Bauhaus-Universität Weimar
Faculty of Media
Degree Programme Computer Science and Media

Deep Neural Ranking Models for Argument Retrieval

Master's Thesis

Saeed Entezari
Born Feb. 2, 1988 in Tehran

Matriculation Number 118048

1. Referee: Prof. Dr. Benno Stein
2. Referee: PD. Dr. Andreas Jakoby

Submission date: September 1, 2020

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, September 1, 2020

. . . .
Saeed Entezari

Abstract

Conversational argument retrieval is the problem of ranking argumentative texts in a collection of focused arguments, ordered based on their quality. This study investigates eight different deep neural ranking models proposed in the literature with respect to their suitability to this task. In order to incorporate the insights from multiple models into an argument ranking, we further investigate a simple linear aggregation strategy.

The Touché @ CLEF shared task on Conversational Argument Retrieval (Task 1) targets a retrieval scenario in a focused argument collection composed of roughly 387740 arguments to support argumentative conversation.

We approach the problem of argument retrieval as finding relevant premises to the input query. Considering the lack of annotated data for the purpose of ranking, we take a distant supervision approach. Based on this approach, the conclusion (claim) of the arguments play the role of queries and the corresponding premises of the argument are the related documents to that queries in an ad-hoc information retrieval scenario. Using a fuzzy similarity measure we have assigned unrelated premises to each conclusion and formed all the information provided by a qrel file in a typical ranking task.

To learn the relation of the queries and the related and unrelated documents we have utilized deep neural ranking models. We first use a Siamese network to generate similarity score. DRMM, KNRM, , and CKNRM are the deep neural ranking models used for retrieving arguments. We use the contextualized embedding known as BERT in three networks including vanilla-BERT, DRMM and KNRM. We also exploited an end-to-end neural ranking model called SNRM which does not require any candidate document in the inference phase.

Finally, we aggregate the document scores for the test queries achieved by different networks. Before the aggregation, we analyzed how diverse the result of the networks are from each other. A linear regression is performed on the validation set to learn how to estimate the final document scores for the test queries.

Test results suggest that focusing on the models whose main concern is to map the interaction between the input pairs would contribute to the better results. The poor test scores of the models highlight the fact that argument retrieval in the sense of retrieving arguments meeting different argument quality dimensions is not a trivial problem.

Contents

Acknowledgements

First and foremost, my special gratitude goes out to Micheal Völske without whose unwavering guidance and dedicated support, this thesis would have been impossible.

I would like to extend my sincere thanks to Prof. Dr. Benno Stein for accepting the supervision of my work.

Last but not least, I am grateful to my parents and my sibling for their devoted support and encouragement along the way.

Thanks for all your encouragement!

Chapter 1

Introduction

Considering that there are different types of opinions toward different topics and typically there is not necessarily one unique correct answer for them, getting an overview of different opinions would be an exhaustive practice and takes considerable amount of time (Wachsmuth et al. [2017b]). In this situation a model which can neutrally retrieve arguments regarding to the controversial topics would be of a high importance. It can provide the user with a reasonable approach toward the questions whose answers may not seem that easy or help them build up a stance for a given topic. Retrieving related arguments for a given topic can be exploited in automated decision making and opinion summarization.

The main concern of this study, as suggested by Bondarenko et al. [2020], is "retrieval in a focused argument collection to support argumentative conversations". The dataset that we work on in this study in order to take part in the Touché shared task competition, is the result of crawling arguments from 4 different debate portals which leave us with 387,740 arguments whose components (claim and premises) are specified.

According to Wachsmuth et al. [2017a], argument quality has different dimensions which can be categorized in logical, rhetorical, and dialectical. It is important to note that getting a mathematical explanation of the quality aspects of the arguments introduced by Wachsmuth et al. is not an easy task and based on our knowledge, it is still an open question. Furthermore, we believe that creating a dataset having the annotation which can cover such quality measures would be expensive as a complex manually human annotation in a large scale is required. Actually these two mentioned problems makes the task of argument retrieval a challenging task.

In this study we have focused on retrieving the related arguments according to the given query by the user. This means that our main focus would be on the logical dimension. To this end, we will use the deep neural ranking model.

In other words, using distant supervision technique, we train the deep neural ranking model for the purpose of ad-hoc retrieval in a conversational argument collection. We hope that among the related arguments that the neural ranking models will retrieve, other quality aspects would also be met.

The dataset provided in this study includes argument component annotations; namely conclusion (claim) and premise. Based on this annotation, we have used a kind of distant supervision technique. We actually hold some assumptions to map the problem of argument retrieval as an ad-hoc retrieval problem. We introduce the problem of argument retrieval as finding related premises to a query. In order to train and validate the exploited models, we take the approach that the premises of the arguments can be considered as the related documents to the queries which are in fact the conclusion of the arguments. Based on this approach the network would provide the user with the arguments whose premises are related to the given query. The premises of the retrieved arguments may support or attack the query (e.g. retrieving the arguments will be done regardless of the stance).

Holding the approach of distant supervision, we have already defined the premises of a conclusion as the related documents to the queries. For the ranking tasks however, the models still require unrelated documents to the query. With the help of fuzzy similarity, we have assigned unrelated premises to each conclusion in the dataset. This way we can construct a dataset including the information of the query relevance known as qrel file which is mandatory for building the ranking models. Forming a dataset with the query relevance information without having click-through or query log data could be considered as the main innovation of this study. This makes training the networks used for ad-hoc retrieval task possible.

By splitting the dataset into training and validation set, we train and validate different deep neural ranking models which learn how to produce similarity scores between queries and documents. For the training set we will provide each query (conclusion) with a related and unrelated document (premise) and in the validation phase, thanks to the validation measures for ranking, we inspect how good the network is in retrieving the related documents.

In this study; we exploited Siamese based networks with recurrent units as sub-networks. We can see how good the performance of a representation-focused network is in retrieving the relevant arguments.

We then inspect the performance of the models which try to grab information from the interaction of the input pairs (interaction-focused networks). DRMM, KNRM and CKNRM produce the ranking score based on different approach toward mapping the interaction between input pairs.

We have also made use of contextualized embedding networks to inspect the effect of state-of-the-art techniques for the text representation in the performance of the ranking. We hope that a deeper understanding of the text pairs, thanks to the contextualized embedding, can lead to better arguments retrieval. Vanilla BERT, KNRM with BERT and DRMM with BERT are the networks with contextualized embedding that have been used in this study.

All the models that are introduced up to now, require candidate documents to retrieve related documents. They do a kind of re-ranking for the test phase. To avoid the problem of error propagation in cascade pipelines, we inspect an end-to-end neural ranking model called stand alone neural ranking model (SNRM).

In order to improve the ranking results and aggregate the assumptions held behind the various exploited networks, we utilized a linear regression technique trained on the normalized validation scores to aggregate the test ranking scores produced by the networks.

In the test phase, we are given 50 test queries and asked to retrieve the best arguments. BM25 scores between the given test queries and conclusions are used to select the candidate arguments. The networks then do the task of re-ranking of the premises of the candidate arguments. Note that SNRM, as an end-to-end neural ranking model, doesn't require any candidate document to do a re-ranking thanks to the inverted index that it produces from the document and query representations.

Based on the test results provided by the competition committee, KNRM has illustrated the best result among the exploited models. For the case of DRMM, using the contextualized embedding improved the results. Finally, the test results suggest that a certain level of performance is achieved when we use contextualized embedding.

The performance of the retrieval models of all competitors of this shared task confirms that the argument retrieval is not a trivial task specially when evaluating the quality of the arguments based on certain measures such as utility and rhetorically well-written criteria come to the discussion. This raises up the need for developing a dataset which can provide the models with the information required for retrieving "good" arguments (arguments which meet different quality dimensions). It also highlights the need for preparing a mathematical modeling or explanation of the argument quality dimension to be included in the cost function of the models.

Chapter 2 provides some background and introduce the concept of arguments. We explain different types of the deep neural ranking models and at the end, introduce the Touché shared task.

Chapter 3 explains about the dataset. We explain in detail about the visualization and the preprocessing steps that are required for each type of network. We explain how form the training and validation sets. We introduce the networks that have been used in this study. We give a detailed explanation about the model structures and the cost functions. We also introduce the metrics that have been used for evaluating the rankings.

Chapter 4 explains the experiments. We elaborate on the details of training the models and show the resulting curves resulted from training and validation of the models. We explain how the trained models are then used to rank the documents given a test query. We finally explain about the analysis of the model results and how diverse the retrieved documents from each other are and based on this analysis explain the linear regression process in order to aggregate the model results.

Finally in **Chapter 5** gives a summary of what we have done in the study, a brief analysis of the results are discussed and future works are suggested.

Chapter 2

Background and Related work

In this chapter, we give a background about the study. We, first, define what we mean by "argument" in the context of this work, clarify why argument retrieval is important, and what criteria can be considered for ranking the arguments. We then take a look at the deep neural network architectures offered by scientists which have been used for the purpose of ranking.

2.1 Argument Retrieval

In this section we introduce what an argument is and why argument retrieval is important. Finally, different contributions toward the problem of argument retrieval will be introduced.

2.1.1 What is an Argument

According to Dumani [2019], arguments may have existed since humans started communicating with each others. People use arguments in order to prove or contradict an opinion. Argument topics are controversial and different people have different opinion toward them. Rieke et al. [1997] defines an argument as an argumentation unit which is composed of a claim and its support. We refer to the supports as premises.

Premises can support or attack a claim. It can be possible to use the premises of one claim to support or attack other claims. A conclusion could be a word, phrase or even a sentence. Typically the premises are texts composed of multiple sentences or paragraphs.

The relation between the argument units can be shown by graphs. This way a better understanding scheme could be achieved for the structure of the arguments. Figure 2.1 shows an example of representing graphs with its unit relations through a graph. In this figure premise 1 and 2 (p1 and p2) are the

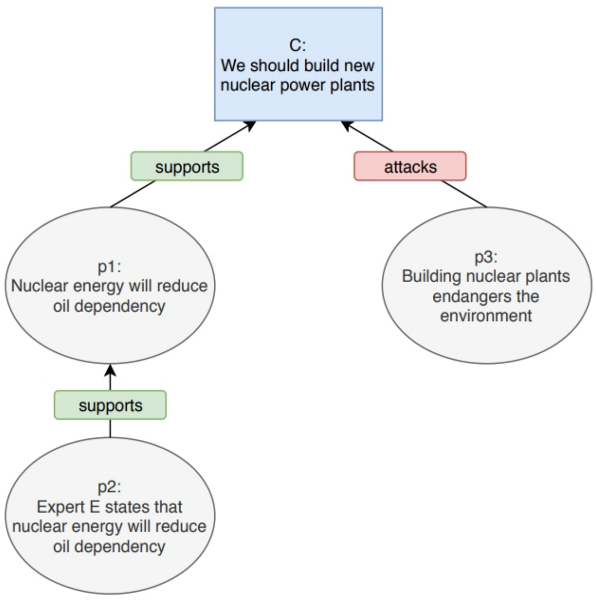

C:
We should build new nuclear power plants
supports
attacks
p1:
Nuclear energy will reduce oil dependency
p3:
Building nuclear plants endangers the environment
supports
p2:
Expert E states that nuclear energy will reduce oil dependency

Figure 2.1: The relation between the argument units (Dumani [2019])

supporting components for the main conclusion while the third premise (p3) attacks it.

2.1.2 Why Argument Ranking

Considering that there are different types of opinions toward different topics and typically there is not necessarily one unique correct answer for them, getting an overview of different opinions would be an exhaustive practice and takes considerable amount of time (Wachsmuth et al. [2017b]). In this situation, a model which can neutrally retrieve the arguments regarding to the controversial topics would be of a high importance. It can provide the user with a reasonable approach toward the questions whose answers may not seem that easy. Retrieving arguments with good quality for a given topic can be exploited in automated decision making and opinion summarization (Wachsmuth et al. [2017b]).

2.1.3 Approaches

Wachsmuth et al. [2017b] introduced one of the first prototypes for a search engine for arguments. The system worked on the crawled arguments from debate websites and ranked them. They exploited a classical ranking model;

namely BM25F, built on top of Luence, in order to rank the arguments for a given query.

ArgumenTex was a system for retrieving topic-related arguments among a large collection of web documents offered by Stab et al. [2018]. The main approach is composed of two stages: firstly the relative documents were retrieved, then the arguments were identified in the retrieved documents and finally, a classification of pro and con was applied to the arguments. For the phase of achieving top-ranked documents they applied Elasticsearch and BM25.

For the evaluation of how convincing an argument is, Habernal and Gurevych [2016] exploited neural networks. After getting the subject judgments regarding to how convincing the arguments are, they feed the arguments into bidirectional LSTM to make a prediction on which argument is more convincing.

Dumani [2019] proposed a two-stage system for retrieving argumentation. He firstly retrieves the related conclusions to a given query. Then the similar claim's premises can be retrieved. He argued that different similarity measures can be exploited in order to retrieve semantically similar claims to the given query such as plain language models with additional smoothing and taking the textual context of the claim into account. For the second phase a search through the clusters of similar premises happens. Figure 2.2 illustrates a diagram of the two stages of the approach.

When ranking the arguments comes to the discussion, the very first question comes up is that, based on which criteria the arguments would be ranked. Different argument quality aspects can be categorized into three main groups: logical, rhetorical, and dialectical dimensions. (Wachsmuth et al. [2017a]).

Logical aspects focus on the *soundness* of the arguments. Wachsmuth et al. argued that the logical arguments have acceptable premises and are relevant to their conclusions.

Rhetorical aspects as defined by Kennedy [2007], is the ability to know how to persuade. According to Blair [2011], these criteria show how successful the argument is in persuading the target audience of a conclusion.

Dialectical arguments are the ones by help of which the user can build his/her own stance for a given topic (Wachsmuth et al. [2017a]). This category of quality may somehow be considered as the 'utility' of the arguments.[1]

In this study we have a special focus on the related arguments to a given query so our main concern would be to retrieve somewhat *logical* arguments.

[1]Wachsmuth et al. [2017a]

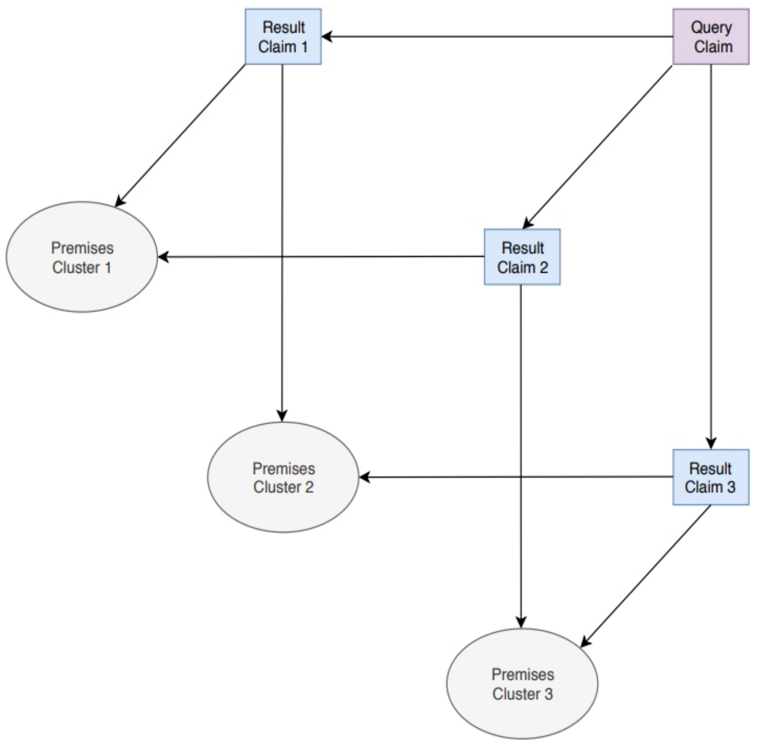


Figure 2.2: A two stage model for retrieving related premises for a given query(Dumani [2019])

2.2 Neural Ranking Models

According to Guo et al. [2019] neural ranking models have an extensive use in ad-hoc retrieval, question answering and automatic conversation. Guo et al. [2019] state that the similarity function for two input texts s and t can be achieved by the equation 2.1.

$$f(s,t) = g(\psi(s), \phi(t), \eta(s,t)) \tag{2.1}$$

in which ψ(s), ϕ(s) and η(s,t) are representation of the texts s, t and the pair of s and t respectively.

2.2.1 Symmetric vs. Asymmetric

These structures are based on homogeneity assumption for documents and queries (Guo et al. [2019]). For the symmetric networks, as the input texts are homogeneous, by exchanging the input texts the output would remain the same. The functions for extracting features in such structures are identical and if we have an interaction function it would also be symmetric. In the auto-conversation and question answering tasks, as the input text pairs are of the same lengths, these networks represent good performance (Guo et al. [2019]). Table 2.1 includes some neural networks based on this assumption.

In the Asymmetric networks if we exchange the text pairs we get different results. They are used for the heterogeneous text pairs where the texts are of the different lengths. Such neural networks would be good choices for the ad-hoc retrieval tasks. (Guo et al. [2019]). Table 2.1 has named some of the networks which been offered with this type of structure.

2.2.2 Representation vs. Interaction focused Networks

Another way of categorizing the neural ranking models is based on their approach toward extracting features of the input texts. Representation-focused neural structures assume that the relevance between the text pairs relies on compositional meaning of the input texts. They typically have a complex function (deep neural network) for the text feature extractors; namely ϕ and ψ. This type of architecture is better for the short input texts (Guo et al. [2019]). Table 2.1 names the networks which hold this assumption for producing ranking score.

