea that in some datasets like SQUAD, where the questions were created after seeing the text, the query overlaps will be higher than the type of questions that we expect to see in a real use case.

Also, longer queries have higher probabilities of words appearing in the document, and the focus of this analysis is not to analyse how the length of the query affects the performance of the model but to establish how the word overlap affects each model. This will motivate the use of normalised overlapping to analyse the results independently of the query length.

query_overlaps	mean_query_lenght	number_of_queries
0	1.888889	18
1	2.298969	194
2	3.096367	633
3	3.825871	1206
4	4.657495	1381
5	5.621442	1054
6	6.620791	683
7	7.649867	377
8	8.659898	197
9	9.670103	97
10	10.511628	43
11	11.619048	21
12	12.800000	10
13	13.857143	7
14	14.500000	2
15	15.000000	1
16	16.000000	2

Figure 4.1: Statistics of average query length and number of queries per each query overlaps category.

For the different values of k, we will show the MAP@k values over distinct overlapping thresholds (figures 4.2, 4.4, 4.6) and also a more graphical representation in which we can see the intersection point between the MAP@k values for BERT and the MAP@K values for BM25 (see figures 4.3, 4.5, 4.7). On the left side of the intersection point, i.e for low threshold overlappings, BERT retrieves better results than its counterpart BM25. On the other hand, when we increase the maximum overlap between query and document, we

can see that BM25 improves significantly whereas the results for BERT stay rather flat. Even though the results and numbers vary, we see this pattern for every k.

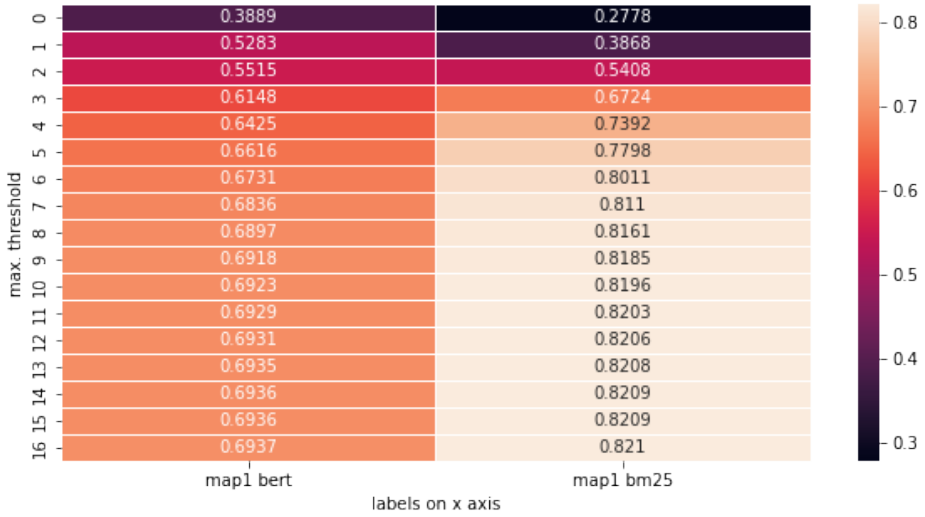


Figure 4.2: Table comparing the MAP1K for different maximum overlaps thresholds between the query and the document.

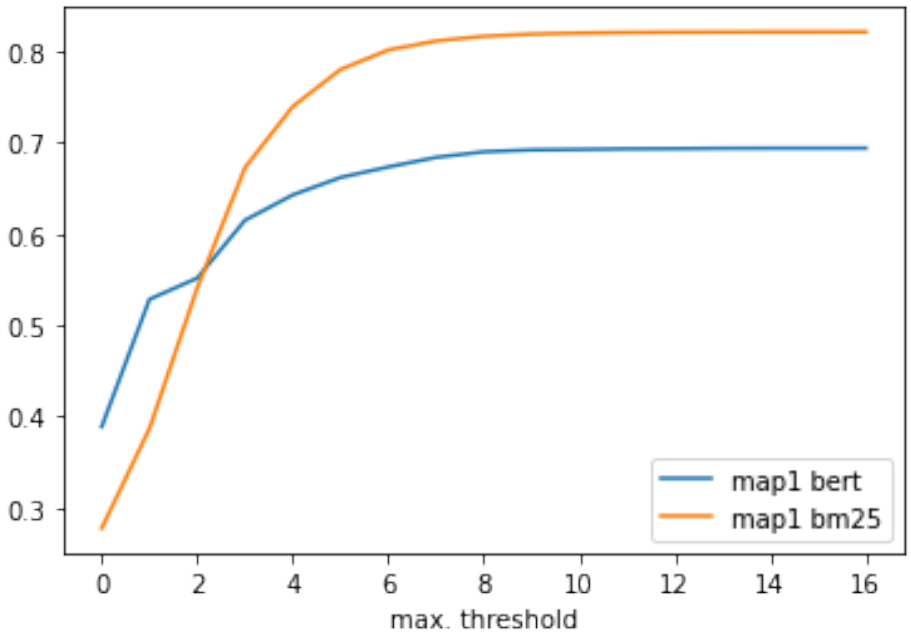


Figure 4.3: Graph comparing the MAP1K for different maximum overlaps thresholds between the query and the document.

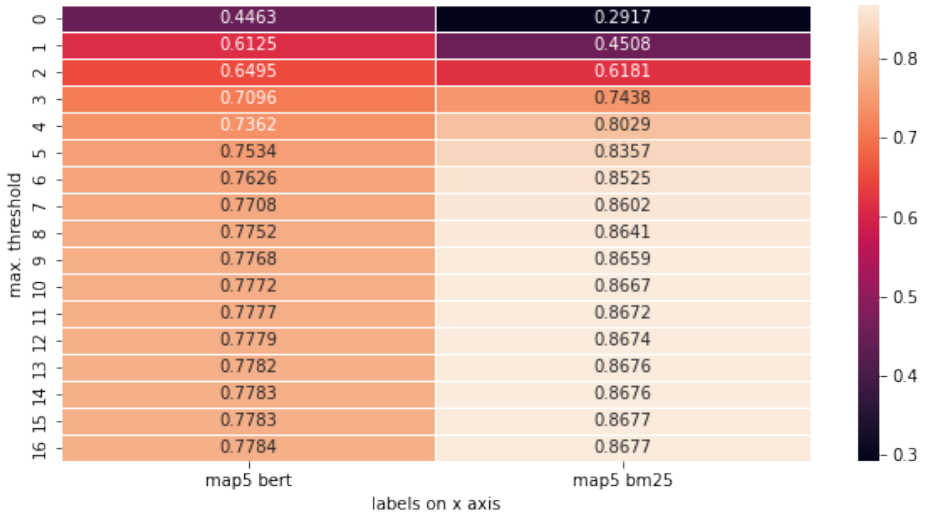


Figure 4.4: Table comparing the MAP5K for different maximum overlaps thresholds between the query and the document.