Interaction-focused neural structures consider that the relevance relies on the relation between the input pairs. They are divided into non-parametric (ni) and parametric interaction function (pi). Models with parametric interaction function require sufficient dataset. Non-parametric models on the other hand, are efficient online models compared to the parametric interaction function structures. Parametric types better matches the IR tasks rather than representation based models (Guo et al. [2019]). Table 2.1 shows the networks based on interaction focused networks.

Hybrid architectures take the advantage of representation and interaction focused network at the same time. Combined and coupled strategy are two types of the hybrid structures. In the combined model, the two types of architectures are considered as two sub-models. Coupled strategy is a compact hybrid strategy(Guo et al. [2019]). Table 2.1 shows the networks with this approach.

2.2.3 Single vs. Multi-Granularity Networks

Single granularity architectures in which the relevance is computed on the features extracted from a single form of the text inputs. The evaluation function just takes as the input the output of the feature and the interaction function. In multi-granularity architecture the relevance is computed over different granularity of the features, either based on different feature level (vertical multi-granularity) or based on different types of language units (horizontal multi-granularity) of the inputs. The features and interaction function and the input to them become important for the evaluation function. In horizontal multi-granularity, we consider multiple language units (such as word, n-grams, or sentences) as input assuming that the language has intrinsic structure. Table 2.1 contains the name of the networks with this approach for architecture.

2.3 Touché 2020 shared task

As defined by Bondarenko et al. [2020], the task is "retrieval in a focused argument collection to support argumentative conversations". The goal of the task is to prepare a model which can support a user who searches for arguments for debate with appropriate data. For the given queries the model should come up with the good and supportive arguments which is retrieved from arg.me corpus[2]. An argument with a good quality meets different measures dimensions for argument quality. Alongside the topic relevance that a good argument has with the input query, it should be rhetorically well-written. The other quality aspect of a good argument is utility. This means that to what extent the argument can help the user build a stance regarding to a topic.

The retrieved arguments by the models would be annotated manually and the performance of the models are reported by taking the mean of nDCG@5 scores over the 50 test queries. The performance measure for the test phase was unknown to the participants at the time of submission.

In this study, we try to retrieve the arguments which are relative to the given query. Argument retrieval is done regardless of the stance of the arguments. In the machine learning terminology this task is known as a ranking task.

[2]https://webis.de/data/args-me.html

Table 2.1: Neural ranking models and their types offered by different researchers

type	model name	year
Symmetric Architectures	DSSM	Huang et al. [2013]
	DeepMatch	Lu and Li [2013]
	Arc-II	Hu et al. [2014]
	MatchPyramid	Pang et al. [2016]
Asymmetric Architectures	DRMM	Guo et al. [2016]
	KNRM	Xiong et al. [2017]
	HiNT	Fan et al. [2018]
	DeepRank	Pang et al. [2017]
	PACRR	Hui et al. [2017]
Representation-focused Architectures	DSMM	Guo et al. [2016]
	Arc-I	Hu et al. [2014]
	CNTN	Qiu and Huang [2015]
	CLSM	Shen et al. [2014]
	MV-LSTM	Wan et al. [2016a]
Interaction-focused Architectures parametric(pi) and non-parametric(ni)	DRMM (ni)	Guo et al. [2016]
	KNRM (ni)	Xiong et al. [2017]
	Arc-II (pi)	Hu et al. [2014]
	Match-SRNN (pi)	Wan et al. [2016b]
	BERT-based (pi)	Yang et al. [2019]
Hybrid Architectures	DUET	Mitra et al. [2017]
	IARNN	Wang et al. [2016]
	CompAgg	Wang and Jiang [2016]
Multi-granularity Architectures Vertically (v) vs. Horizontally (h)	MutligranCNN (v)	Yin and Schütze [2015]
	MACM (v)	Nie et al. [2018b]
	MP-HCNN (v)	Rao et al. [2019]
	MultiMatch (v)	Nie et al. [2018a]
	Conv-KNRM (h)	Dai et al. [2018]
	MIX (h)	Chen et al. [2018]

Chapter 3

Dataset and Models

In this chapter we introduce the dataset that has been used in the study. A visualization which gives an image of the dataset would be represented. Thereafter, all the models that are exploited in the study will be introduced. Their structures, training process, and loss function terms are discussed.

3.1 Dataset

The dataset has been taken from the corpus of the args.me[1] which is composed of arguments that have been crawled from 4 different debate portals. Debatewise (14,353 arguments), IDebate.org (13,522 arguments), Debatepedia (21,197 arguments), and Debate.org (338,620 arguments) leave us with 387,740 arguments in total. They are in json format and each entry is an argument. For each argument we have a unique ID, conclusion, premise, the source where the argument has been obtained from, the stance of argument whether it supports the topic or contradicts or it just holds a neutral point of view toward the topic, the time of obtaining the argument and other metadata.

In this study we just make a focus on ID, which makes the arguments unique, the conclusion and premises. Note that it is important for us to obtain the relevant arguments, and this should be done regardless of the stance of the premise with respect to the conclusions. Because of that the stance information of the arguments are useless for us.

Note that in the dataset there are arguments whose premises have the length of less than 15 words. We put our assumption on the fact that such arguments cannot represent a valid and convincing argument by which the user can build a stance toward a topic. As a result, we just ignored them and this left us with 317,133 arguments.

[1]https://webis.de/data/args-me.html

3.2 Preprocessing and Visualization

Exploiting the distant supervision technique contributes us in the training and validation phase to have an approach that the conclusions could be considered as the queries which can be given by the users. The premises would play the role of documents in a ranking scenario in which the networks try to rank them. In order to apply the normalization functions simultaneously on the arguments, we have converted the json format to the panda dataframe in which each row represents an argument. For each row we then have the columns named: ID, conclusion, premise. Note that each premise is considered as a related document to its corresponding conclusion (query).

3.2.1 Preprocess Conclusions

In order to get an impression how many premises each unique conclusion may have, we intend to group all the arguments with the same conclusion. This requires a text normalization so that we make sure that we group the arguments correctly. For instance, it is desirable to group the arguments with the claims "Abortion", "abortion!", and "abortion" in the same group as they are discussing about the same topic. We convert all of the claims to have the form of lower case and store them for each argument which will be called "normalized conclusion". Note that the normalized conclusion will be also use to assign unrelated premises to the arguments. It is also exploited for calculating the BM25 score in the test phase when we want to select candidate arguments to be re-ranked. In the next chapter (section 4.3) we will elaborate on the two-step retrieval process.

3.2.1.1 Punctuation Removal and Case Sensitivity

In some cases, we have the same phrases with and without punctuation. In order to avoid to consider them as two separated arguments when we want to analyze them, we decided to simply remove the punctuation. An example of such cases is *"gay marriage should be allowed"* and *"gay marriage should be allowed!!!"*. Also for the cases where the conclusion is just a name, case sensitivity can contribute us take the same conclusions as two separated ones. The example is *"abortion"* and *"Abortion"*.

3.2.1.2 Stop Words Removal and Stemming

Similarity of the conclusions are used in two steps: when we are assigning the unrelated premises to the conclusions and in the test phase when we want to retrieve candidate arguments for a given user query. When we want to evaluate a similarity between two conclusions, stop words can mislead us. They may increase the similarity or decrease it unrealistically. We create a new column for each argument called "norm_con" which is the normalized version of conclusion of the argument achieved by dropping the stop words. For detecting the stop words in the claims we have used nltk[2] package. Stemming methods did not necessarily give us convincing results. In other words, using the existing tools lead to a situation where different forms of the same stem were treated differently and considering the length of the conclusions, it had a destructive effect on the measuring the similarity of the conclusions.

3.2.1.3 Conclusion Visualization and Statistics

With the aim of getting a picture of the conclusions, we have done some analysis on them. Information such as how many premises each conclusion has and how long each conclusion is. To get such information about the data we work on the processed conclusions (normalized conclusions).

Table 3.1 shows the normalized conclusions which have the highest number of premises.

In general there are 66,473 unique conclusions. The histogram of the arguments based on the length of the unique conclusions are shown in figure 3.1. The vocabulary size of the set of unique claims is 29,970 tokens.

3.2.2 Preprocessing Premises

Premises are the documents that we try to retrieve in the training phase and they need a specific preprocess in order to get ready to be fed to the network. Normalization of the premises completely depends on the network that we use as the networks use different types of tokenization.

3.2.2.1 Tokenizing Punctuation

For the classic embedding such in which we assign a static vector to each token, we require to handle the punctuation such as "," as "<COMMA>" or "." as "<PERIOD>". For the contextualized embedding provided by networks BERT, we do not require to tokenize the punctuation as the network will handle them properly. In our study KNRM and CKNRM and SNRM network exploit static tokenization and punctuation tokenization is mandatory. Using a manually developed python dictionary, we could simply replace the punctuation with their corresponding assigned tokens.

[2]https://www.nltk.org/

Table 3.1: Normalized conclusions with the highest number of premises

norm cons	number of premises
abortion	2,401
gay marriage	1,259
rap battle	1,256
god exists	942
death penalty	941
gun control	645
contradict	631
god real	484
god	450
ivf debate	425
abortion illegal	350
existence god	334
abortion legal	332
earth flat	292
euthanasia	290
gay marriage legal	290
death penalty abolished	262
marijuana legalized	256
gun rights	252

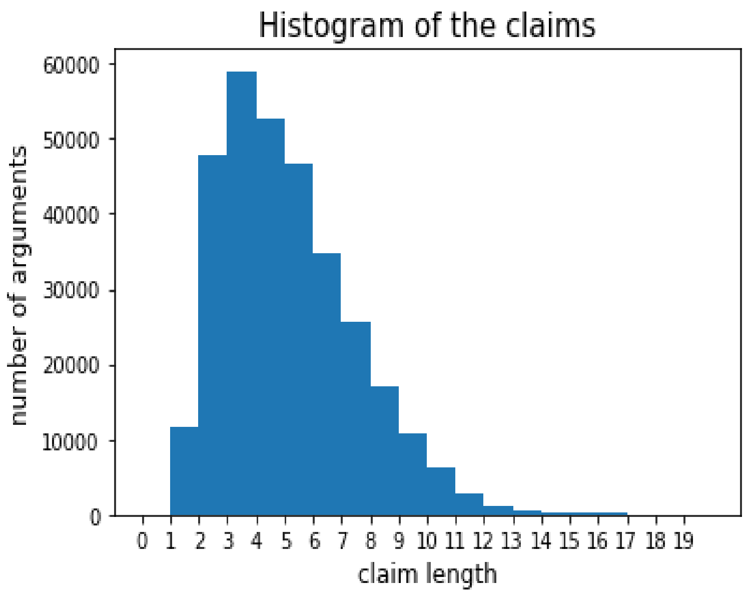


Figure 3.1: Histogram of the unique claims based on the number of tokens

3.2.2.2 Removing the Consecutive Repetitive Tokens

After replacing the punctuation with special tokens, we should focus on consecutive repetitive tokens which could be originally a word or a mapped punctuation. It is sometimes the case that a user in a debate portal has used a special punctuation repetitively in a row or just used the same word for multiple times in order to insist on a fact. In any case this would not bring any semantic information for our network, so we replace them with just one of repeated word.

3.2.2.3 Mapping Digits to Words

During a debate it is pretty possible that a person uses numbers. Sometimes people write the numbers in digit, which could not be informative for the network. It is a common practice to change the digits into word formats. For mapping the digits to the words we have used a python package called "inflect"[3].

[3]source code and installation instruction found in: https://pypi.org/project/inflect/

3.2.2.4 Removing the URLs

A premise may contain URL to refer to a specific source. As the URLs do not contain any white space, they are treated as a single token for the static embedding. They also contain special characters, which make the tokenization complicated without holding any specific semantic meaning. Because of that, we drop all the URLs in the premises. Using regex, we could detect the URL stings in the premises.

3.2.3 Annotating Premises as Related and Unrelated

It is typical for the shared ranking tasks that the queries and the related and unrelated documents are given in a qrel file, and such file is used for training the models. In the ranking tasks such information is provided by query log or click-through data. Considering the type of the annotation that is provided in the dataset in which just the argument components are specified, we have used a distant supervision approach for getting the required information to solve the ranking problem. This means that in order to avoid an expensive procedure of generating human annotated qrel file for the ranking task (assigning relevant score to the documents for each query), we have taken the benefit of the existing annotation information.

This distant supervision happens when we assume the premises as the documents and the conclusions or claims of the arguments as the queries. Note that it is possible that a claim has multiple premises. This means that there are several arguments with the same claim, but the premises are different. In each argument, we put our assumption on the fact that the corresponding premise for each query is a related document to that query.

For the task of ranking we still require to assign unrelated document to each query. To do so, we group the arguments by their conclusion text. Regarding the number of samples, we refer to the number of premises that the specific conclusion text has. For each unique query, we take an argument sample of 20 times of the number of premises. Then we compute the similarity of the conclusion samples and that specific conclusion. In 3.2.5, we will discuss about the similarity metric that we have used for unrelated document assignment. We take the corresponding premises of the most unrelated query samples. The assumption is that the premise of an unrelated claim to a conclusion, can be considered as an unrelated document of that claim. This way we assign for each argument an unrelated premise.

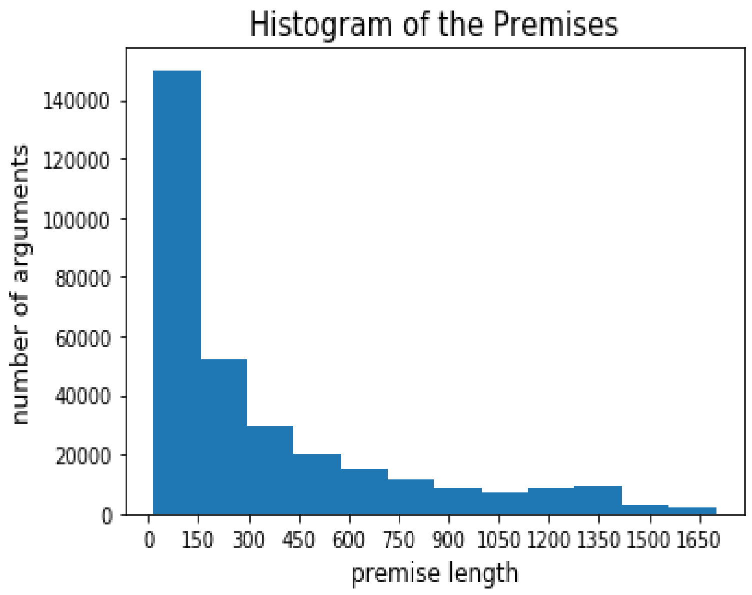


Figure 3.2: Histogram of the premises based on their length (number of tokens separated by white space)

3.2.4 Visualization of Premises

In general, we have 317410 number of premises. The vocabulary size for the set of all premises is 586,796. Figure 3.2 shows the histogram of the arguments based on the premises (Number of tokens separated by white space). Note that we had some arguments whose premises exceed the length of 1,750. As the number of such arguments were not that much, we considered them as outlier to get an appropriate and meaningful histogram scaling. The mean of the number of words for the premises is about 403 words. 85% of the premises have the length of less than 200 words.

Such visualizations for the premises and queries will give us a hint about selecting the maximum sequence lengths in the experiments.

3.2.5 Selection of the Similarity Measure

Assigning unrelated premise to a specific conclusion completely depends on the definition of the similarity measure. For the selection of the similarity measure we took all the arguments whose conclusion include the term *abortion* as a sample. This sample left us with nearly 2000 unique conclusion text which could be claimed that they are relatively related to each other. Then we took a random argument whose conclusion does not have any relation with this topic.

A good similarity measure should give a high score for the two conclusions that are taken from the "abortion" set and relatively give a small score between a conclusion from "abortion" set and "non-abortion" set.

Based on the multiple trials that we had, cosine similarity of the embedded representation (with fast-text and glove)of the conclusions, or the TF-IDF features did not give us a meaningful difference for the similarity values of the two conclusion pairs. A possible explanation for not giving a meaningful result is the sparse vector representation of the conclusions, considering the length of the conclusions. Among the similarity measures of the sentences we have found that the fuzzy similarity provide us with a meaningful similarity score between conclusions, so that we can rely on them for detecting unrelated conclusions. "fuzzywuzzy" provides the tools for string matching and the fuzzy similarity is based on the Levenshtein distance.

3.3 Train and Validation Dataset

Now that we have the required fields for each argument we can start forming the training and validation dataset. For the training dataset we do not have any limitation for the arguments. In the validation dataset however, we have just concentrated on the arguments whose conclusions have exactly 5 related documents. In other words, after grouping the arguments based on the normalized conclusion, we have 5 premises as related premises. For each of the related document we provide 20 unrelated ones. We do so as in the validation phase we provide the network with 5 related and 100 unrelated premises for each query of the validation set. We have assigned more irrelevant documents to each validation query compared to the training query to make the validation phase a bit challenging. Compared to the case in which each validation has 5 positive and 5 negative documents, such number of irrelevant documents makes the validation more realistic. For evaluating the model performance we focus on the first 20 retrieved documents and get the MAP@20, MRR@20, and nDCG@20 score.

In the training and validation phase each query is fed into the network along with the related and the unrelated premise. Based on the visualization of the conclusions and premises we have found that 10 and 100 would be a good choice for the query(conclusion) and document(premise) length respectively. The shorter queries and documents have been padded to these lengths and the longer ones have been truncated to these values.

In order to explore the hyper-parameters such as learning rate and the number of hidden layers, and check how the models work we took a small sample of training (including 10,000 arguments) and validation (with the size of

105 arguments) and ran the model with the sample dataset. We then exploited the the networks to the large scale original dataset.