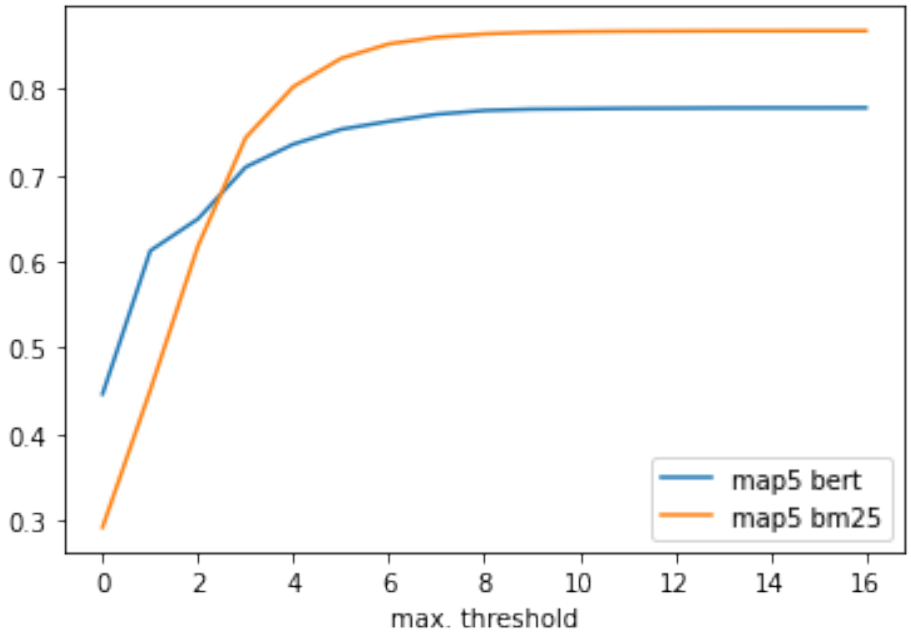


Figure 4.5: Graph comparing the MAP5K for different maximum overlaps thresholds between the query and the document.

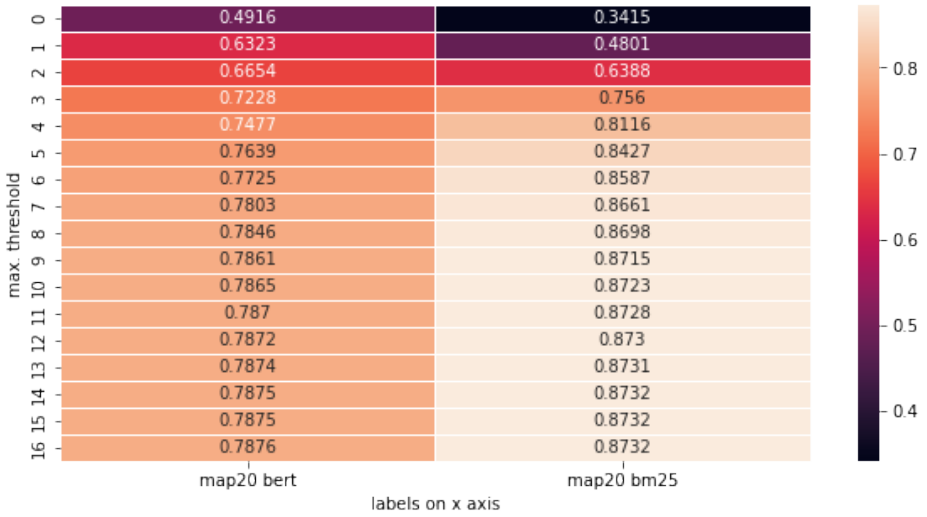


Figure 4.6: Table comparing the MAP20K for different maximum overlaps thresholds between the query and the document.

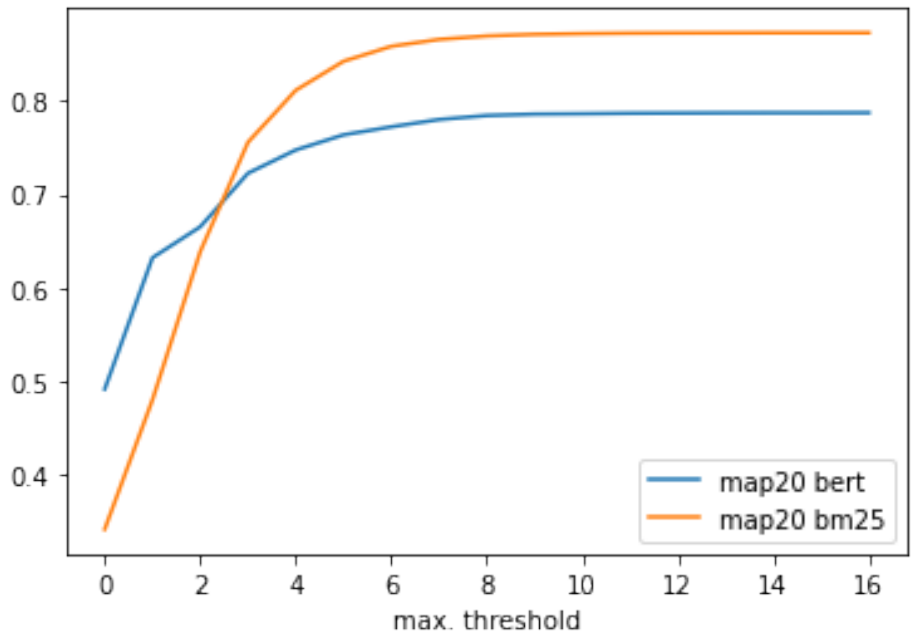


Figure 4.7: Graph comparing the MAP20K for different maximum overlaps thresholds between the query and the document.

Given that the query overlaps highly correlate with the average query length, these results suggest that for short queries, a model that is not based

on word matching, like BERT, might be more suitable. As we mentioned ealier, studying the normalised query overlap might be more relevant for this analysis to extract some conclusions independently of the query length.

4.4 Normalized Overlapping Analysis

We will normalise the overlaps, i.e we will divide the query overlaps by the query length. To make it easier to analyse we will create categories in 10 different splits between 0.0 and 1.0. For example, if the percentage of overlap is 0.6789 we will assign this overlap to the 0.6 split. The analysis will be carried out in the same way. First, we will study the query length and number of queries that lie in each split and then for each maximum normalised overlap we will study the performance of BM25 vs BERT.

query_overlaps_norm	mean_query_lenght	number_of_queries
0.1	1.888889	18
0.3	4.840000	25
0.4	3.894737	57
0.5	5.903846	52
0.6	4.463343	341
0.7	4.796502	629
0.8	5.307806	679
0.9	6.270780	1167
1.0	10.952381	63

Figure 4.8: Statistics of average query length and number of queries per each normalised query overlaps bracket.

As we can see in the table 4.8 the average length of the query is more diverse across the different splits. Also, there higher number of queries are concentrated between the splits 0.7 and 0.9, and in particular the highest number of queries is contained in the split 0.9, this indicates a very big overlap between queries and documents in the SQUAD dataset.

These statistics give us a better overview of how the query overlapping affects the results independently of the query length. Note that in the table 4.8 we do not have the normalised query overlap of 0.2, that is because none of the queries have an overlap between 0.2 and 0.3.

We will analyse these results in the same way we have done for the not normalised query overlaps. We can see the results for every maximum normalised overlap threshold and their corresponding MAP@k values for BM25 and BERT (see figures 4.9, 4.11, 4.13).

For all different k values, we see that BERT performance is better when the overlapping between query and document is less than 0.6. These results suggest that a semantic model like BERT might be more suitable in some instances where the overlap between query and document is low and a model like BM25 underperforms.

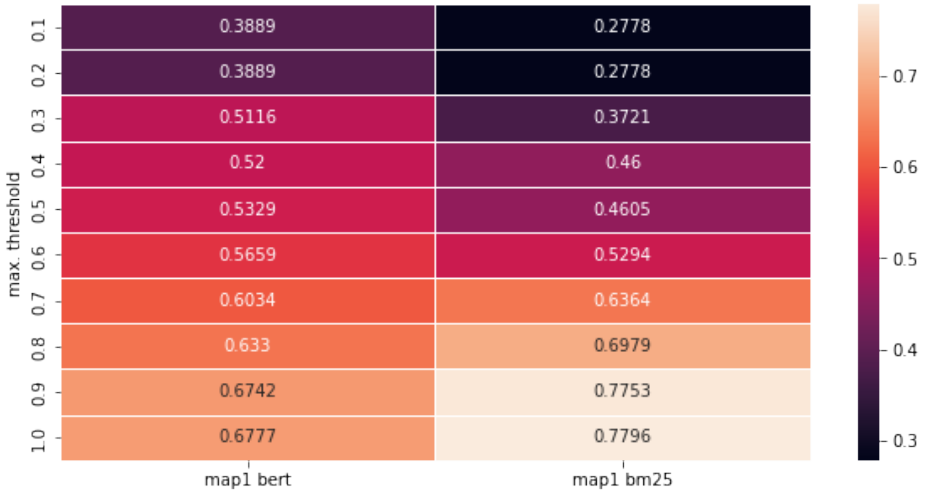


Figure 4.9: Table comparing the MAP1K for different maximum normalised overlaps thresholds between the query and the document.

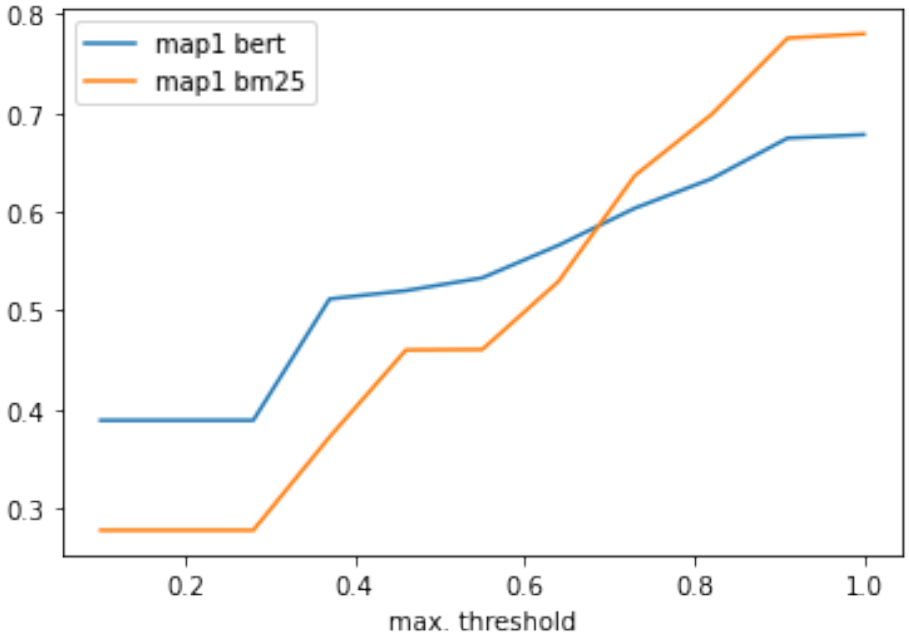


Figure 4.10: Graph comparing the MAP1K for different maximum normalised overlaps thresholds between the query and the document.

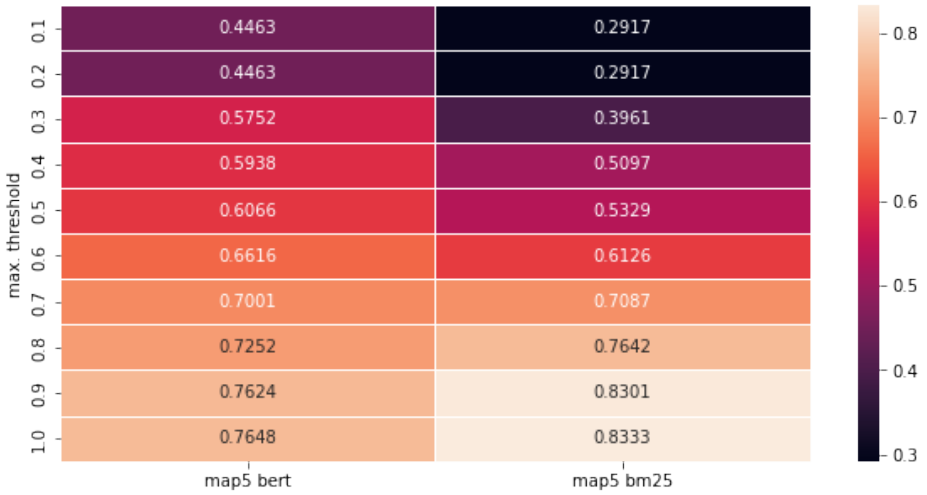


Figure 4.11: Table comparing the MAP5K for different maximum normalised overlaps thresholds between the query and the document.

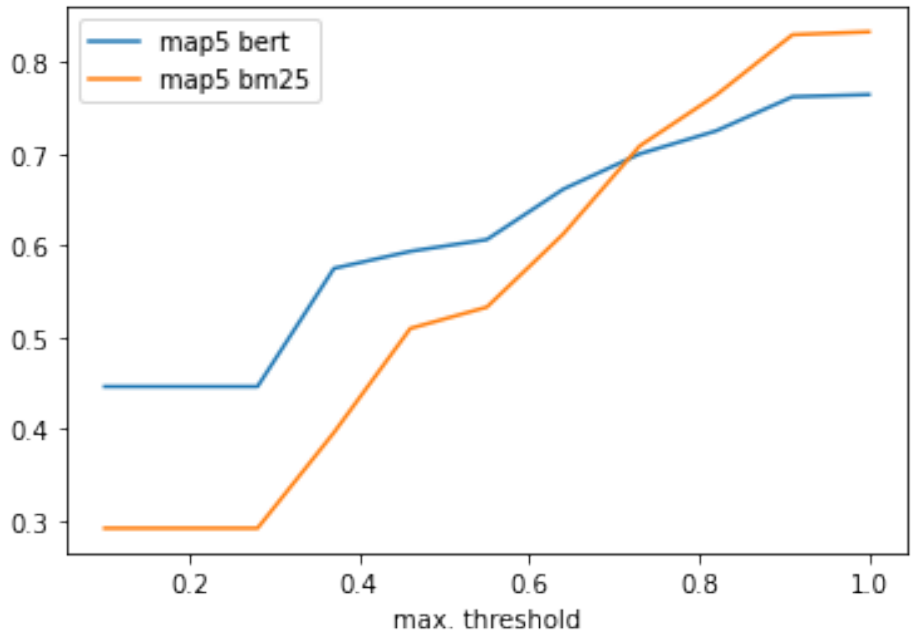


Figure 4.12: Graph comparing the MAP5K for different maximum normalised overlaps thresholds between the query and the document.

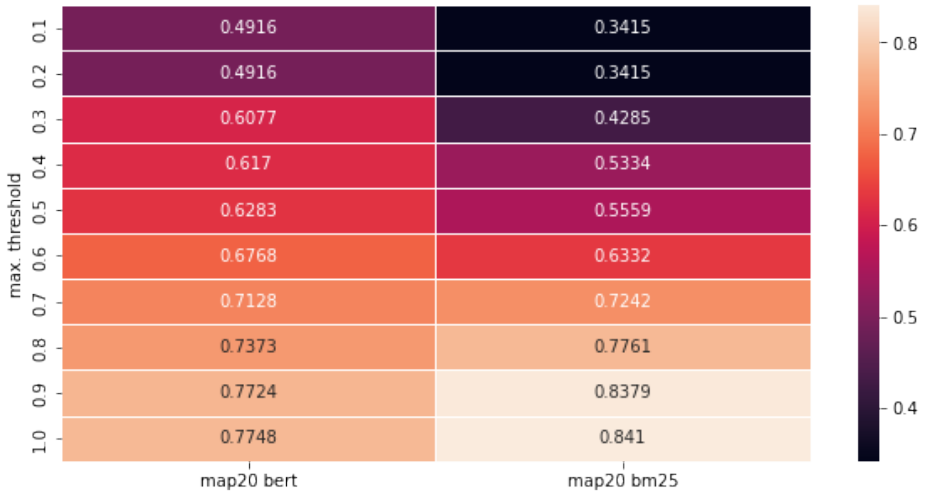


Figure 4.13: Table comparing the MAP20K for different maximum normalised overlaps thresholds between the query and the document.