3.4 Neural Ranking Models

In this section we provide a detailed explanation of the structures of the models that have been used in this study. Note that we have taken the authors' implementation of the corresponding models mostly in PyTorch. The only model implemented in TensorFlow is Stand alone Neural Ranking Model (SNRM). In each subsection, we provide the link of the model implementation.

3.4.1 Recurrent Neural Networks

In this study, we have used Gated Recurrent Units (GRU) in order to have a concentration on the representation-based network. This network creates a representation for the input pairs known as query and documents. The concatenation of the query and the document representations are then fed to a linear layer to produce a similarity score. Figure 3.3 Shows a block diagram of the recurrent network for producing the similarity score for the given query and document. Bidirectional units with the hidden size of 512 have been used for GRU units and the linear layer is a fully connected network with the input of 4*512 to 1 (the concatenation of 2 bidirectional units produces an output with the dimensionality of 4 times of the hidden state). The implementation of the model can be achieved in GitHub.[4]

Gated Recurrent Unit With the aim of capturing the dependencies of the sequence of input these gated units were proposed by Cho et al. [2014]. The flow of the information is modulated by gates. There are in general two types of gates: *updating* and *reset gate*.

Activation gate h_i^j is in fact the linear combination of the previous activation and the candidate activation. Equation 3.1 indicates how the activation gate is updated (Chung et al. [2014]):

$$h_t^j = \left(1 - z_t^j\right) h_{t-1}^j + z_t^j \tilde{h}_t^j \tag{3.1}$$

where the updating gate $\mathcal{Z}_t^j$ decides about the influence of previous activation gate and the candidate activation. The updating gate is computed by the following equation (Chung et al. [2014]):

[4]model implementation of Siamese, KNRM and CKNRM in GitHub: https://github.com/thunlp/Kernel-Based-Neural-Ranking-Models/tree/master/src

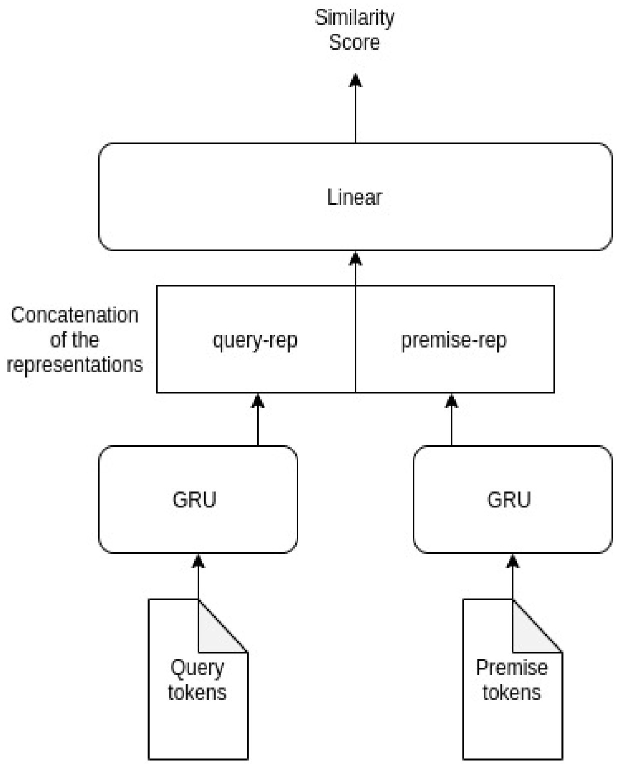


Figure 3.3: Similarity scores using recurrent neural network

$$z_t^j = \sigma\left(W_z \mathbf{x}_t + U_z \mathbf{h}_{t-1}\right)^j \tag{3.2}$$

Candidate activation can be obtained by the equation 3.3 (Chung et al. [2014]):

$$\tilde{h}_t^j = \tanh\left(W \mathbf{x}_t + U\left(\mathbf{r}_t \odot \mathbf{h}_{t-1}\right)\right)^j \tag{3.3}$$

where $\mathbf{r}_t$ is the reset gate and $\odot$ is the element-wise multiplication. Reset gate has also the same equation to the updating gate and can be obtained by the equation 3.4 (Chung et al. [2014]):

$$r_t^j = \sigma\left(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1}\right)^j \tag{3.4}$$

Figure 3.4 shows a GRU with its related gates (Chung et al. [2014]).

Cost function The cost function used for this network is a typical pair-wise learning to rank loss which is represented by the equation 3.5 (Xiong et al. [2017]):

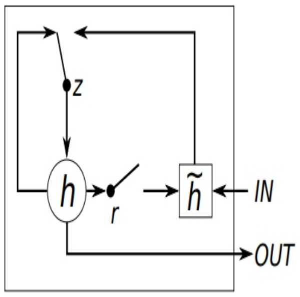


Figure 3.4: GRU with its gates (Chung et al. [2014])

$$l(w, b, \mathcal{V}) = \sum_{q} \sum_{d^{+}, d^{-} \in D_{g}^{+,-}} \max\left(0, 1 - f\left(q, d^{+}\right) + f\left(q, d^{-}\right)\right) \tag{3.5}$$

In this equation $D_q^{+,-}$ is the ground truth for the related and unrelated document sets. w and b are the fully connected layer parameters to be learned and the word embedding is $\mathcal{V}$. Minimizing this cost function would lead the model to put the related documents in a higher rank compared to the unrelated ones.

3.4.2 Deep Relevance Matching Model (DRMM)

Guo et al. [2016] proposed a novel model which concentrates more on the interaction relation that the text pairs have rather than the representation of them. They have addressed the problem of relevance matching by a joint deep structure over the local interaction of the query and the document terms. The local interaction of the text pairs are calculated based on the term embedding. The variable-length interaction terms are transformed to the fixed-length matching histograms. Based on the histograms, a feed forward network is deployed to learn hierarchical matching patterns and produce matching score for each query term. Finally, the similarity score is the result of weighted aggregation of the similarity matching of the query terms. The weights are achieved through a gating network (Guo et al.). The link to the model implementation is given in 3.4.4.

3.4.2.1 Model Architecture

The general structure of the model has been shown in the figure 3.5.

Based on the explanation of the network, the similarity score between the query terms w and the document terms d are calculated by the following equations 3.7 (Guo et al. [2016]):

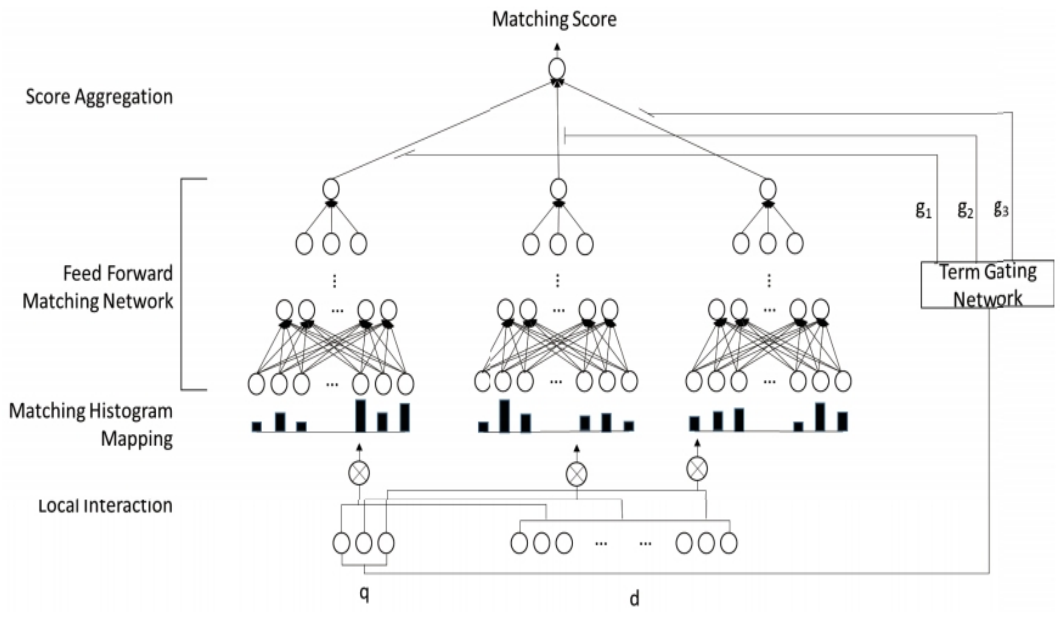


Figure 3.5: DRMM architecture (Guo et al. [2016])

$$
\begin{aligned}
\boldsymbol{z}_i^{(0)} &= h\left(w_i^{(q)} \otimes d\right), && i = 1, \ldots, M \\
\boldsymbol{z}_i^{(l)} &= \tanh\left(\boldsymbol{W}^{(l)} \boldsymbol{z}_i^{(l-1)} + \boldsymbol{b}^{(l)}\right), && i = 1, \ldots, M, l = 1, \ldots, L \\
s &= \textstyle\sum_{i=1}^{M} g_i z_i^{(L)}
\end{aligned}
\tag{3.6}
$$

where $\otimes$ is the intersection operator, $z^{(l)}$ represent the intermediate hidden layers, $\boldsymbol{W}^{(l)}$ is the l-th element of the weight matrix and $\boldsymbol{b}^{(l)}$ is the bias. Finally h is the mapping function from the local interaction and matching histograms. Now, we explain the different parts of the network.

Matching Histogram Mapping Due to the various document and query lengths the local interactions have different lengths. Rather than location preserving representation through matching matrix Guo et al. proposed a strength preserving representation for the local interaction relation. In this form of representation the local interactions (i.e. cosine similarity) are grouped according to the discrete levels of signal strength bins. They used five bins in their study. Thanks to the matching histogram we come up with a fixed-sized representations for the local interactions. Note that in our experiment due to the simplicity of learning of the multiplicative relationships and reducing the range, we have exploited the LogCount-based Histogram. This means that the logarithm of the counts for each bin has been calculated (Guo et al.).

Feed Forward Matching Network In this study, we have used a three layer feed forward neural network for learning the term similarity scores. For the input layer we have used 1024 units and for the hidden layers we have used 256 and 5 units. Finally the output layer has a neuron which clarifies the matching score of the query term.

Term Gating Network Query term importance has been modeled in this model by the term gating network. The gate of each term will clarify to what extend the term will have contribution for calculating the similarity relevance. The gating functions are softmax function which is shown in equation 3.7:

$$g_i = \frac{\exp\left(\boldsymbol{w}_g \boldsymbol{x}_i^{(q)}\right)}{\sum_{j=1}^{M} \exp\left(\boldsymbol{w}_g \boldsymbol{x}_j^{(q)}\right)}, \quad i = 1, \ldots, M \tag{3.7}$$

where $\boldsymbol{w}_g$ denotes the weight vector of the term gating network and $\boldsymbol{x}_i^{(q)}$ is the i-th query term.

Cost Function The cost function for the learning phase is the same as the one that we use for the recurrent network. This is typical for training the neural networks for the task of ranking.

3.4.3 Neural Ranking Models with Kernel Pulling

In this type of neural ranking models kernel pulling is applied to a function of interaction between documents and queries. The output of such kernels would be an input to a ranking model which typically is a linear layer. For the kernels we have used radial basis functions (RBF).

3.4.3.1 Kernel-Based Neural Ranking Model (KNRM)

In the Kernel based Neural Ranking Models (KNRM), we aim to produce a similarity score for a given query and document. This model composes of three important parts: translation model, kernel pooling, and learning to rank model (Xiong et al. [2017]). Figure 3.6 shows the layers and components of the KNRM model.

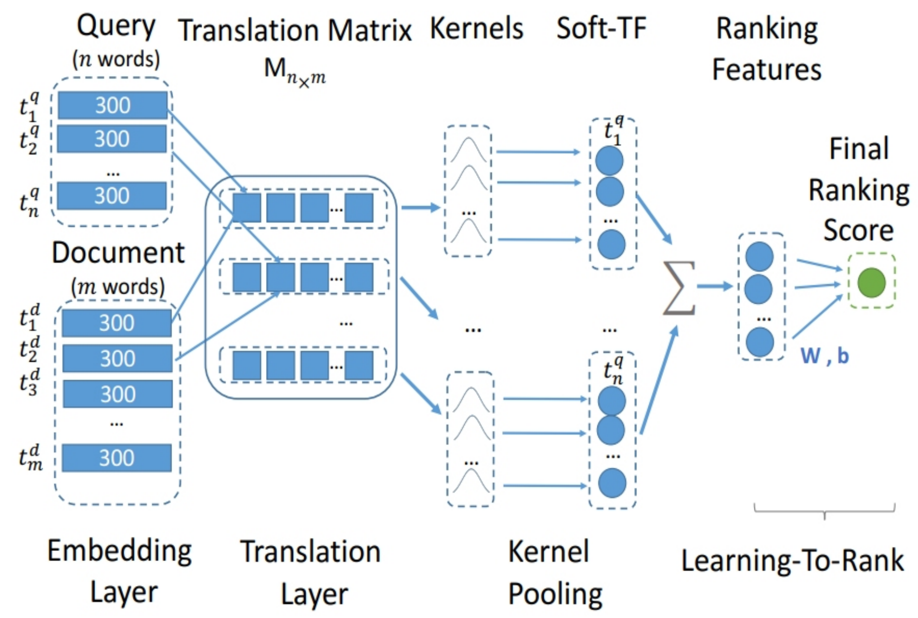


Figure 3.6: Different layers of the KNRM structure (Xiong et al. [2017])

Translation model The translation matrix is used for word-level similarity. Each element of the matrix is the cosine similarity of the embedding vector for the query and document terms. The element M_{ij} of the translation matrix can be calculated by the equation 3.8 (Xiong et al., 2017):

$$M_{ij} = \cos\left(\vec{v}_{t_i}^{q}, \vec{v}_{t_j^d}\right) \tag{3.8}$$

in which $\vec{v}_{t_i}q$ and $\vec{v}_{t_j}d$ are the embedding of the query and document term i and j respectively.

Kernel pooling The kernels will transform the result of the translation matrix into query-document ranking features $(\phi(M))$(Xiong et al. [2017]). This conversion can be obtained by the equation 3.9:

$$\phi(M) = \sum_{i=1}^{n} \log \vec{K}\left(M_i\right) \tag{3.9}$$

Note that $\vec{K}\left(M_i\right)$ will apply K kernels to the i-th query word's row of the translation matrix and results in a k-dimensional feature vector. This fact has been shown in 3.10:

$$\vec{K}\left(M_i\right) = \left\{K_1\left(M_i\right), \ldots, K_K\left(M_i\right)\right\} \tag{3.10}$$

$K_k(M_i)$ are the RBF kernels which are obtained by the following equation (3.11):

$$K_k(M_i) = \sum_j \exp\left(-\frac{(M_{ij} - \mu_k)^2}{2\sigma_k^2}\right) \tag{3.11}$$

Learning to rank layer Finally, the learning to rank model is a multiple linear layer model with the *tanh* as the activation function (Xiong et al. [2017]). Therefore, the learning layer can be represented by the equation 3.12:

$$f(q,d) = \tanh\left(w^T \phi(M) + b\right) \tag{3.12}$$

in which the w and b are the weights and biases of the final linear layer respectively.

Learning process and back propagation process The cost function of this network is the same as the one used in the recurrent network.

The learning process is done through a back-propagation process. Back-propagating the gradients from the ranking features to the embedding layer is obtained by the following equation (equation 3.13). Based on gradients that have been received from the learning to rank layer, the kernels will pull the term similarities to their μ to increase the TF-counts or push them away. The force depends on the distance from the μ and the value of σ for each kernel (Xiong et al. [2017]).

$$g(M_{ij}) = \sum_{k=1}^{K} \frac{g(K_k(M_i)) \times \sigma_k^2}{(\mu_k - M_{ij}) \exp\left(\frac{(M_{ij}-\mu_k)^2}{-2\sigma_k^2}\right)} \tag{3.13}$$

Figure 3.7 shows the effect of kernel layers in the forward and in the back propagation of the gradient.

For the learning to rank layer (linear layer) the error is propagated back through the linear layer.

3.4.3.2 Convolutional Kernel-Based Neural Ranking Models (Conv-KNRM or CKNRM)

The most important difference between this network and KNRM is the use of a set of convolutional filters to form different n-gram embedding. Figure 3.8 illustrates the different layers and components of the network. In this figure the uni- and bi-gram embedding have been shown. Kernel pooling, learning to rank layer, and the cost function stay the same as for the previous network.

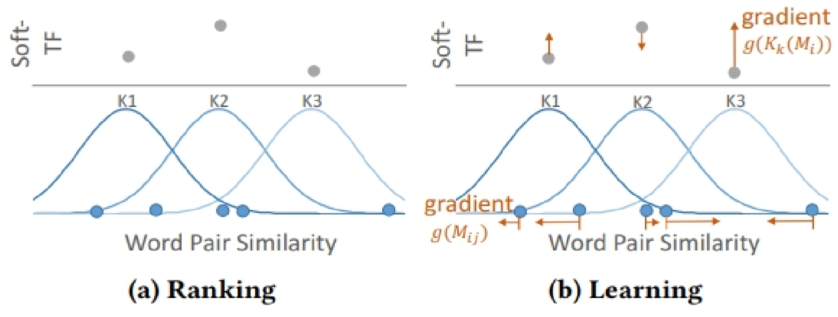


Figure 3.7: The effect of kernels in the ranking and learning process (Xiong et al. [2017])

N-gram Embedding and Cross-matching In convolutional Kernel-based Neural Ranking models (Conv-KNRM) the n-grams of the document and queries would be calculated through the convolutional filters. The convolutional filters slide over the text and compute a continuous score v for each window using the word embedding and the filter weights. This has been shown by the following equation (Dai et al. [2018]):

$$v = w \cdot T_{i:i+h}, v \in \mathbb{R} \tag{3.14}$$

where the filter window has the size of h and w represents the filter weights. Exploiting F filters of the size h, combining the results using a linear layer, and using a relu activation function will contribute to a representation with the dimensionality of F. The F dimension embedding for the h-gram would be obtained by the equation below (3.15) (Dai et al. [2018]):

$$\vec{g}_i^h = relu\left(W^h \cdot T_{i:i+h} + \vec{b}^h\right), i = 1 \dots m \tag{3.15}$$

where $\vec{g}_i^h \in \mathbb{R}^F$ is the embedding of the ith dimension for the h-gram.

Cross-Matching In this layer the similarity of the query and document n-grams is calculated using the cosine similarity (Dai et al. [2018]). The equation for each element resulting from the cross-matching is gained by equation 3.16 (Dai et al. [2018]):

$$M_{i,j}^{h_q,h_d} = \cos\left(\vec{g}_i^{h_q}, \vec{g}_j^{h_d}\right) \tag{3.16}$$

Note that if the dimensionality of n-gram embedding is considered to have the dimensionality of h_q and h_d for the query and document respectively, then the resulted cross-matching matrix has h_q by h_d dimensions.

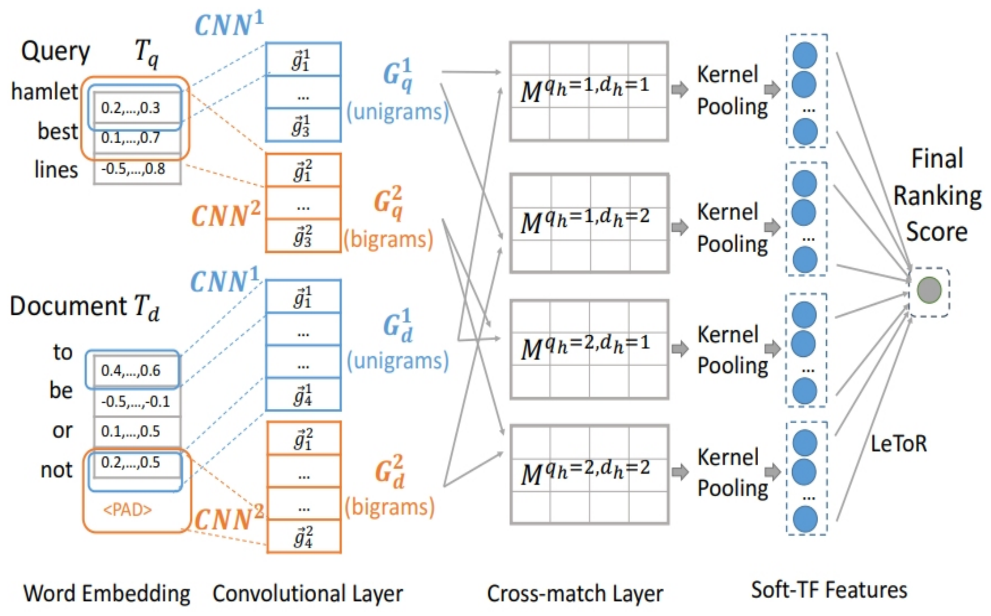


Figure 3.8: Different layers of the Conv-KNRM structure (Dai et al. [2018])

3.4.4 BERT: A Contextualized Language Model

In order to achieve to a better performance for the ranking task, we try to get a better understanding of the text. Due to MacAvaney et al. [2019] contextualized embedding would help us getting to such goal. Unlike the traditional embedding such as word2vec or glove, contextualized language models will consider the context of appearance of a word in order to assign it an embedding. For instance the word *bank* may have different representations in different sentences depending on the context of the occurrence.

Among the contextualized embedding techniques, BERT (Bidrectional Encoder Representation from Transformers) has represented one of the best performances in different NLP tasks. According to Devlin et al. [2018] BERT is trained with a masked language model task and can be fine tuned for different downstream NLP tasks. In Masked Language Model (MLM) objective, some of the tokens are masked randomly and the model tries to predict the id of the original token based on its context. This objective enables the model to encode the right and left contextual information of the tokens in the representation. BERT provides us with the opportunity to encode multiple text segments and considering this ability, we can make judgments about text pairs (MacAvaney et al. [2019]).

According to Vaswani et al. [2017], BERT network is the result of stacking transformer layers which has been illustrated in 3.9. Each transformer unit

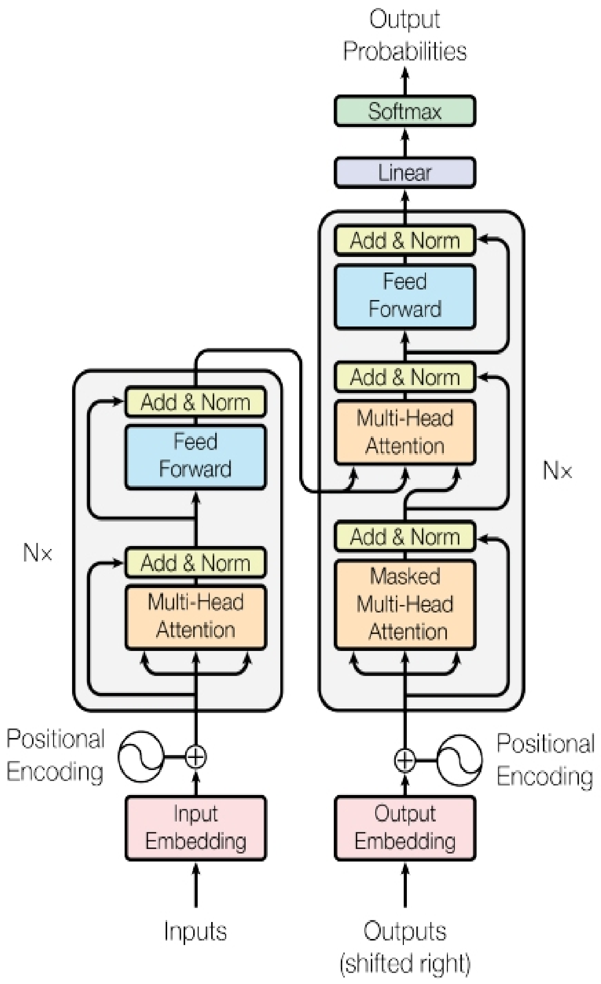


Figure 3.9: Structure of Transformer (Vaswani et al. [2017])

has an encoder (left half) and decoder (right half of figure 3.9). The **Encoder** is composed of 6 layers each of which is composed of 2 sublayers. Firstly a self-attention mechanism and secondly is position-wise fully connected feed-forward network (a two linear layer network with ReLU activation function) (Vaswani et al. [2017]). The **Decoder** is also a 6 layer model where each layer has the 2 sub-layers as in the encoder plus a third sub-layer: a multi-head attention over the output of the encoder stack. In this study, we have utilized a BERT base uncased model with the hidden layers of 12 and the hidden size of 768. Note that for the implementation of the neural ranking models with the contextual embedding can

In the following subsections we will discuss about the neural ranking models whose embedding is based on BERT. Note that the model implementations are accessible in GitHub[5].

3.4.4.1 Vanilla BERT

According to MacAvaney et al. [2019] this network is obtained by the fine-tuning of the BERT model with a linear layer stacked at top. This means that a linear layer would be at the top of BERT model and for the training process we use larger learning rate for the linear layer and a smaller value for modifying the pretrained BERT weights. Note that the loss function is the pairwise loss typical for the ranking tasks.

3.4.4.2 BERT and DRMM

With the aim of aggregating the ranking knowledge and the contextualized embedding, we can stack any neural ranking model on the top of BERT model (MacAvaney et al. [2019]). In this study, we have put DRMM model to see how the performance will change.

3.4.4.3 BERT and KNRM

The other neural ranking model which uses the contextualized embedding in our study is KNRM. As we have used the model with static embedding, this network will give us a good illustration of how the contextualized embedding will effect the performance of the model.

3.4.5 Standalone Neural Ranking Model

All the networks that have been discussed up to now require some candidate documents for the test query to give some results. This means that they typically do a re-ranking for the top document hits from a traditional retrieval model. Consequently, it is possible that the performance of the model gets restricted to what the first ranker would provide. In other words, an error caused by the first ranker is propagated to the following ranker. Guo et al. [2019] called this phenomena as error propagation in cascade structures. To overcome this problem Zamani et al. [2018] suggested the Standalone Neural Ranking Model (SNRM).

The main important innovation of this network is the latent sparse representation of the document and query for constructing an inverted index. Meanwhile, this representation grabs the semantic relationships between the

[5]https://github.com/Georgetown-IR-Lab/cedr

query and documents (conclusions and premises in our study) (Zamani et al. [2018]).

The retrieval process is defined as finding the documents which have non-zero elements for the non-zero elements of the query representation. Zamani et al. argued that considering the retrieval process, the sparser the query representation, the faster the retrieval would be. It is also clear that the queries are typically of a shorter length compared to the documents. As a result, assuming the same representation dimension for the documents and query, the query representation should have much more zero elements compared to the document representation. For the model implementation we have used its GitHub repository[6].

3.4.5.1 Model Architecture

Based on the aforementioned facts for the query and document latent space representation, Zamani et al. suggested a model which learns the representation of the n-grams for the query and documents. The learned sparse representation is in fact the average pooling of the sparse representation of different n-grams (Zamani et al. [2018]). Equation 3.17 states the sparse representation of document n-grams with the length of $|d|$. This is how the query representation can also be obtained (Zamani et al. [2018]):

$$\phi_D(d) = \frac{1}{|d| - n + 1} \sum_{i=1}^{|d|-n+1} \phi_{ngram}\left(w_i, w_{i+1}, \cdots, w_{i+n-1}\right) \tag{3.17}$$

In this equation, ϕ_{ngram} is a fully connected feed-forward network which learns the sparsity representation for a given n-gram with the terms $w_1, w_2, \cdots, w_{|d|}$. Note that using n-grams as the input to the model helps the network get the interaction relation and term dependencies between the words (Zamani et al. [2018]).

The hourglass shape of the ϕ_{ngram} forces the input information gets dense in the middle layers and then it goes through the later layers for getting a high dimension (e.g. 20000). Figure 3.10 shows the structure of the network for creating a sparse representation of the input text.

3.4.5.2 Training and Loss Function

For the training phase the query and the related and unrelated documents are given to the network. For the related document a label of 1 and for the unrelated ones -1 would be assigned. Figure 3.11 illustrates the training phase of the network.

[6]https://github.com/hamed-zamani/snrm

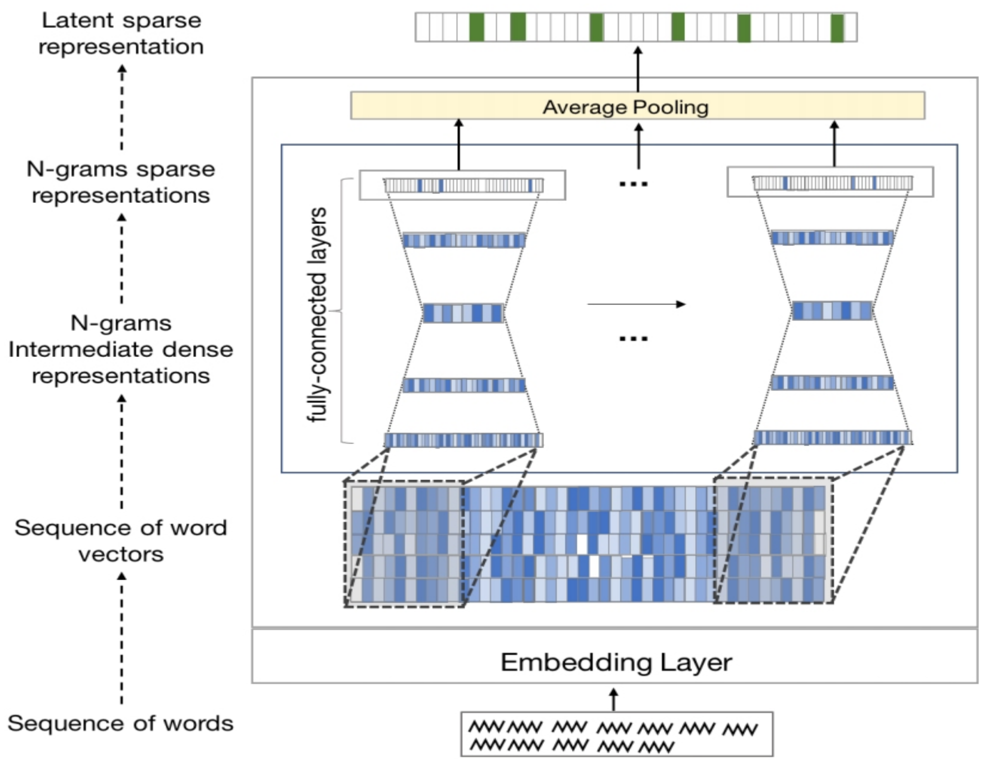

Latent sparse representation
N-grams sparse representations
N-grams Intermediate dense representations
Sequence of word vectors
Sequence of words
Average Pooling
fully-connected layers
Embedding Layer

Figure 3.10: Sparse representation network (Zamani et al. [2018])

The loss function for this network has two terms: retrieval term and sparsity term (Zamani et al. [2018])) which are explained in the following sub-sections. Note that in contrast to a typical auto-encoder-decoder, the representation network in this model does not have a reconstruction term in its cost function as we do not plan to recusntruct the input. Rather than that, a sparse representation is of our interest.

Retrieval Objective This term of the loss function is the one that other neural ranking models also have. Hinge is a kind of pair-wise loss which is used and is achieved by the equation 3.18

$$\mathcal{L} = \max\left\{0, \epsilon - y_i\left[\psi\left(\phi_Q\left(q_i\right), \phi_D\left(d_{i1}\right)\right) - \psi\left(\phi_Q\left(q_i\right), \phi_D\left(d_{i2}\right)\right)\right]\right\} \tag{3.18}$$

where ϵ is a hyper parameter for defining the margin of the related and the unrelated documents. This kind of pair-wise loss function tries to put the related documents on a higher rank comparing to the unrelated ones with a margin.

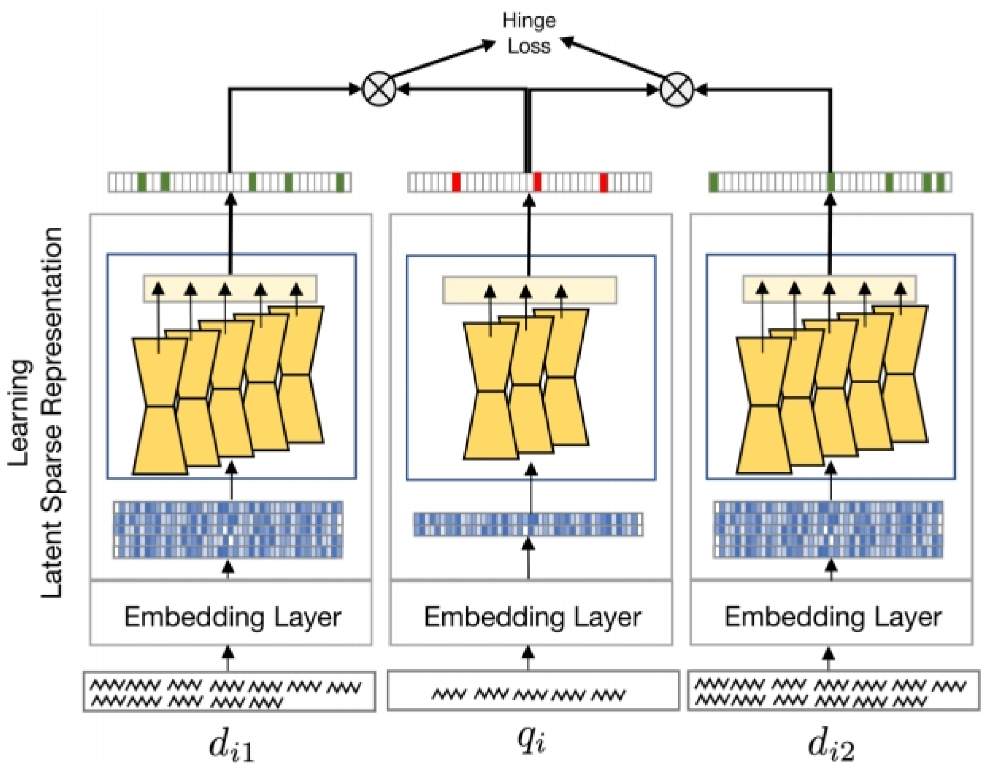


Figure 3.11: Training process of SNRM (Zamani et al. [2018])

Sparsity Objective As we have pointed out before, sparsity is an important parameter specially for query representation. Sparsity means the ratio of the number of zero elements to the total number of elements in the representation. For a representation vector $\vec{v}$ This can be demonstrated by the equation 3.19:

$$\text{sparsity ratio } (\vec{v}) = \frac{\text{total number of zero elements in } \vec{v}}{|\vec{v}|} \tag{3.19}$$

Zamani et al. argued that optimizing the L_0 norm of the representation vector could contribute to the maximum sparsity but the point is that taking the derivative of such norm is not possible as it is non-differentiable. Alternatively they suggested minimizing L_1 which can be stated as:

$$L_1(\vec{v}) = \sum_{i=1}^{|\vec{v}|} |\vec{v}_i| \tag{3.20}$$

Loss Function By concatenating the two terms explained above we get the loss function used in SNRM which can be stated by 3.21 (Zamani et al. [2018]):

$$\mathcal{L}\left(q_i, d_{i1}, d_{i2}, y_i\right) + \lambda L_1\left(\phi_Q\left(q_i\right) \| \phi_D\left(d_{i1}\right) \| \, \phi_D\left(d_{i2}\right)\right) \tag{3.21}$$

where $||$ means the concatenation operator and the hypert-parameter λ controls the sparsity of the representations.