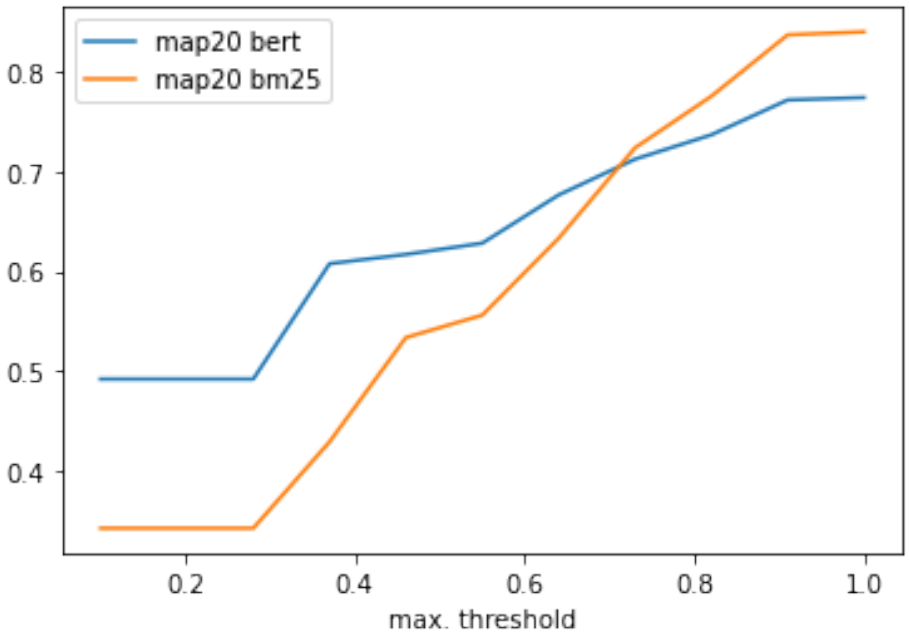


Figure 4.14: Graph comparing the MAP20K for different maximum normalised overlaps thresholds between the query and the document.

Chapter 5

CNN's for ad-hoc retrieval

In the previous section, we have run some experiments over the embeddings and concluded that using BERT embeddings and the aggregation function sum return the best results. Although BERT has not surpassed BM25, we have proved that in cases where there is low overlapping between query and document, a semantic model like BERT is more suitable. We will use these results to construct a neural network aiming to learn more meaningful embeddings than a simple average pooling.

5.1 Description

There are several uses cases that have proven the efficacy of applying CNNs to text data. In the next experiment, we will create a neural network combining different strategies already introduced in section 2.4.4.

Similarly to (Shen et al., 2014) we will follow a representation-focused approach applying CNNs and aiming to learn vector embeddings for both the documents and the queries. To the best of our knowledge, there is not any study that shows how using the BERT embeddings to train a CNN can improve the vector representations for both documents and queries.

An approach similar to (Reimers & Gurevych, 2019) but using an asymmetric architecture instead of a symmetric one can be applied to the ad-hoc retrieval task. Symmetric architectures usually work well when the two inputs of the net have similar sizes and characteristics but in our particular case, the two inputs fed into the net will be the document and the query, and they differ significantly in both size and characteristics. In (Reimers

& Gurevych, 2019) they fine-tune BERT while training on the downstream task but for our case, this will be very costly and inefficient so instead of fine-tunning BERT we will use the vector representations from BERT as the input to our neural network.

Using embeddings as the input to a neural network is an approach similar to the one followed by (Kim, 2014), but instead of using embedding like word2vec, in this experiment, we will use the contextualised embeddings from BERT. The use of embeddings as a starting point for the NN has been widely used for different task along with different types of networks. For example, (X. Fu et al., 2017) uses word2vect embeddings as the input of a recursive autoencoder to improve sentiment analysis. In a similar way, (Wang, Huang, Zhu, & Zhao, 2016) applies Glove embeddings along with aspect embeddings to improve an aspect-level sentiment classification model. Also, (Qin, Xu, & Guo, 2016) trains word2vect embeddings and then feeds those embeddings into a convolutional neural network to improve relation classification.

After applying CNN's to both the question and the document BERT embeddings we need to identify the best way to establish similarity between these two representations. Two common methods are applying a fully connected layer or a cosine similarity function between the representation vectors (Shen et al., 2014). Another approach could be the use of a similarity matrix M to establish the similarity between the question vector representation x_q and the answer text representation (Severyn & Moschitti, 2016):

$$sim(x_q, x_a) = x_q^T M x_a \tag{5.1}$$

This seeks to find a candidate document representation $x_a^{'} = Mx_a$ that is close to the input query x_q. We will test a similar approach in our experiments.

When designing a CNN architecture there are several common techniques that can help in the training phase. Dropout (Kim, 2014) and ReLU (Ghasemi, Fahimeh and Mehridehnavi, Alireza and Pérez-Garrido, Alfonso and Pérez-Sánchez, Horacio, 2018) are good regularisation techniques that can add performance to the model if used with CNN's. In our experiment, we will make use of these regularisation techniques to avoid overfitting.

Also, we can extract non-linear features by using a convolutional layer in conjunction with an activation function (Severyn & Moschitti, 2016). Generally, CNN's architectures, when applied to text data, make use of a single

CNN layer (Shen et al., 2014)(Collobert & Weston, 2008) and a pooling layer, in some cases followed by a fully connected layer (Collobert & Weston, 2008).

Multiple convolutional layers may be stacked by taking the output of one convolutional layer as input to the next layer(Kalchbrenner, Grefenstette, & Blunsom, 2014). Note that in some literature, they refer to the convolutional layer or the convolutional operator as the convolution followed by the activation function (Shen et al., 2014). We will follow the same convention and refer to the convolution as the convolution followed by the activation function.

In this experiment, we will apply a one-dimensional convolution operator. The one-dimensional convolution computes a weighted sum of input channels or features, i.e it is an operation between a vector of weights $m \in \mathtt{R}^m$ and a vector of inputs viewed as a sequence $s \in \mathtt{R}^s$ (Kalchbrenner et al., 2014).

Note that all the experiments carried out up until this point were unsupervised, i.e no labelled data for training were needed. To train the CNN we will make use of both the training set for SQUAD and the development set, which we will refer to as the development set or test set interchangeably.

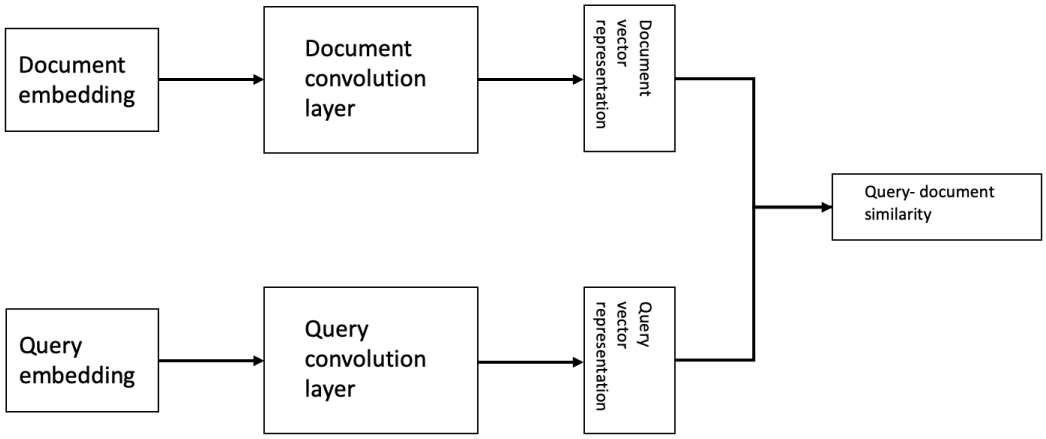


Figure 5.1: General architecture that we will use to train. We will test different convolution layers and similarity functions.

The first layer of our architecture will be the BERT embedding (see doc-

ument and query embeddings on Figure 5.1). We have chosen these embeddings based on the results from the previous experiment. The next step will be to apply a convolution layer. Here we refer to the convolution layer as all the convolution-type of operations and layers. This means that a convolution layer could be a single convolution or it could be many convolution layers stacked. This is where most of the learning during training will happen. From the convolution layer, we will create a vector representation using a pooling strategy. The last step will be to measure the similarity between the document and query embeddings, we will try different approaches which include using cosine similarity, a fully connected layer (Collobert & Weston, 2008) and creating a similarity matrix between the query and document embedding (Severyn & Moschitti, 2016).

As we can appreciate there are several combinations that we could try. Due to the limitations of resources and time, we will restrict the experiments to the most successful ones. We will not provide numbers for all the experiments, but rather guide the experiments towards the most successful combination of parameters. Also, it is important to mention that the aim of these experiments is not to find the best parameters but to narrow down the search and to improve upon those that return the bests results.

Convolution layer:

- Depth: this refers to the number of layers stacked
  - Single layer: this is where we apply a single convolution layer to the input embeddings. In many articles of the literature (Collobert & Weston, 2008), a single layer is applied while working with text data and this technique has proved efficient in our experiments.
  - 2-layers: We have tried to increased the depth of the network by adding another layer. The performance decrease dramatically when increasing the depth, this can be due to the complexity of the problem. Note that due to time and resources limitations we have not been able to test if this type of network will improve if trained for several days or even weeks.
- Width: this refers to the width of the convolution. Notice that if we use more than convolution layer, each layer will have its own width.
  - 50: small width
  - 100: medium width
  - 400: large width

Similarity functions:

- Matrix similarity: on the experiments, we see that the network is not able o learn any patterns when using a similarity matrix
- Fully connected layer: adding a fully connected layer to measure the similarity between both representations gave us better results than using
- Cosine similarity: between the queries and documents vector representations
- Max cosine similarity of the pooling document against every term of the query

Pooling strategies:

- Average pooling: This pooling method mimics one of the aggregation techniques that we applied when we made the previous experiment with the embeddings.
- Max pooling: Max pooling is a pooling strategy used for CNNs when applied to image data. In our case, the experiments have shown a slight decrease in performance when using this pooling strategy compared to average pooling. And therefore for the sake of interpretability, we have restricted the experiments to average pooling.

5.2 Results

Among all these combinations, the best results were obtained using the following parameters:

- On the document branch: using an initial average pool with kernel size 10 followed by convolution with kernel size 400 and average pooling over the resulting vector.
- On the query branch: average pooling over the BERT embeddings
- Cosine similarity function

For this experiment, the MAP@k values for $k \in [1, 2, 3, 5, 10, 20, 50]$ are shown in table 5.1. These results have not improved upon the results of the previous experiment. We have seen that CNNs can add semantic value and

improve performance in the presence of low overlap between queries and documents. We have done the same overlap analysis as in the first experiment and in figures 5.2, 5.4, 5.6 we can see that when the normalised overlap is low, we obtain better results with CNNs than BM25. We have observed that the intersection point between BERT and BM25 happens between the 0.2 and 0.3 normalised overlap for k=1 (see figure 5.3), and between 0.5 and 0.6 for $k \in 5, 20$ (see figures 5.5, 5.7). Although these improvements are not as good as the ones we have seen using the BERT embeddings.

If we compare the BERT embeddings using the sum aggregation with the CNNs, the results for this second experiment are worse for every MAP@K. One of the main problems we have seen when training the CNNs is that they have difficulties learning, this can be due to the size of the document and the issues of CNNs learning long-distance dependencies.

map1	map2	map3	map5	map10	map20	map50
0.6225	0.6866	0.7043	0.7192	0.7279	0.73105	0.7319

Table 5.1: Results after training the CNN for 150 epochs with a convolution applied to the document branch and applying cosine similarity between the document and query vector representations.

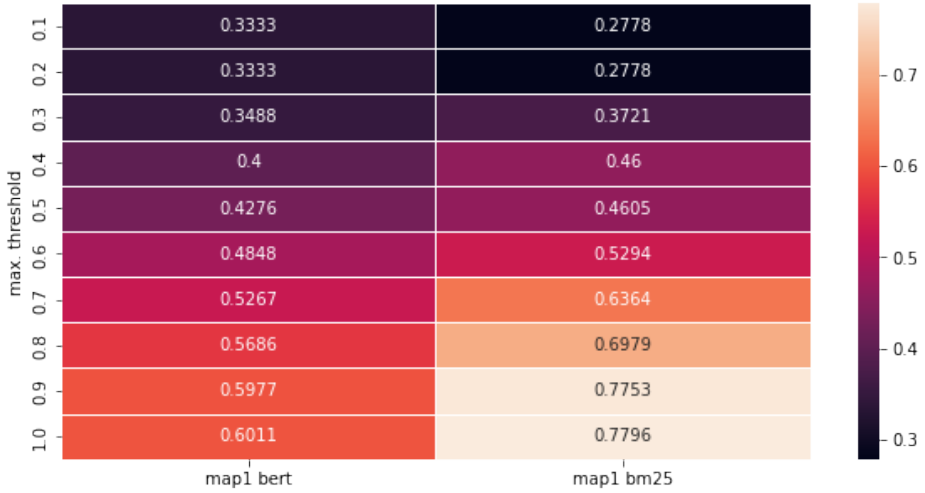


Figure 5.2: Table comparing the MAP1K for different maximum normalised overlaps thresholds between the query and the document.

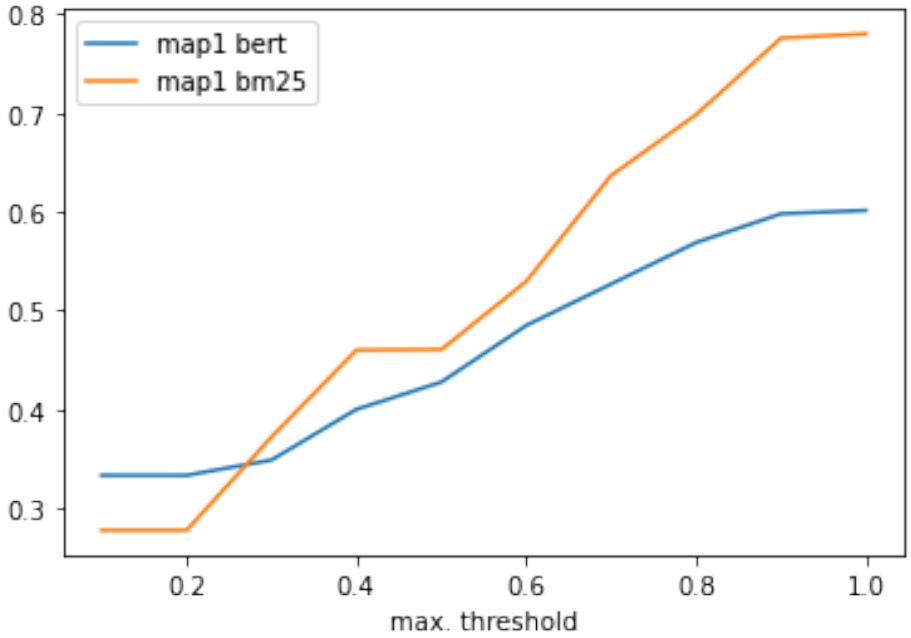


Figure 5.3: Graph comparing the MAP1K for different maximum normalised overlaps thresholds between the query and the document.