3.4.5.3 Document Retrieval

After the model has been trained by the related, unrelated documents and queries, we can create sparse representation of the documents offline for the purpose of inverted index construction. Note that the invert index of the documents are constructed not based on the words but the elements of the latent space. The score of each document can for a given query in the test phase can be calculated by the equation 3.22 (Zamani et al. [2018]):

$$\text{retrieval score}\ (q, d) = \sum_{\vec{q_i}|>0} \vec{q_i}\vec{d_i} \tag{3.22}$$

The retrieval score is in fact the dot product of the query and document representation in the latent space. Figure 3.12 shows the retrieval process of the documents for the given query and the index construction.

3.5 BM25

After the models have been prepared in the training phase, for a given query in the test phase, they still require some candidate documents, as the process of giving ranking score to all of the documents in the corpus would be computationally expensive and time consuming. In order to provide the queries with the related documents (or better to say premises) we have retrieved the related conclusions using BM25 (Best Matching) retrieval model. The trained networks try then, to re-rank the premises corresponding to the retrieved related conclusions. Compared to premise retrieval which will be conducted by the trained models, this phase of related conclusion retrieval is fast (considering the length of the conclusions).

BM25 is a probabilistic retrieval model which uses some attributes such as term frequency, document frequency and document length, in order to calculate the relevance score for each document given the query terms. If we assume the query terms as q_1, ..., q_n the ranking score for the document D can be calculated by the following equation (Pérez-Iglesias et al. [2009]).

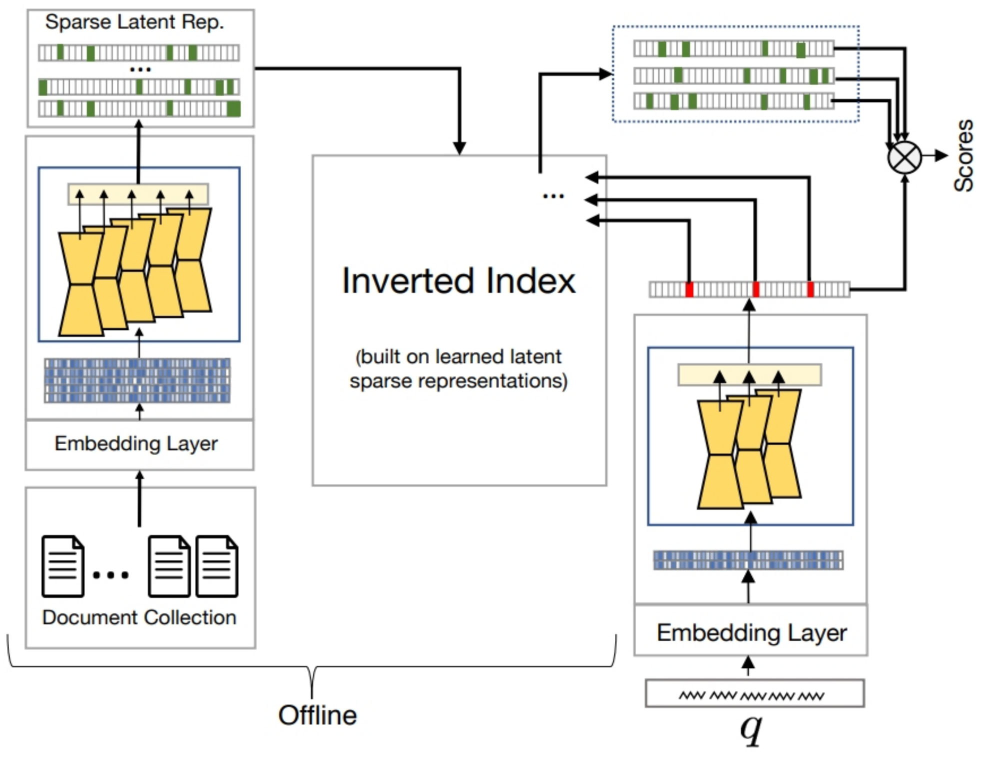


Figure 3.12: Document retrieval process (Zamani et al. [2018])

$$\operatorname{score}(D, Q)=\sum_{i=1}^{n} \operatorname{IDF}\left(q_{i}\right) \cdot \frac{f\left(q_{i}, D\right) \cdot\left(k_{1}+1\right)}{f\left(q_{i}, D\right)+k_{1} \cdot\left(1-b+b \cdot \frac{|D|}{\text { avgdl }}\right)} \tag{3.23}$$

where $|D|$ is the length of the document D in words and *avgdl* is the average length of the documents in the corpus. b and k_1 are constant and for our study the values are 0.75 and 1.5 respectively. $f(q_i, D)$ is the frequency of the query term q_i in the document and the IDF is given by the equation 3.24:

$$\operatorname{IDF}\left(q_{i}\right)=\log \frac{N-n\left(q_{i}\right)+0.5}{n\left(q_{i}\right)+0.5} \tag{3.24}$$

where N is the number of documents in the collection and $n(q_i)$ is the number of documents having the query term q_i.

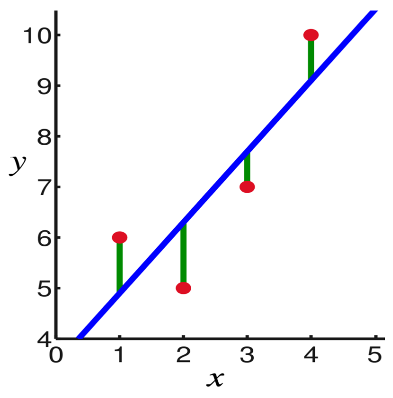


Figure 3.13: Linear regression

3.6 Aggregation

In order to improve the performance of the ranking and aggregate the intuition behind the networks used in this study, we performed a regression among the scores calculated by different models for each of the documents. The aggregation is done through a linear regression. The linear regression is fitted to the validation and then the final scores are interpreted for the given test queries.

For the validation queries we calculate the relative scores of the related and unrelated premises using the networks. In order to get the scores in the range of [0,1] we do a normalization using the following formula:

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{3.25}$$

By fitting a linear regression, we assume that there is a linear relation between the network scores and the similarity labels of the conclusion and premises. Figure 3.13 shows the main idea of linear regression in which we try to minimize the error (in our case mean squared error) between the estimated line and the real label values.

Note that in our regression problem, the labels get the value of 0 and 1 (for unrelated and related documents respectively) and the normalized similarity scores of the networks (x-axis) vary between 0 and 1. In the linear regression problem the feature space is the normalized network scores. As we aggregate the result of 7 networks (excluding SNRM), the feature space is 7.

Finally, considering the ranking scores as the feature space, we apply the linear regression on the normalized calculated similarity scores of the documents to get the final score for the given test query.

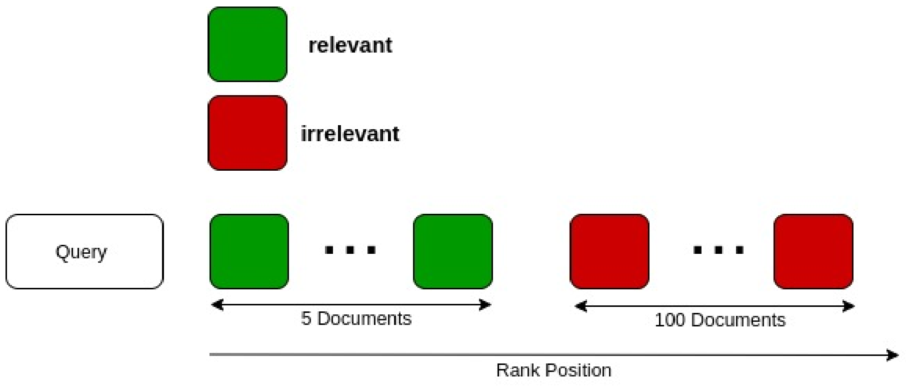


Figure 3.14: An ideal ranking for a validation query

3.7 Validation Metrics

In this study we have used three well-known metrics for the task of ranking namely MRR, MAP, and nDCG. Note that the validation data in our study are arguments whose conclusions have 5 premises as the related and 100 unrelated premises. As a result, for each validation query (conclusion) we rank 105 documents (premises). Figure 3.14 shows the ideal ranking for a validation query. We then take the top 20 hits and calculate the metrics on them.

3.7.1 Mean Reciprocal Rank (MRR)

Based on the definition that Radev et al. [2002] provides, reciprocal ranking is the multiplicative inverse of the rank of the first correct answer. The mathematical equation for this statement is shown by the equation 3.26:

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \tag{3.26}$$

where $rank_i$ refers to the rank of the first relevant document for the i-th query. Figure 3.15 can better explain the concept of this measure through an example. For the first query, the model provides a ranking for 3 documents. The very first relevant document happens to be ranked as the third place. Because of that we assign $\frac{1}{3}$ to the model for this query. The scores for different queries are calculated, and finally we take the average of all the scores.

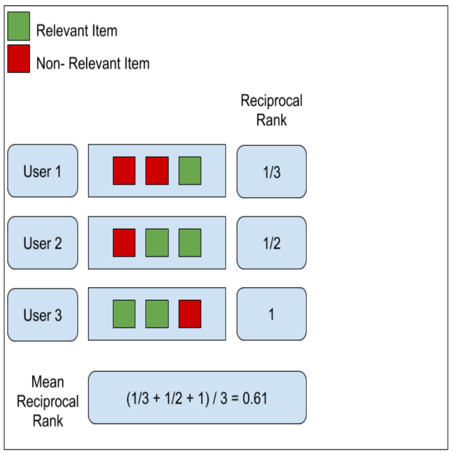


Figure 3.15: An example of MRR calculation

3.7.2 Mean Average Precision (MAP)

According to Turpin and Scholer [2006], the average precision for a query can be calculated by the equation 3.27:

$$AveP = \frac{\sum_{k=1}^{n}(P(k) \times rel(k))}{\text{number of relevant documents}} \tag{3.27}$$

where $rel(k)$ is a function which returns 1 if the item in the rank k is relevant and 0 otherwise. $P(k)$ is the precision at the cut-off k. When the score for each query has been calculated we take the mean of all of them which is reflected in the equation 3.28:

$$MAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q} \tag{3.28}$$

where Q is the total number of queries. Figure 3.16 demonstrates the concept of MAP through a simple calculation.

3.7.3 Normalized Discounted Cumulative Gain (nDCG)

Järvelin and Kekäläinen [2002] used graded relevance values for the documents in order to produce model scores for the queries. The main intuition in their definition is to punish the highly relevant documents which have not been ranked at the top list. The DCG at a particular rank position p is given by

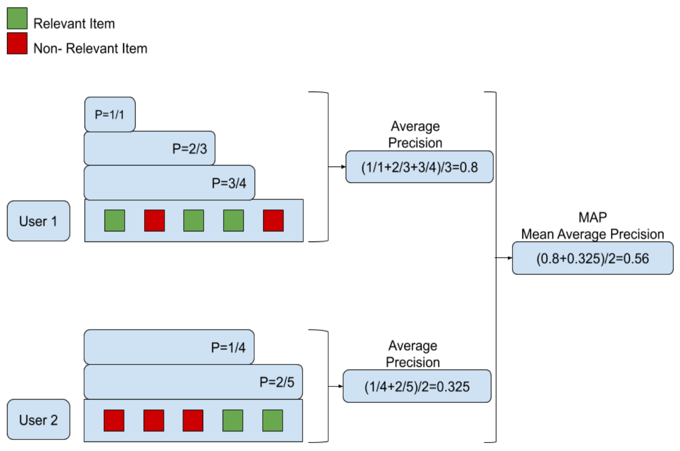


Figure 3.16: An example of MAP calculation

the equation 3.29:

$$DCG_p = \sum_{i=1}^{p} \frac{rel_i}{log_2(i+1)} \tag{3.29}$$

Note that the punishment of the high relevant documents ranked at lower rank positions is done through a logarithmic term in the denominator.

The normalized DCG can be obtained by dividing the obtained value of DCG_p by the ideal DCG ($IDCG_p$). This normalization has been shown by the equation 3.30:

$$nDCG_p = \frac{DCG_p}{IDCG_p}. \tag{3.30}$$

In this study, as we do not have a graded relevant values for the premises and conclusions, we made use of a binary version of the formula. This means that the documents are either relevant or irrelevant to the query.

Chapter 4

Experiments and Results

This chapter discusses the experiments and also the results that have been achieved in each step of this study. After a brief explanation of forming the training and validation set from the result of preprocess phase, this chapter demonstrates the training and validation phase of the models, the parameters that have been used and the resulting curves. This chapter also explains how to the inference phase is performed in the neural ranking models in the test phase. By having the network results for the queries, we explain how we can aggregate the results so that we can improve the ranking for the test queries. The model scores for the test queries and an analysis of it is the last section of this chapter.

4.1 Training and Validation Data

After parsing the json file of the collection and taking the id, conclusion, and premise parts, applying the preprocessing steps explained in the previous chapter, we are left off with 2 pandas dataframes for the train and validation arguments. Each row in the tables represent an argument. The columns also represent the attributes of the arguments which are:

- id: which is unique for each argument
- conclusion: the conclusion of the arguments
- normalized conclusion: the conclusion of the argument with stop words removed
- premise: the premise unit of the argument
- unrelated id: the id of an argument with an unrelated premise to the conclusion (For the validation arguments we have 20 unrelated ids)

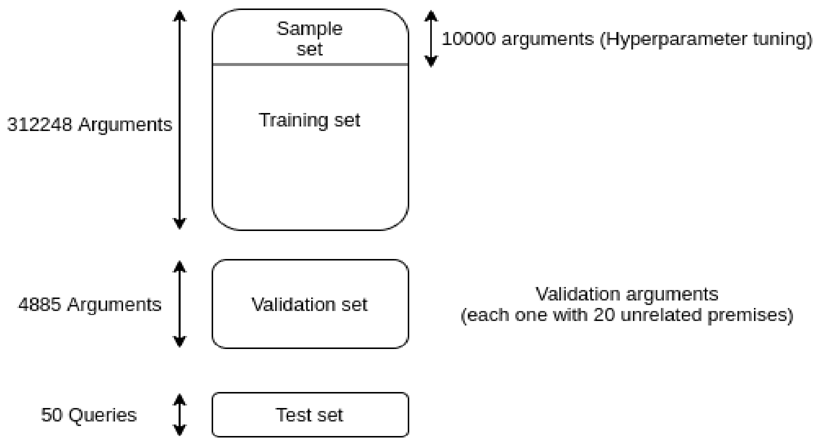

Sample set
Training set
312248 Arguments
10000 arguments (Hyperparameter tuning)
4885 Arguments
Validation set
Validation arguments (each one with 20 unrelated premises)
50 Queries
Test set

Figure 4.1: Different datasets and their number of arguments

- unrelated premise: the unrelated premises to the conclusion (the way of assigning the unrelated premises have been discussed in 3.2.3) (For the validation arguments we have 20 unrelated premises)

We have 312,248 arguments for the training and 4,885 arguments for the validation set. We tried to keep the validation set small in order to incorporate more information in the training phase while still allowing a meaningful assessment of model performance during validation. Note that the validation set will be used later as the training set for the regression model which performs score aggregation. For training the aggregation model we have just used positive and one of the negative premises of the arguments. Figure 4.1 shows the different sets that have been created in this study with their corresponding number of arguments.

As the pre-processing phase of the contextualized embedding networks is a bit different (they do not require the punctuation to be tokenized), we formed two separate train and validation table for these networks. Note that the train and validation arguments are the same for these sets so that the results could be comparable.

4.2 Model Training

Now that we have training and validation sets, we can form the tensors to feed to the network for the training and the validation phase. We have taken 10000 random arguments from the training set to tune the hyper-parameters of the models to achieve best performance for minimizing the cost function.

This sample data is also used to inspect whether the training and validation process are executed correctly without any error. Debugging these processes with smaller number of data would be easier and requires less time.

Table 4.1 shows the result of the tuning of the most important model hyper-parameters. BDRMM stands for the DRMM with the contextualized embedding (which in our case is BERT). Same explanation holds for BKNRM. VBERT is Vanilla BERT network which is the result of stacking linear layers to the BERT model. Note that the notation of x:y for the linear layers row in the table illustrates the number of units in two consecutive fully connected (linear) layers. For the case of not having a hyper-parameter by a model we have used the symbol "x" for it.

Table 4.1: Hyper-parameters of the Networks

Parameters	GRU	DRMM	KNRM	CKNRM	VBERT	BDRMM	BKNRM	SNRM
word-embedding	300	300	300	300	768	768	768	300
hidden-size	512	x	x	x	x	x	x	x
hidden-layer	1	x	x	x	x	x	x	x
linear-layer	2048:1	1024:256:5:1	21:1	21*9:1	768:1	911:5:1	911:5:1	x
dropout	0.5	0.2	0.2	0.2	0.2	0.2	0.2	0
learning rate	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.0001
number of bins	x	1024	x	x	x	11	x	x
Kernel function	x	x	RBF	RBF	x	x	RBF	x
number of kernels	x	x	21	21	x	x	11	x
window sizes	x	x	x	1,2,3 * 300	x	x	x	x
output channels	x	x	x	128	x	x	x	x
learning rate BERT	x	x	x	x	$2 * 10^{-5}$	$2 * 10^{-5}$	$2 * 10^{-5}$	x
n-grams	x	x	x	x	x	x	x	5
regularization coeff	x	x	x	x	x	x	x	0.001

Now that we get the models running with a good performance in reducing the cost function it is time to feed in the original training dataset. The original training dataset with 312,248 arguments have been fed to the networks. Considering the model and data sizes and the complexity of the computations, all the computations are done on a GPU to make the computations faster in a parallel way. This means that for the training and validation every input batches and models should be transferred into GPU. The following algorithm 1 shows the training and validation phase of the models.