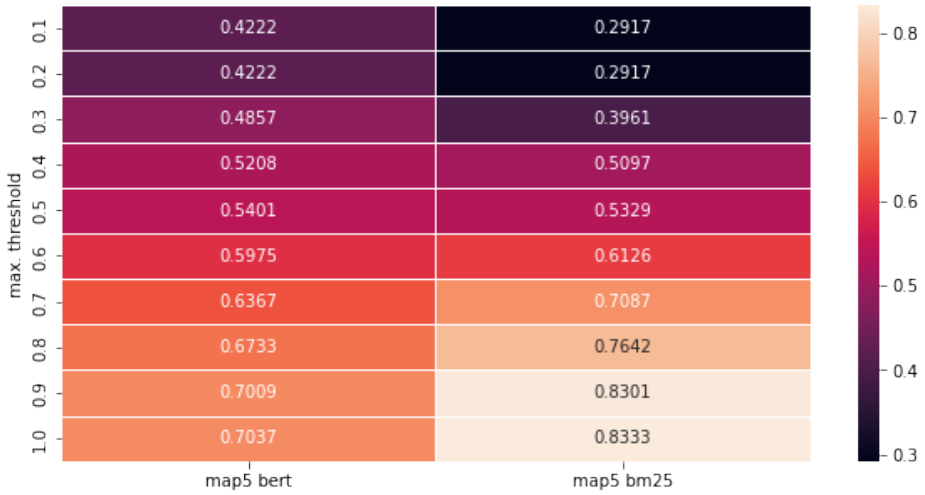


Figure 5.4: Table comparing the MAP5K for different maximum normalised overlaps thresholds between the query and the document.

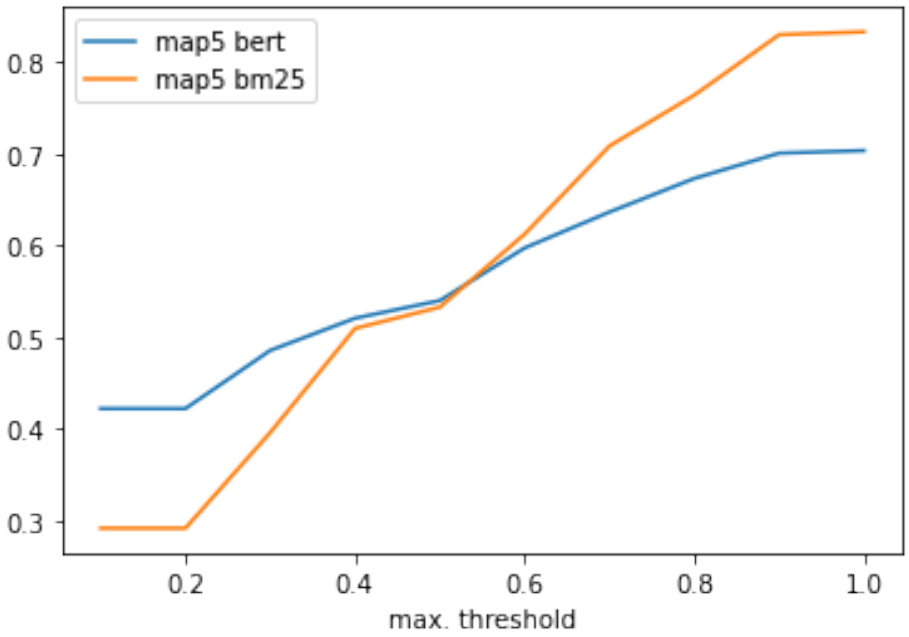


Figure 5.5: Graph comparing the MAP5K for different maximum normalised overlaps thresholds between the query and the document.

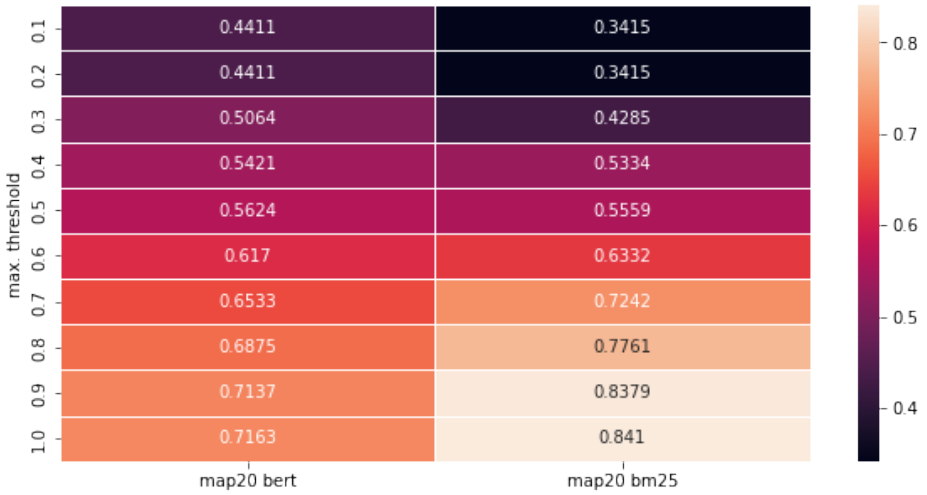


Figure 5.6: Table comparing the MAP20K for different maximum normalised overlaps thresholds between the query and the document.

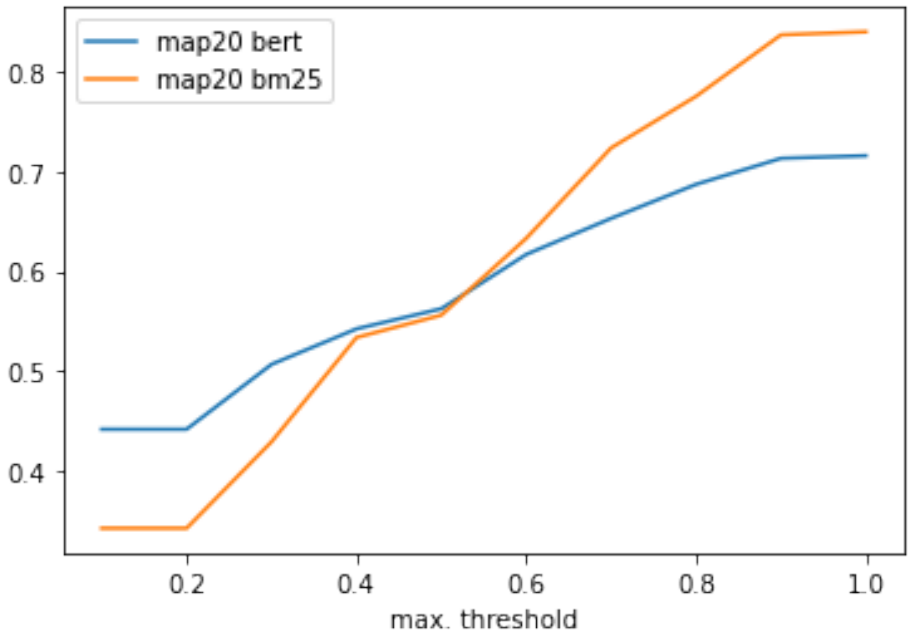


Figure 5.7: Graph comparing the MAP20K for different maximum normalised overlaps thresholds between the query and the document.

Chapter 6

Conclusions and Future Work

In this work, we have tested different approaches for the information retrieval phase of a Question Answering system. In the first experiment, we have tested a non-supervised approach that ranks the similarity between queries and documents based on the similarity of their embeddings. We have tested different pre-trained embeddings and ranked their similarity using the cosine similarity function. The best results were obtained using Bert embeddings and the aggregation function sum. Despite this, the results of this first experiment did not outperform those obtained by the baseline BM25.

In the second experiment, we have evaluated a supervised approach using CNNs and fed the BERT embedding into the network. These results have not shown any improvement compared to the first experiment.

Moreover, we have conducted an analysis to compare the first experiment with the baseline BM25 and test which one performs better in cases where the term overlap between queries and documents is low. In this analysis, we have shown that while BM25 performs better in the presence of high overlap between terms, in cases where the overlap is low, the embedding approach outperforms the baseline. However, one of the main problems is that the datasets available for evaluation usually have a higher overlap than the overlap we expect to see in real systems. In fact, one of the conclusions that this analysis has shown is that for the SQUAD dataset the overlap between queries and documents is big. We think that a new collection with a more realistic overlap should overcome this problem.

We leave for future work the study of other similarity measures that could offer different results.

Bibliography

Almeida, F., & Xexéo, G. (2019). *Word embeddings: A survey.*

Altszyler, E., Ribeiro, S., Sigman, M., & Fernández Slezak, D. (2017, Nov). The interpretation of dream meaning: Resolving ambiguity using latent semantic analysis in a small corpus of text. *Consciousness and Cognition*, *56*, 178–187. Retrieved from http://dx.doi.org/10.1016/j.concog.2017.09.004 doi: 10.1016/j.concog.2017.09.004

Berger, A., Caruana, R., Cohn, D., Freitag, D., & Mittal, V. (2000). Bridging the lexical chasm: Statistical approaches to answer-finding. In *Proceedings of the 23rd annual international acm sigir conference on research and development in information retrieval* (p. 192–199). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/345508.345576 doi: 10.1145/345508.345576

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). *Enriching word vectors with subword information.*

Cairns, B., Nielsen, R., Masanz, J., Martin, J., Palmer, M., Ward, W., & Savova, G. (2011, 01). The mipacq clinical question answering system. *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium*, *2011*, 171-80.

Carpineto, C., & Romano, G. (2012, jan). A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.*, *44*(1). Retrieved from https://doi.org/10.1145/2071389.2071390 doi: 10.1145/2071389.2071390

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: deep neural networks with multitask learning. In *Icml '08*.

Cormack, G. V., & Lynam, T. R. (2006). Statistical precision of information retrieval evaluation. In *Proceedings of the 29th annual international acm sigir conference on research and development in information retrieval* (p. 533–540). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/1148170.1148262 doi: 10.1145/1148170.1148262

Craswell, N. (2016). Mean reciprocal rank. In L. Liu & M. T. Özsu (Eds.), *Encyclopedia of database systems* (pp. 1–1). New York, NY: Springer New York. Retrieved from https://doi.org/10.1007/978-1-4899-7993-3_488-2 doi: 10.1007/978-1-4899-7993-3_488-2

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., & Harshman, R. A. (1990). Indexing by latent semantic analysis. *J. Am. Soc. Inf. Sci.*, *41*, 391-407.

Deng, L., & Liu, Y. (2018). *Deep learning in natural language processing* (1st ed.). Springer Publishing Company, Incorporated.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *Bert: Pre-training of deep bidirectional transformers for language understanding.*

Dhingra, B., Mazaitis, K., & Cohen, W. W. (2017). *Quasar: Datasets for question answering by search and reading.*

Dumais, S., Landauer, T., & Littman, M. (1996, 09). Automatic cross-linguistic information retrieval using latent semantic indexing. *AAAI Tech. Rep*, *1*.

Dwivedi, S., & Singh, V. (2013, 12). Research and reviews in question answering system. *Procedia Technology*, *10*, 417-424. doi: 10.1016/j.protcy.2013.12.378

Fu, B., Qiu, Y., Tang, C., Li, Y., Yu, H., & Sun, J. (2020). *A survey on complex question answering over knowledge base: Recent advances and challenges.*

Fu, X., Liu, W., Xu, Y., & Cui, L. (2017, 02). Combine hownet lexicon to train phrase recursive autoencoder for sentence-level sentiment analysis. *Neurocomputing*, *241*. doi: 10.1016/j.neucom.2017.01.079

Ghasemi, Fahimeh and Mehridehnavi, Alireza and Pérez-Garrido, Alfonso and Pérez-Sánchez, Horacio. (2018, 06). Neural network and deep-learning algorithms used in qsar studies: merits and drawbacks. *Drug Discovery Today*, *23*. doi: 10.1016/j.drudis.2018.06.016

González, José. (2003, 01). La Búsqueda de Respuestas: Estado Actual y Perspectivas de Futuro. *Inteligencia artificial: Revista Iberoamericana de Inteligencia Artificial, ISSN 1137-3601, null 8, NÂº. 22, 2004, pags. 37-56*, *8*. doi: 10.4114/ia.v8i22.805

Guo, J., Fan, Y., Ai, Q., & Croft, W. B. (2016, Oct). A deep relevance matching model for ad-hoc retrieval. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. Retrieved from http://dx.doi.org/10.1145/2983323.2983769 doi: 10.1145/2983323.2983769

Guo, J., Fan, Y., Pang, L., Yang, L., Ai, Q., Zamani, H., ... Cheng, X. (2019). *A deep look into neural ranking models for information retrieval.*

Haav, H.-M., & Lubi, T.-L. (2001). A survey of concept-based information retrieval tools on the web..

Hui, K., Yates, A., Berberich, K., & de Melo, G. (2017). *Co-pacrr: A context-aware neural ir model for ad-hoc retrieval.*

Kaddari, Z., Mellah, Y., Berrich, J., Bouchentouf, T., & Belkasmi, M. G. (2020). Biomedical question answering: A survey of methods and datasets. *2020 Fourth International Conference On Intelligent Computing in Data Sciences (ICDS)*, 1-8.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). *A convolutional neural network for modelling sentences.*

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., ... tau Yih, W. (2020). *Dense passage retrieval for open-domain question answering.*

Kim, Y. (2014). *Convolutional neural networks for sentence classification.*

Kim, Y., Jernite, Y., Sontag, D., & Rush, A. M. (2015). *Character-aware neural language models.*

Kolomiyets, O., & Moens, M.-F. (2011). A survey on question answering technology from an information retrieval perspective. *Information Sciences*, *181*(24), 5412-5434. Retrieved from https://www.sciencedirect.com/science/article/pii/S0020025511003860 doi: https://doi.org/10.1016/j.ins.2011.07.047

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017, may). Imagenet classification with deep convolutional neural networks. *Commun. ACM*, *60*(6), 84–90. Retrieved from https://doi.org/10.1145/3065386 doi: 10.1145/3065386

Lee, J.-T., Kim, S.-B., Song, Y.-I., & Rim, H.-C. (2008, 01). Bridging lexical gaps between queries and questions on large online q&a collections with compact translation models. In (p. 410-418). doi: 10.3115/1613715.1613768

Loni, B. (2011). A survey of state-of-the-art methods on question classification..