Algorithm 1: Model training and validation algorithm

```
Result: Model with the best validation result
model initialization;
forming the training and validation tensors;
best-score = 0;
while $n <$ number-of-epochs do
    train the model with the training set and modify the weights;
    if $n$ % validation-step == 0 then
        get the score for 105 premises for the validation query;
        generate the validation score;
        compute the mean of error over the previous training batches;
        if validation-score > best-score then
            best-score $\leftarrow$ validation-score;
            save model;
        end
    end
end
```

We keep the batch size to 32 for different networks. For all the networks, after 1239 training batches we run the validation to evaluate the performance of the network and if the MAP@20 measure was better than the best result obtained so far, the saved model will be replaced correspondingly. For the case of nDCG@20 measure a more realistic score could be obtained if the query-relevance information of the ranking task was soft. Based on the definition of MAP@20 and MRR@20, it is clear that the MAP@20 measure yields a more realistic score for the models as it takes into the consideration all of the retrieved documents and only on the very first related document.

In each epoch, we perform 8 evaluation runs and for the cases of networks with static embedding, we run train the models for 10 epochs. For the BERT models as it will be seen, there is no need to do the fine tuning for this many epochs. We have fine tuned the models with the contextualized embedding for 5 epochs. This saves the time and avoids complex computations out of which we do not get noticeable improvement.

The average error and validation curves for different networks are displayed for every evaluation that we have done on the validation set. This results in displaying 80 points for every single model with static embedding and 40 points for the BERT-based models. Note that the validation points are displayed in percentage and the coordinates of the best MAP@20 achieved in the corresponding run (the step number and the MAP@20 value) have been specified with a blue dot and written on the curves.

4.2.1 Recurrent Network

The embedding dimensionality for the input tokens is 300 and the learning rate of 0.001 represented reasonable results on the sample data. The hidden size for the GRUs have been selected to be 512. For the linear layer we have the dropout layer with the rate of 0.5. Dropout is a regularization technique in which we define a probability ratio by which, the neurons in the feedback path will be omitted. According to Srivastava et al. [2014] this may contribute to a more generalization for the network and is a technique for avoiding over-fitting. This means that the neurons will be excluded from the process of training in the back-propagation path with the probability of 0.5. Figure 4.2 shows the results of training loss and the validation phase of the model. Note that on the x-axis, the term "step" means every 1,239 training batches in which we do a validation.

4.2.2 DRMM

Based on the sample data we have decided to have 1,024 bins. This number of bins is also the number of units in the input of the feed-forward network. For the feed-forward network we exploited 2 hidden layers of 256 and 5 units. With the learning rate of 10^{-3} a learning curve which is illustrated in figure 4.3 could be achieved.

4.2.3 KNRM

After we run the network on the sample of 10,000 training set we decided to have 21 bins as it was suggested by Xiong et al. [2017]. Learning rate and word embedding dimensionality are as the same as recurrent network. Figure 4.4 shows the train and validation phase.

4.2.4 CKNRM

The parameters for the network are the same as for KNRM model. Convolutional layers are 2D filters whose input is of dimension 1 and the output has the dimensionality of 128. The window sizes of the convolution layers are 1, 2, and 3 as suggested by Dai et al. [2018]. The ReLu activation function has been applied on the output of the convolutional layers. Figure 4.5 shows the training loss as well as the validation phase for CKNRM.

4.2.5 Vanilla BERT

The learning rate for the BERT layers is much smaller than for the linear layer, as we do not intend to make large changes to the pre-trained contextualized embedding. We keep the learning rate of the BERT layers to be $2 * 10^{-5}$ and for the linear layer the learning rate is 10^{-3}. For the purpose of generalization we add a dropout layer with the probability of 0.1. The linear layer has the input size of 768 to 1. 768 is the embedding dimensionality for a token in BERT model. Figure 4.7 illustrates the training and the validation phase of the network.

4.2.6 BERT and DRMM

The learning rates for the BERT and non-BERT layers are the same as the Vanilla BERT. The number of bins is 11, as suggested by MacAvaney et al. [2019]. The feed-forward network is the same as the one used in DRMM. Figure 4.8 shows the training and validation curves.

4.2.7 BERT and KNRM

The Learning rate for the fine tuning of the BERT layers and training the KNRM layers are kept the same as for the Vanilla BERT model. The number of bins is 11, and the parameters for RBF functions are kept as what was suggested by the authors in MacAvaney et al. [2019] as the results on the sample data were acceptable. Figure 4.9 shows the loss function in the training phase as well as the results in the validation phase for different epochs.

4.2.8 SNRM

For this model we did not use any hidden layer and it showed reasonable results on the sample data. Learning rate is selected to be 10^{-4}, and no drop out was used. Figure 4.6 shows the loss curve for the training and the validation metrics during the training and the validation phase.

Table 4.2: Achieved evaluation scores of the models in the best MAP score

Metrics @20			
Model	MRR	MAP	nDCG
GRU	28.4	24.1	38.05
DRMM	67.00	52.8	64.77
KNRM	84.35	72.64	80.24
CKNRM	86.72	73.32	82.08
SNRM	82.41	70.14	78.97
Vanilla BERT	95.12	88.5	91.00
KNRM BERT	94.57	90.18	89.80
DRMM BERT	95.97	88.09	91.34

Table 4.2 summarizes the best validation results achieved from the models.

4.3 Test Queries

After training the models and getting the best one from the validation phase, it is time to give the models the test queries and see what documents would be ranked top. Except the SNRM model which has generated inverted index and can retrieve the documents on its own, other networks require to be provided with the candidate documents (premises). Inspired by Dumani [2019] approach, we also suggest a two stage argument retrieval process, in which BM25 will clarify the candidate premises and in the second step of the pipeline, the trained networks re-rank the candidate premises suggested by BM25 for the given test queries.

After getting the test queries in the format of XML, we have parsed them into Pandas dataframe. We applied the pre-processing steps used for training and validation arguments to form the normalized version of the test queries (punctuation removal and removing the stop words).

We first group all the arguments in the collection based on the normalized conclusion column. Using BM25 we retrieve the most relevant normalized conclusions. We select the top 100 normalized conclusions. The premises corresponding to retrieved normalized conclusions are the candidate documents to be ranked by the neural networks. Note that each of the normalized conclusion may have a different number of premises. Consequently, the number of documents to be re-ranked may vary for different test queries. Figure 4.10 shows how we provide the trained networks with the document-query pairs to

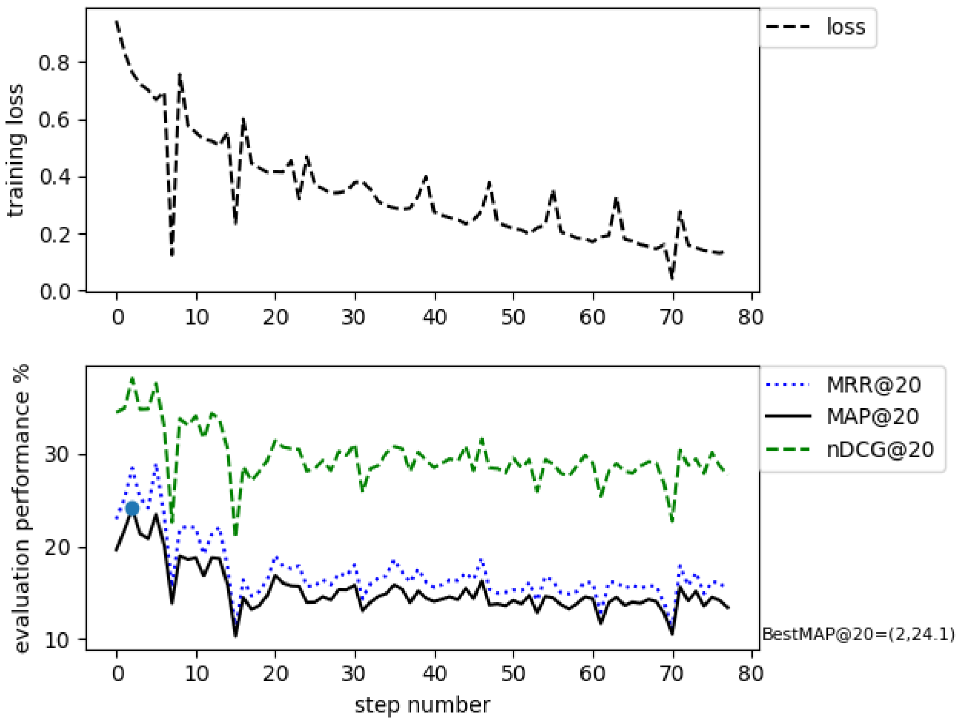


Figure 4.2: Train and validation curve - Siamese Network with GRUs

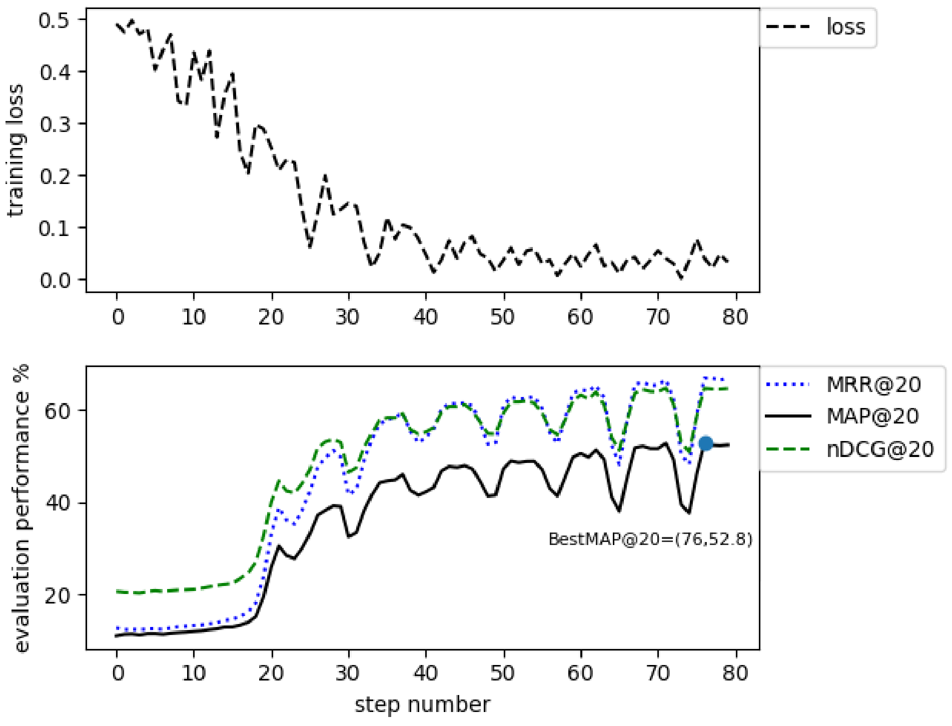


Figure 4.3: Train and validation curve - DRMM

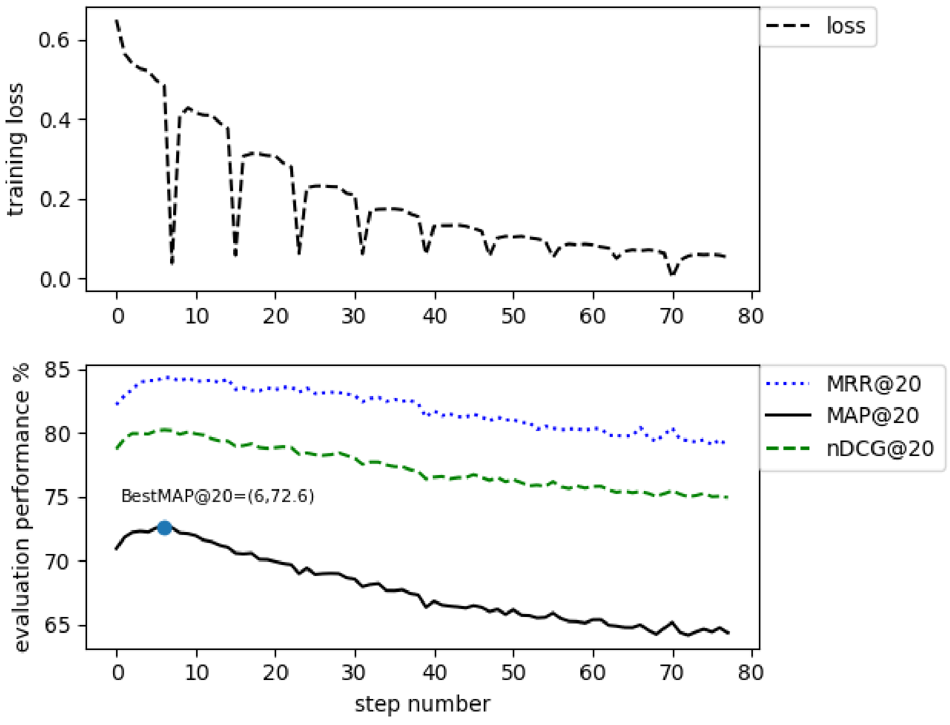


Figure 4.4: Train and validation curve - KNRM

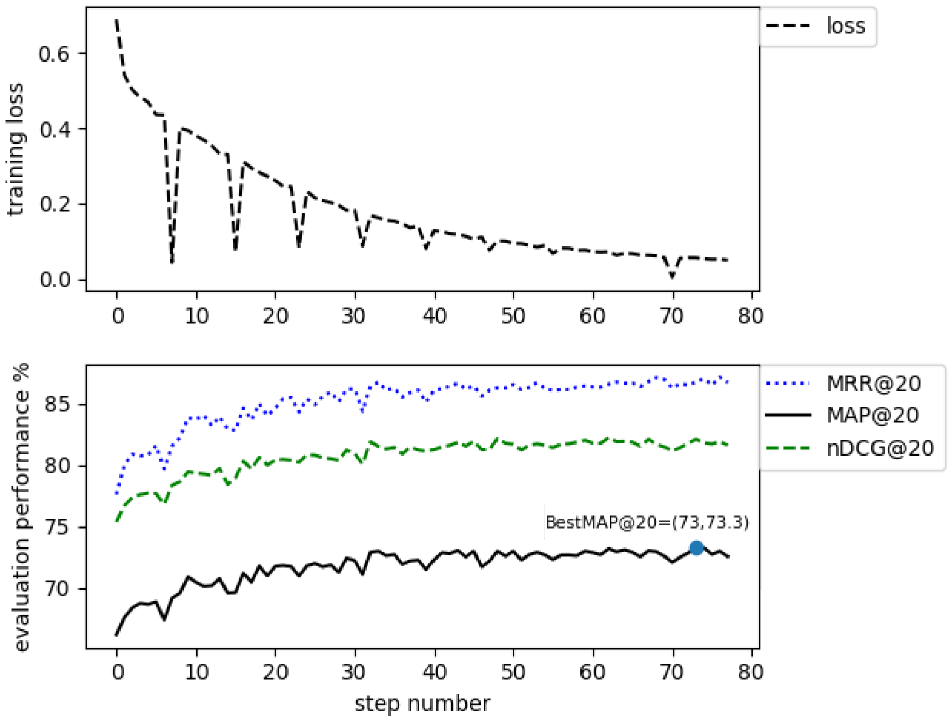


Figure 4.5: Train and validation curve - CKNRM

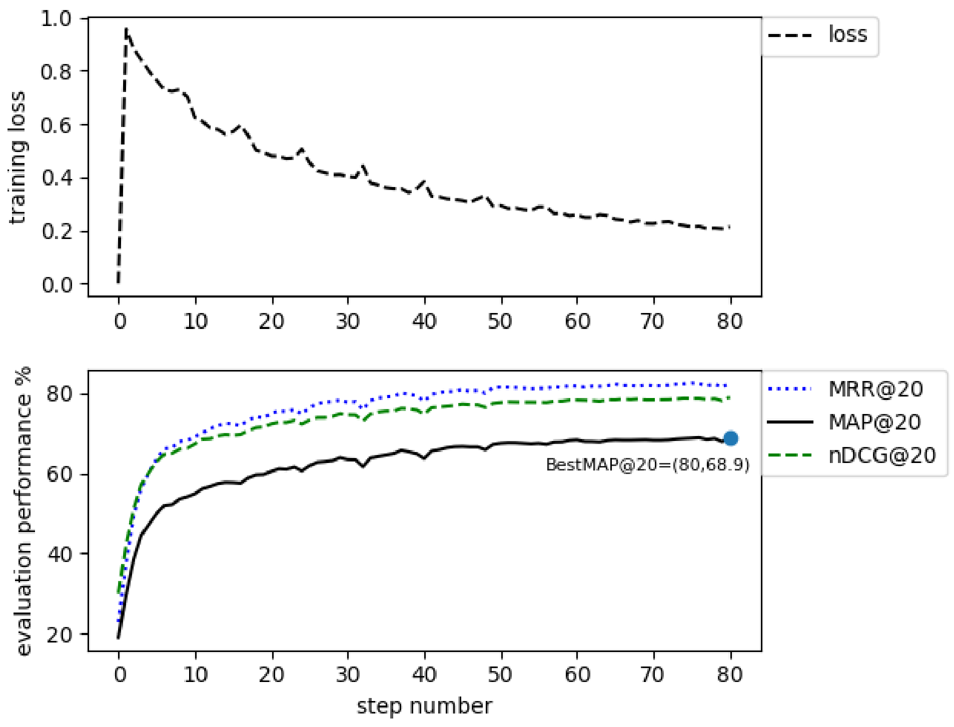


Figure 4.6: Train and validation curve - SNRM

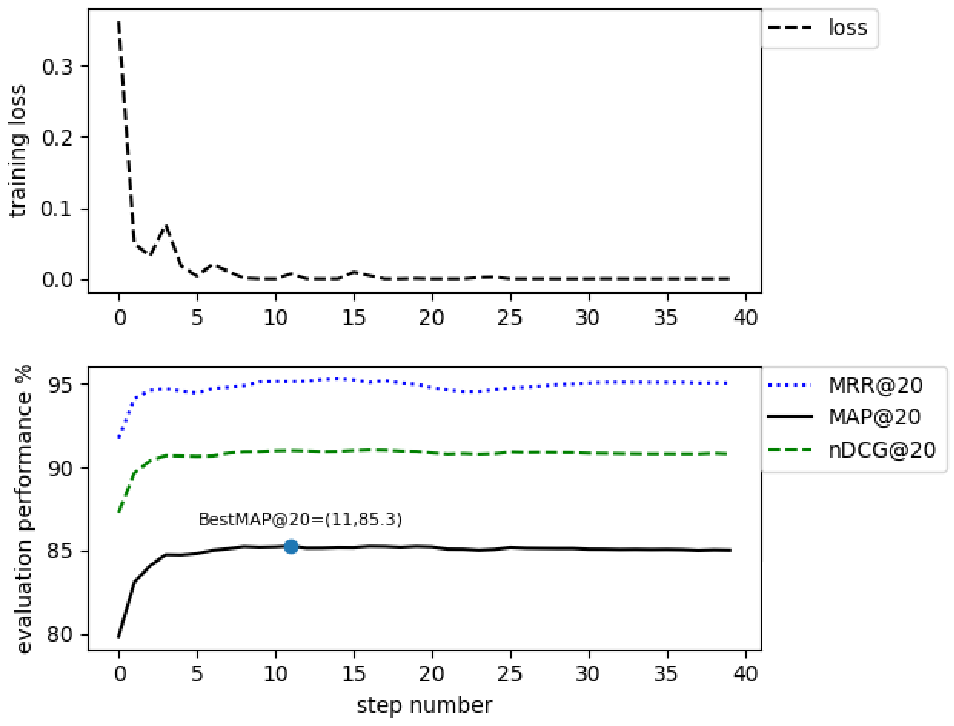


Figure 4.7: Train and validation curve - Vanilla BERT

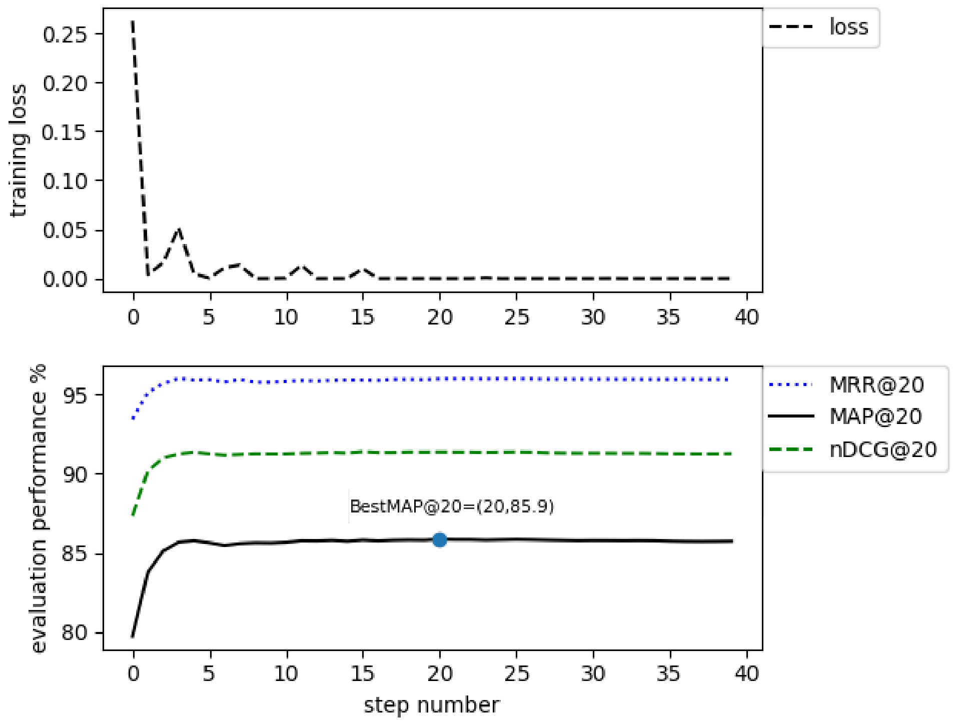


Figure 4.8: Train and validation curve - BERT and DRMM

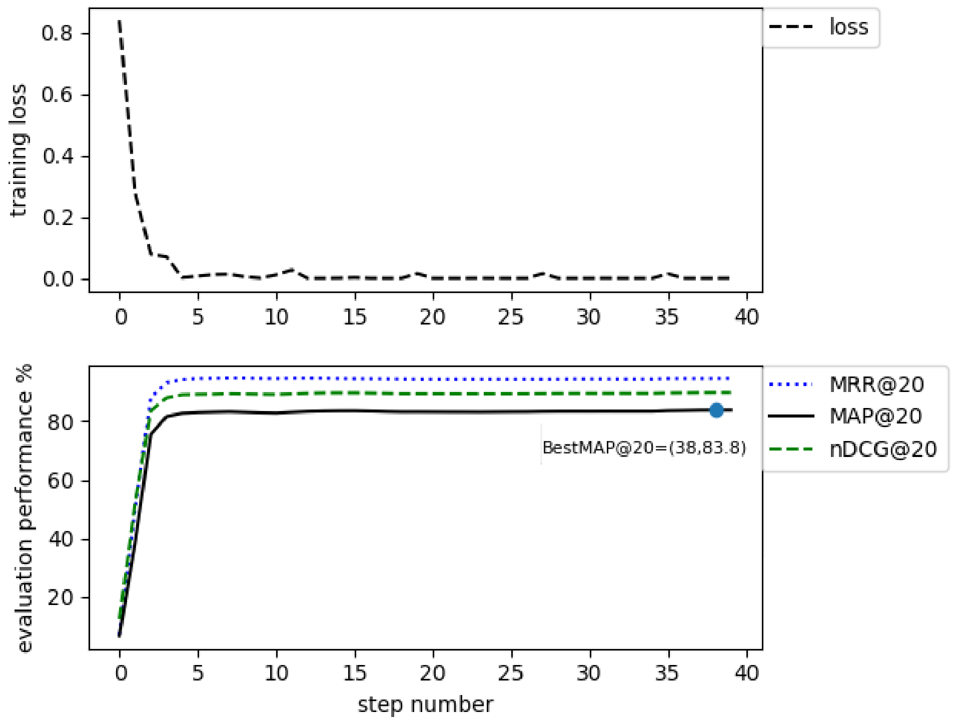


Figure 4.9: Train and validation curve - The stack of BERT and KNRM model

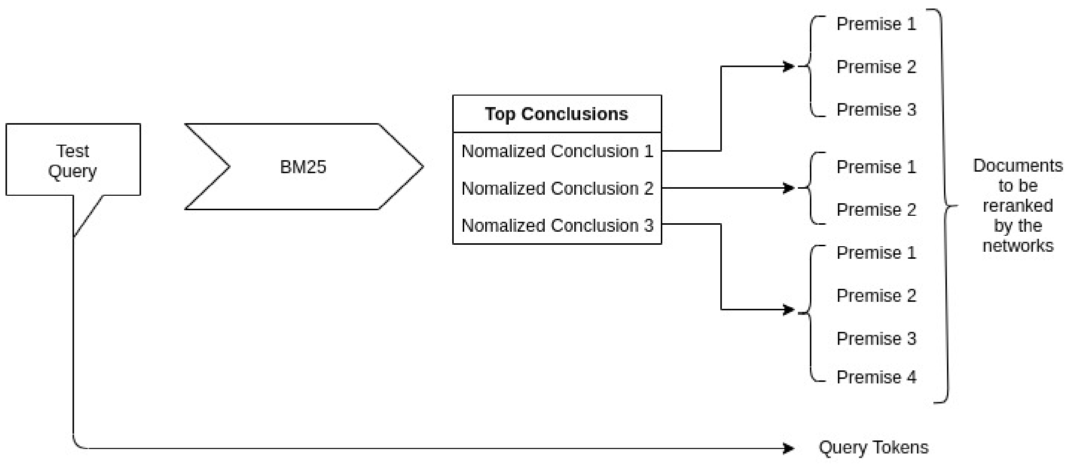


Figure 4.10: A two stage retrieval process for the inference phase: Candidate documents to be re-ranked by the networks are suggested by BM25

rank in the test phase.

After getting the document scores, we sort them based on the score in a descending order. We introduce the top 100 premises as the retrieved arguments for each test query. For the competition the results are saved in TREC format and for each trained model we provided a list of (50*100=5000) retrieved arguments. The fields in the prepared file for the test queries contained the following columns (each row belongs to an argument):

- The topic (query) number
- The document ID retrieved by the model (the argument ID of the retrieved premise)
- The rank of the document based on the ranking score
- The ranking score calculated by method
- Name of the retrieval method alongside the group name

In the appendix A we have shown the very first retrieved argument by each model for a sample test query.

4.4 Analysis of Documents Retrieved for Test Queries

In this section, we analyze how diverse the retrieved documents by different models in the test results are. As the 100 top hits would be different from model to model, for each test query, we have exploited 2 measures to analyze how the results overlap with each other.

Firstly, the Jaccard index will tell us to what extend the retrieved documents are in common for different models. The equation 4.1 shows that the coefficient measures the intersection of the results.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.1}$$

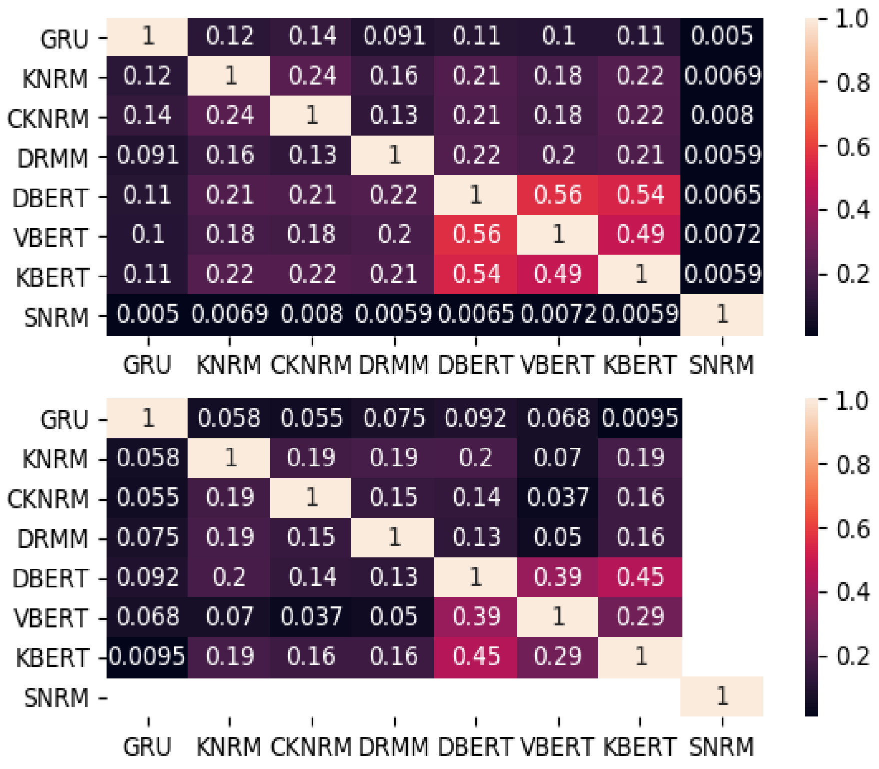
The second index is the Spearman correlation coefficient. It is in fact the Pearson correlation coefficient between the ranking variables (Daniel et al. [1978]). The following equation (4.2) shows how Spearman index can be calculated for two ranking sets rg_X and rg_Y with n samples (Daniel et al. [1978]).

$$r_s = \rho_{\mathrm{rg}_X,\mathrm{rg}_Y} = \frac{\operatorname{cov}(\mathrm{rg}_X, \mathrm{rg}_Y)}{\sigma_{\mathrm{rg}_X}\sigma_{\mathrm{rg}_Y}} \tag{4.2}$$

where σ is the standard deviation and cov is the covariance of the ranking variables.

The document scores achieved by the models for each test query can be considered as a vector of scores whose dimensions are the documents (arguments) that have been retrieved by the corresponding model. In the first step, we calculate the explained ranking similarity measures over the achieved results by each model pair for each test query. In order to visualize the results, we have taken the mean of the indexes calculated over the 50 test queries. The heat-map of the resulting calculation is illustrated by the the figure 4.11. The upper one is the mean of the Jaccard for 50 queries and the lower one is for the Spearman index.

The ones on the diagonals of the heat maps mean that both indexes produce 1 for two identical sets. It is clear that the networks which are based on the contextualized embedding have produced more similar results rather than the other networks. SNRM which retrieves the documents without using preliminary ranking has retrieved pretty different results compared to other networks. The Spearman coefficient for the SNRM model doesn't produce a number which means that the number of documents in common are extremely low that no coefficient can be calculated. Kernel based networks (KNRM and CKNRM) have more documents in common when we compare their retrieved documents with the ones retrieved by other models. The Jaccard index



	GRU	KNRM	CKNRM	DRMM	DBERT	VBERT	KBERT	SNRM
GRU	1	0.12	0.14	0.091	0.11	0.1	0.11	0.005
KNRM	0.12	1	0.24	0.16	0.21	0.18	0.22	0.0069
CKNRM	0.14	0.24	1	0.13	0.21	0.18	0.22	0.008
DRMM	0.091	0.16	0.13	1	0.22	0.2	0.21	0.0059
DBERT	0.11	0.21	0.21	0.22	1	0.56	0.54	0.0065
VBERT	0.1	0.18	0.18	0.2	0.56	1	0.49	0.0072
KBERT	0.11	0.22	0.22	0.21	0.54	0.49	1	0.0059
SNRM	0.005	0.0069	0.008	0.0059	0.0065	0.0072	0.0059	1

	GRU	KNRM	CKNRM	DRMM	DBERT	VBERT	KBERT	SNRM
GRU	1	0.058	0.055	0.075	0.092	0.068	0.0095	
KNRM	0.058	1	0.19	0.19	0.2	0.07	0.19	
CKNRM	0.055	0.19	1	0.15	0.14	0.037	0.16	
DRMM	0.075	0.19	0.15	1	0.13	0.05	0.16	
DBERT	0.092	0.2	0.14	0.13	1	0.39	0.45	
VBERT	0.068	0.07	0.037	0.05	0.39	1	0.29	
KBERT	0.0095	0.19	0.16	0.16	0.45	0.29	1	
SNRM								1

Figure 4.11: The heat map of the Jaccard (upper) and Spearman (lower) correlation coefficient for the 50 test queries

calculated between recurrent based siamese network and the other networks demonstrates that the number of retrieved documents in common is low. The Spearman coefficients are also too small.

This heat map will help us to understand which models could be included for the aggregation and which not.

4.5 Aggregation

Now that we have the top 100 arguments for each query resulting from 8 different networks and visualized how diverse the model results are for the test queries, we can aggregate the network results. Note that we do not include the SNRM results in the aggregation as the produced results are pretty different from the ones offered by the other networks. We consider the results produced by SNRM as an outlier for the task of regression.

We train the linear aggregation model on the validation set. This means that fitting the regression model happens on the scores achieved by the models for the conclusion of the validation arguments and their correponding premises and the validation conclusion and one of the unrelated premises. We then normalize the ranking scores and fit the linear model to the normalized scores. Note that the regression model happens in a space whose dimensions are the normalized model scores. In our case, as we ommit the SNRM model from participating in the final aggregation, we have a dimensionality of 7 for the regression model. We have (2 * 4885 = 9770) query-document pairs for training the regression model. For the related premises to the query we assigned a label of 1 and otherwise 0.

For each test query we take the union of all the retrieved documents by different networks. If a document does not exist in the list of retrieved arguments by a specific model, we assign a score of zero to the corresponding dimension. The final score of each document is the estimation of the linear regression model on the model scores between the test query and that specific argument.

4.6 Test results

Evaluation of the test results is done by the competition committee. They have manually annotated the retrieved arguments based on the argument quality dimensions such as general topic relevance, logically cogent arguments, rhetorically well-written arguments and utility (e.g. To what extend the retrieved argument can be used for building a stance regarding to a topic). The performance of the models is stated by taking the mean of nDCG@5 over the 50 test queries. This means that for each test query the 5 very top hits of the models have been evaluated. The performance of the models developed by the competitors were not that promising. This could be realized by the fact that the most of the performances were bellow the baseline model performance. The baseline model is Drichlet language model and its nDCG@5 score is 75.6%. Table 4.3 shows the performance of the utilized model for the test queries. The symbol "x" has been used for the models whose performances are not reported by committee. We assume that for the excluded models the performances were not higher than the ones displayed in table 4.3.