Marwah, D., & Beel, J. (2020). Term-recency for tf-idf, bm25 and use term weighting. In *Wosp.*

McCann, B., Bradbury, J., Xiong, C., & Socher, R. (2018). *Learned in translation: Contextualized word vectors.*

Mervin, R. (2013, 10). An overview of question answering system. , *1*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). *Distributed representations of words and phrases and their compositionality.*

Mitra, B., Diaz, F., & Craswell, N. (2016). *Learning to match using local and distributed representations of text for web search.*

Monz, C. (2003). Document retrieval in the context of question answering. In F. Sebastiani (Ed.), *Advances in information retrieval* (pp. 571–579). Berlin, Heidelberg: Springer Berlin Heidelberg.

Nogueira, R., & Cho, K. (2020). *Passage re-ranking with bert.*

Pang, L., Lan, Y., Guo, J., Xu, J., Xu, J., & Cheng, X. (2017, Nov). Deeprank. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. Retrieved from http://dx.doi.org/10.1145/3132847.3132914 doi: 10.1145/3132847.3132914

Pennington, J., Socher, R., & Manning, C. (2014, 01). Glove: Global vectors for word representation. In (Vol. 14, p. 1532-1543). doi: 10.3115/v1/D14-1162

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). *Deep contextualized word representations.*

Platt, J., Toutanova, K., & Yih, W.-t. (2010, October). Translingual document representations from discriminative projections. In *Proceedings of the 2010 conference on empirical methods in natural language processing* (pp. 251–261). Cambridge, MA: Association for Computational Linguistics. Retrieved from https://aclanthology.org/D10-1025

Pérez-Iglesias, Joaquín and Garrido, Guillermo and Rodrigo, Álvaro and Araujo, Lourdes and Peñas, Anselmo. (2009, 09). Information retrieval baselines for the respubliqa task. In (p. 253-256). doi: 10.1007/978-3-642-15754-7_28

Qin, P., Xu, W., & Guo, J. (2016, may). An empirical convolutional neural network approach for semantic relation classification. *Neurocomput.*, *190*(C), 1–9. Retrieved from https://doi.org/10.1016/j.neucom.2015.12.091 doi: 10.1016/j.neucom.2015.12.091

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., & Huang, X. (2020, Sep). Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, *63*(10), 1872–1897. Retrieved from http://dx.doi.org/10.1007/s11431-020-1647-3 doi: 10.1007/s11431-020-1647-3

Qu, C., Yang, L., Chen, C., Qiu, M., Croft, W. B., & Iyyer, M. (2020, Jul). Open-retrieval conversational question answering. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Retrieved from http://dx.doi.org/10.1145/3397271.3401110 doi: 10.1145/3397271.3401110

Radford, A., & Narasimhan, K. (2018). Improving language understanding by generative pre-training..

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). *Squad: 100,000+*

questions for machine comprehension of text.

Reimers, N., & Gurevych, I. (2019). *Sentence-bert: Sentence embeddings using siamese bert-networks.*

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2015). *Fitnets: Hints for thin deep nets.*

Schnabel, T., Labutov, I., Mimno, D., & Joachims, T. (2015, September). Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 298–307). Lisbon, Portugal: Association for Computational Linguistics. Retrieved from https://aclanthology.org/D15-1036 doi: 10.18653/v1/D15-1036

Séaghdha, D. Ó., & Korhonen, A. (2014). Probabilistic distributional semantics with latent variable models. *Computational Linguistics*, *40*, 587-631.

Sennrich, R., Haddow, B., & Birch, A. (2016, August). Neural machine translation of rare words with subword units. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 1715–1725). Berlin, Germany: Association for Computational Linguistics. Retrieved from https://aclanthology.org/P16-1162 doi: 10.18653/v1/P16-1162

Severyn, A., & Moschitti, A. (2016). *Modeling relational information in question-answer pairs with convolutional neural networks.*

Shen, Y., He, X., Gao, J., Deng, L., & Mesnil, G. (2014). A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd acm international conference on conference on information and knowledge management* (p. 101–110). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2661829.2661935 doi: 10.1145/2661829.2661935

Song, Y., Li, C.-T., Nie, J.-Y., Zhang, M., Zhao, D., & Yan, R. (2018, 7). An ensemble of retrieval-based and generation-based human-computer conversation systems. In *Proceedings of the twenty-seventh international joint conference on artificial intelligence, IJCAI-18* (pp. 4382–4388). International Joint Conferences on Artificial Intelligence Organization. Retrieved from https://doi.org/10.24963/ijcai.2018/609 doi: 10.24963/ijcai.2018/609

Tinega, G., Mwangi, P., & Rimiru, R. (2018, 10). Text mining in digital libraries using okapi bm25 model. *International Journal of Computer Applications Technology and Research*, *7*, 398-406. doi: 10.7753/IJCATR0710.1003

Tsatsaronis, G., Schroeder, M., Paliouras, G., Almirantis, Y., Androutsopoulos, I., Gaussier, E., ... Ngonga Ngomo, A.-C. (2012, 01). Bioasq: A

challenge on large-scale biomedical semantic indexing and question answering..

Wang, Y., Huang, M., Zhu, X., & Zhao, L. (2016, November). Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 606–615). Austin, Texas: Association for Computational Linguistics. Retrieved from https://aclanthology.org/D16-1058 doi: 10.18653/v1/D16-1058

Wiegreffe, S., & Marasović, A. (2021). *Teach me to explain: A review of datasets for explainable natural language processing.*

Wu, L., Barker, R. J., Kim, M. A., & Ross, K. A. (2013, jun). Navigating big data with high-throughput, energy-efficient data partitioning. *SIGARCH Comput. Archit. News*, *41*(3), 249–260. Retrieved from https://doi.org/10.1145/2508148.2485944 doi: 10.1145/2508148.2485944

Xiong, C., Callan, J., & Liu, Z. (2017). Convolutional neural networks for so-matching n-grams in ad-hoc search zhuyun dai..

Xiong, C., Dai, Z., Callan, J., Liu, Z., & Power, R. (2017, Aug). End-to-end neural ad-hoc ranking with kernel pooling. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Retrieved from http://dx.doi.org/10.1145/3077136.3080809 doi: 10.1145/3077136.3080809

Xiong, W., Wang, H., & Wang, W. Y. (2021). *Progressively pretrained dense corpus index for open-domain question answering.*

Yang, P., Fang, H., & Lin, J. (2018, oct). Anserini: Reproducible ranking baselines using lucene. *J. Data and Information Quality*, *10*(4). Retrieved from https://doi.org/10.1145/3239571 doi: 10.1145/3239571

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). *Recent trends in deep learning based natural language processing.*

Yu, A. W., Dohan, D., Luong, M.-T., Zhao, R., Chen, K., Norouzi, M., & Le, Q. V. (2018). *Qanet: Combining local convolution with global self-attention for reading comprehension.*

Yu, H., Lee, M., Kaufman, D., Ely, J., Osheroff, J. A., Hripcsak, G., & Cimino, J. (2007). Development, implementation, and a cognitive evaluation of a definitional question answering system for physicians. *Journal of Biomedical Informatics*, *40*(3), 236-251. Retrieved from https://www.sciencedirect.com/science/article/pii/S1532046407000202 doi: https://doi.org/10.1016/j.jbi.2007.03.002

Zhu, F., Lei, W., Wang, C., Zheng, J., Poria, S., & Chua, T.-S. (2021). *Re-*

trieving and reading: A comprehensive survey on open-domain question answering.