Generally speaking, although the result of the validation scores had a predictable behavior (the performance of the models on the validation sets improved over different models as we expected), such behavior cannot be realized

Table 4.3: Test scores of the models

Model	nDCG@5 (%)
GRU	x
DRMM	x
KNRM	68.4
CKNRM	x
SNRM	x
Vanilla BERT	40.4
KNRM BERT	31.9
DRMM BERT	37.1
Aggregation	37.2

in the test results. There is no correlation between the test and validation results. This could be mostly related to the training and the validation phase. Based on the limitation that we had on the annotated data and the lack of mathematical definition of the quality measures of the arguments to be included in the cost function, we mainly focused on the relevance aspect of the retrieved arguments with the hope of satisfying the other aspects. In other words, the cost function to be minimized in the training phase and then the validation measures we provided for the networks in validation phase are based on the relevance assumption. It is not guaranteed that a relevant argument is necessarily rhetorically well-written or is good in terms of utility.

Manually annotating the retrieved arguments is time consuming and is an expensive procedure. Because of that the nDCG metric has been calculated for the first 5 top-hits of the models. We have calculated our automatic metrics in the validation phase for the top 20 hits. Considering a larger set of retrieved data (i.e. metric @20) may affect the reported performances in the test set.

For the validation phase the models had to rank 105 documents for a given query (claim of the validation argument). In the test phase however, we provided different number of candidate premises to be re-ranked by the network. This might vary from 300 to even 1,200 arguments. Considering the number of documents to be scored in the validation and test phase, it can be realized that the network has a harder to task to do. This might contribute the validation scores not be able to give a realistic picture of the model performances. Increasing the number of premises for the validation data would make the validation and consequently, the training phase more computationally expensive. Decreasing the number of candidate premises provided by B25 would lead the good arguments to be skipped.

Considering the validation scores of the models and the analysis of the diversity of the model outputs for the test phase, it is not surprising that the performance of the recurrent based Siamese network and also the SNRM would not be that satisfying. Note that both of the networks compute the similarity score based on the representation of the input-pairs. For the task of ad-hoc retrieval such representation focused models would not be a good choice as they cannot handle the exact matches and also the interaction of the input pairs is overlooked.

For the SNRM model the representation of the input pairs for creating the inverted index is under question. Maybe the structure of fully connected network for n-grams is not a good choice to grab the contextual information. The regularization term (minimizing the L1 norm) that has been used in the cost function of the model could be also challenged for the purpose of creating representation. On the other hand, in order to accelerate the retrieving process, the authors came up with the idea of sparse representation of the input pairs. In the retrieval phase, we calculate the inner product of the query and the document representations which may not be that informative for the sparse vectors. The calculated scores for the related and the unrelated documents would not differ that much.

Due to the idea of using RBF kernels for modeling input interaction, it could be expected that the kernel based models may outperform DRMM. KNRM holds a novel approach for modeling the interaction of the input pairs compared to DRMM.

For the case of kernel based networks, contextualized embedding illustrates better performance than the Conv-KNRM which highlights the fact that for grabbing the contextual information the state-of-the-art methods outperform the approach of using convolutional windows. A better configuration of the window sizes of the convolutional layers may improve the results of the network.

Outperforming the performance of the models with the contextualized embedding is noticeable also for the case of DRMM. This means that the deeper understanding thanks to the contextualized embedding is really a concept that has to be worked on.

Aggregation model did not improve the performance of the argument retrieval in the test phase. This may relate to the selection of the models to take part in the aggregation scenario. Poor performing networks such as Siamese and DRMM could be excluded for the aggregation. Other alternative aggregation strategies would also contribute an improvement in the results.

KNRM with the performance of 68.4% got the 4th place among the competitors and the only model which had a better performance rather than the baseline model achieved the score of 80.4% for the nDCG@5 score. It seems that it holds an approach of language modeling to solve the problem.

The problem of argument retrieval when it comes to retrieving rhetorically well-written and utility is hot and emerging problem and clearly not an easy task. Lacking of the mathematical explanation of the quality dimension for the retrieved arguments to be included in the validation metrics and also lack soft annotated data (a data which gives relative score for the document and query pairs) makes the challenge difficult.

Chapter 5

Conclusion

In this study we exploited deep neural ranking models to retrieve arguments for the given query. In order to map the problem of argument retrieval to an information retrieval and avoiding creating qrel file for the ranking task, we used a distant supervision technique. We defined the conclusion of the arguments as the queries and the premises as the documents. We defined the task of retrieving relevant arguments as searching for related premises for a given query.

The ranking task in this study is composed of multiple stages: data preprocessing, training, validation, tests and result aggregations.

Data preprocessing In this step, we firstly visualized the existing data. This means that the length of query and premises, the vocabulary sizes were inspected. This helped us to understand which arguments could be filtered out. Thanks to the preprocessing, we could take a reasonable query and document lengths to feed into the models. By inspecting the arguments, we also recognized the type of preprocessing that are required such as digital to word conversion for numbers, how to handle the punctuation and repetitive words.

Holding the distant supervision approach, we assumed the premises of the arguments as the related documents to the queries. We still required a set of unrelated documents in the training phase though. In the final step of preprocessing phase we assigned 1 unrelated premise to each training and 20 unrelated premise to each validation argument. This way we have created a dataset containing binary query relevance information. This information made the use of deep neural ranking model for the ad-hoc retrieval task possible.

Training and validation phase We have exploited different deep neural ranking models to compare their performances in ranking. After a certain number of training batches, we fed the network with the validation dataset and if the result was better than the best one achieved up to now, we replaced the saved model with the model configuration with which we achieved the best validation results.

In this study we have exploited 8 different neural networks. Siamese neural network with recurrent sub-network units, deep relevance matching model (DRMM), kernel based neural ranking model (KNRM) and the CKNRM, vanilla BERT, KNRM with BERT embedding, DRMM with BERT embedding and stand alone neural ranking model(SNRM) are the ones that have been studied here.

Test phase The 50 given test queries by the Touché competition is our test set. We believed that the most relevant queries to the test set are the ones whose premises can get the best ranking score when they are fed to the networks along with the test queries.

All of the networks (except SNRM) require a set of candidate data for ranking the arguments. They do a kind of re-ranking of the candidate arguments suggested by BM25 score. The score was calculated over the normalized conclusions and the input test query (query and conclusion without having any stop word). SNRM retrieved the documents (premises) based on the inverted index that it generated from the sparse representations.

Result aggregation In order to improve the results, we made use of linear regression so that we can aggregate the results. Before that, we required to analyze how diverse the results of the networks from each other were. We recognized the test result of the SNRM as outlier as the retrieved documents were completely different from other ranking results.

Test results The reported mean of nDCG@5 score for the performance of the exploited models over the 50 test queries suggests that the KNRM has the best performance with the score of 68.4%. Using the kernels for representing the interaction of the query and the document contributes the best performance compared to the other models.

Utilizing the contextualized embedding contributed to an improvement in performance in DRMM. For the case of kernel based models, compared to CKNRM, in which the convolutional filters were used for grabbing the context information, contextualized embedding represents a better performance. In general it seems that deeper understanding of the input pairs thanks to the contextualized embedding would lead to a better ranking performance based on relevance criterion.

The task of argument retrieval is a challenging task in terms of retrieving the arguments which satisfy the quality dimensions of the arguments. We have focused more on the relevant arguments, however, this may not represent the most rhetorically well-written ones or the ones which really can build a stance for a user for a given topic. Lack of annotated data including information about the argument quality and a concrete mathematical explanation of the quality dimensions are the greatest challenges in creating the models for the aim of argument retrieval.

Future works Preparing a dataset whose annotation can help the model retrieving arguments which meet different argument quality dimension is a feature work which seems to be of a high priority. Providing a concrete mathematical explanation which can map the argument quality dimension is also important to be included in the cost function and validation metrics.

In this study, we used neural ranking models which typically require candidate documents for ranking. This may restrict the model performance to the algorithm by which the candidate arguments are provided. On the other hand, SNRM provided a diverse results compared to other networks which may be a signal for unreliable results. A future work is to make a specific focus on this end-to-end neural ranking model. A better representation of the documents and query may contribute to a reliable result. In this way, we may modify the structure of the network for document representation or we can also exploit contextualized language models for having a better embedding of the tokens.

For the aggregation task, a more intuitive strategy for selecting the models are required as the result did not improve compared to the individual models. Other than linear regression other ranking aggregation strategies may can contribute to a better result.

Although we have tried several structures of deep neural ranking models to retrieve arguments, it is still under question whether any other network structure exists which can provide us with better results. The other work can concentrate on the aggregation models. Considering that there exist different types of neural ranking models, a good strategy of aggregation may contribute to more promising results.

Instead of generating binary classes of related and unrelated premises to the conclusion, we may, somehow, can annotate the premises to have different levels of similarity and this way nDCG can provide us with more meaningful evaluation measures. Generating a corpus with soft similarity level would be a great step toward developing the system for argument retrieval.

In this work, we focused on the related arguments. However, we have different dimensions for measuring the quality of the arguments. Working on deep networks which can concentrate more on these dimensions may contribute to a retrieval system with more convincing and better quality of arguments.

Appendix A

Sample Retrieved Arguments

In this appendix, we show some sample retrieved arguments by the models for a test queries in the Touché shared task. For the test query: "Should Corporal Punishment Be Used in Schools?", the very first arguments retrieved by models are as the table in A.1. This table gives a sense about how the arguments look like in the corpus and how good the retrieved arguments are. Note that for the long premises, because of lack of space, we have used "..." which means that this is not the complete version of the premise.

Table A.1: Sample retrieved arguments by different models for a test query

Model	Claim	Premise
GRU	corporal punishment should be enforced in schools	Round One: Acceptance Round Two: Arguments Round Three: Rebuttals/New Arguments Round Four: Rebuttals, No Further Arguments
DRMM	capital punishment	In this debate pro will have to prove that the death penalty can be justified. Round one : Accepting Round two : Opening Argument Round three : Open Debate Round four : Open Debate Round five : Closing Arguments
KNRM	corporal punishment should be banned from schools	This article shows the ineffectiveness of corporal punishment in schools. In closing, corporal punishment should be banned because it is ineffective and lowers students' IQ's. Other forms of punishment should be explored and tested. Good luck.
CKNRM	corporal punishment on children by thier parents	Corporal Punishment (Con)Introduction For hundreds of years, parents having been using numerous punishments on their children. In many case, these punishments are either cruel, or inhumane. One of the most common examples that many people think of when they think of cruel punishment in modern society is corporal punishment...
SNRM	age limit to drafting people in the military	DEy shuld never be let out cuz dey are Trapped for life Trapped for life
Vanilla BERT	corporal punishment should be enforced in schools	Unfortunately my opponent has forfeited this round however I still thank him for a fun and interesting debate.
KNRM BERT	corporal punishment should be enforced in schools	Round One: Acceptance Round Two: Arguments Round Three: Rebuttals/New Arguments Round Four: Rebuttals, No Further Arguments
DRMM BERT	should schools use corporal punishment	My argument stands, and screw this ten, zero character limit. really makes me angry. To bad my opposition didn't respond.

Bibliography

J Anthony Blair. *Groundwork in the theory of argumentation: Selected papers of J. Anthony Blair*, volume 21. Springer Science & Business Media, 2011. 2.1.3

Alexander Bondarenko, Matthias Hagen, Martin Potthast, Henning Wachsmuth, Meriem Beloucif, Chris Biemann, Alexander Panchenko, and Benno Stein. Touché: First shared task on argument retrieval. In *European Conference on Information Retrieval*, pages 517–523. Springer, 2020. 1, 2.3

Haolan Chen, Fred X Han, Di Niu, Dong Liu, Kunfeng Lai, Chenglin Wu, and Yu Xu. Mix: Multi-channel information crossing for text matching. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 110–119, 2018. 2.1

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014. 3.4.1

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 3.4.1, 3.4.1, 3.4.1, 3.4.1, 3.4.1, 3.4

Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 126–134, 2018. 2.1, 3.4.3.2, 3.4.3.2, 3.4.3.2, 3.8, 4.2.4

Wayne W Daniel et al. *Applied nonparametric statistics.* Houghton Mifflin, 1978. 4.4

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 3.4.4

Lorik Dumani. Good premises retrieval via a two-stage argument retrieval model. In *Grundlagen von Datenbanken*, pages 3–8, 2019. 2.1.1, 2.1, 2.1.3, 2.2, 4.3

Yixing Fan, Jiafeng Guo, Yanyan Lan, Jun Xu, Chengxiang Zhai, and Xueqi Cheng. Modeling diverse relevance patterns in ad-hoc retrieval. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 375–384, 2018. 2.1

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64, 2016. 2.1, 3.4.2, 3.4.2.1, 3.5, 3.4.2.1

Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. A deep look into neural ranking models for information retrieval. *Information Processing & Management*, page 102067, 2019. 2.2, 2.2.1, 2.2.2, 3.4.5

Ivan Habernal and Iryna Gurevych. Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1589–1599, 2016. 2.1.3

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050, 2014. 2.1

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338, 2013. 2.1

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. Pacrr: A position-aware neural ir model for relevance matching. *arXiv preprint arXiv:1704.03940*, 2017. 2.1

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4): 422–446, 2002. 3.7.3

George A Kennedy. *Aristotle, on Rhetoric: A Theory of Civic Discourse, Translated with Introduction, Notes and Appendices.* Oxford: Oxford University Press, 2007. 2.1.3

Zhengdong Lu and Hang Li. A deep architecture for matching short texts. In *Advances in neural information processing systems*, pages 1367–1375, 2013. 2.1

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. Cedr: Contextualized embeddings for document ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1101–1104, 2019. 3.4.4, 3.4.4.1, 3.4.4.2, 4.2.6, 4.2.7

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1291–1299, 2017. 2.1

Yifan Nie, Yanling Li, and Jian-Yun Nie. Empirical study of multi-level convolution models for ir based on representations and interactions. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 59–66, 2018a. 2.1

Yifan Nie, Alessandro Sordoni, and Jian-Yun Nie. Multi-level abstraction convolutional model with weak supervision for information retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 985–988, 2018b. 2.1

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. Text matching as image recognition. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 2.1

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. Deeprank: A new deep architecture for relevance ranking in information retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 257–266, 2017. 2.1

Joaquín Pérez-Iglesias, José R Pérez-Agüera, Víctor Fresno, and Yuval Z Feinstein. Integrating the probabilistic models bm25/bm25f into lucene. *arXiv preprint arXiv:0911.5046*, 2009. 3.5

Xipeng Qiu and Xuanjing Huang. Convolutional neural tensor network architecture for community-based question answering. In *Twenty-Fourth international joint conference on artificial intelligence*, 2015. 2.1

Dragomir R Radev, Hong Qi, Harris Wu, and Weiguo Fan. Evaluating web-based question answering systems. In *LREC*, 2002. 3.7.1

Jinfeng Rao, Wei Yang, Yuhao Zhang, Ferhan Ture, and Jimmy Lin. Multi-perspective relevance matching with hierarchical convnets for social media search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 232–240, 2019. 2.1

Richard D Rieke, Malcolm Osgood Sillars, and Tarla Rai Peterson. *Argumentation and critical decision making*. Longman New York, 1997. 2.1.1

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 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*, pages 101–110, 2014. 2.1

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 4.2.1

Christian Stab, Johannes Daxenberger, Chris Stahlhut, Tristan Miller, Benjamin Schiller, Christopher Tauchmann, Steffen Eger, and Iryna Gurevych. Argumentext: Searching for arguments in heterogeneous sources. In *Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: demonstrations*, pages 21–25, 2018. 2.1.3

Andrew Turpin and Falk Scholer. User performance versus precision measures for simple search tasks. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 11–18, 2006. 3.7.2

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 3.4.4, 3.9

Henning Wachsmuth, Nona Naderi, Yufang Hou, Yonatan Bilu, Vinodkumar Prabhakaran, Tim Alberdingk Thijm, Graeme Hirst, and Benno Stein. Computational argumentation quality assessment in natural language. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 176–187, 2017a. 1, 2.1.3, 1

Henning Wachsmuth, Martin Potthast, Khalid Al Khatib, Yamen Ajjour, Jana Puschmann, Jiani Qu, Jonas Dorsch, Viorel Morari, Janek Bevendorff, and

Benno Stein. Building an argument search engine for the web. In *Proceedings of the 4th Workshop on Argument Mining*, pages 49–59, 2017b. 1, 2.1.2, 2.1.3

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. A deep architecture for semantic matching with multiple positional sentence representations. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016a. 2.1

Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. Match-srnn: Modeling the recursive matching structure with spatial rnn. *arXiv preprint arXiv:1604.04378*, 2016b. 2.1

Bingning Wang, Kang Liu, and Jun Zhao. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1288–1297, 2016. 2.1

Shuohang Wang and Jing Jiang. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*, 2016. 2.1

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval*, pages 55–64, 2017. 2.1, 3.4.1, 3.4.3.1, 3.6, 3.4.3.1, 3.4.3.1, 3.4.3.1, 3.4.3.1, 3.4.3.1, 3.4.3.1, 3.7, 4.2.3

Wei Yang, Haotian Zhang, and Jimmy Lin. Simple applications of bert for ad hoc document retrieval. *arXiv preprint arXiv:1903.10972*, 2019. 2.1

Wenpeng Yin and Hinrich Schütze. Multigrancnn: An architecture for general matching of text chunks on multiple levels of granularity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 63–73, 2015. 2.1

Hamed Zamani, Mostafa Dehghani, W Bruce Croft, Erik Learned-Miller, and Jaap Kamps. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 497–506, 2018. 3.4.5, 3.4.5.1, 3.4.5.1, 3.10, 3.4.5.2, 3.11, 3.4.5.2, 3.4.5.2, 3.4.5.3, 3.12