DEPARTMENT OF INFORMATICS
PROGRAM OF POSTGRADUATE STUDIES IN COMPUTER SCIENCE

M.Sc. Thesis

"Sentence Selection from Biomedical Documents for Question Answering"

Polyvios Liosis

EY1612

Supervisors:
Ion Androutsopoulos
Ryan McDonald

ATHENS, JUNE 2018

Acknowledgements

I would like to express my sincere gratitude to my supervisors Ion Androutsopoulos and Ryan McDonald for their consistent and restless guidance throughout this thesis and for providing me the chance to actively get involved in a very interesting NLP task. I would also like to thank George Brokos and Dimitris Pappas. Finally, a big thanks goes to my family for their continuous support.

Abstract

Sentence selection is a challenging task that could play a significant role in the next generation of advanced web search and Question Answering (QA) systems. Also known as snippet selection, it is the task of identifying sentences of a specific document that contain the answer to a given query. The developers of modern question answering systems are trying to implement this crucial functionality. For example, Google's advanced QA system, not only returns the top relevant documents for a given query but also snippets of these documents that answer it with a great probability. Imagine an advanced QA system that returns documents of many pages and also highlights sentences that answer with success the given question. The result of this would be the fact that the users would retrieve the desired information faster. From the above, the importance of the implementation of a sentence selection functionality for a QA system is quite obvious.

The main objective of this thesis is to investigate the fascinating task of sentence selection, a demanding task that belongs to the broader domains of natural language processing and information retrieval, using advanced deep learning methods. We selected three state-of-the-art neural network models and we evaluated them on well-known datasets that are used for various QA tasks. Then we applied these models on data of the biomedical domain with the purpose to create a snippet retrieval system with which we could participate in 6^{th} BioASQ challenge. Our final results show that deep learning architectures for sentence selection are competitive and outperform traditional information retrieval systems.

Περίληψη

Η επιλογή προτάσεων είναι μια συναρπαστική και δύσκολη εργασία που θα μπορούσε να διαδραματίσει σημαντικό ρόλο στην επόμενη γενιά προηγμένων συστημάτων αναζήτησης στο διαδίκτυο και στα συστήματα ερωτοαπαντήσεων. Επίσης γνωστή και ως επιλογή αποσπάσματος, είναι η εργασία της επισήμανσης προτάσεων ενός συγκεκριμένου κειμένου οι οποίες περιέχουν την απάντηση σε ένα δεδομένο ερώτημα. Οι δημιουργοί των σύγχρονων συστημάτων ερωτοαπαντήσεων προσπαθούν να εφαρμόσουν αυτήν την σημαντική και ζωτικής σημασίας λειτουργικότητα. Για παράδειγμα, το προηγμένο σύστημα αναζήτησης της Google όχι μόνο επιστρέφει τα κορυφαία σχετικά κείμενα για ένα δεδομένο ερώτημα αλλά και αποσπάσματα αυτών των κειμένων που απαντούν το ερώτημα με μεγάλη πιθανότητα. Φανταστείτε ένα προηγμένο σύστημα ερωτοαπαντήσεων που επιστρέφει έγγραφα πολλών σελίδων σημειώνοντας παράλληλα τις προτάσεις που απαντούν με επιτυχία την δεδομένη ερώτηση. Το αποτέλεσμα θα είναι ότι οι χρήστες θα μπορούν να ανακτήσουν ταχύτερα τις επιθυμητές πληροφορίες που αναζητούν. Από τα παραπάνω, η σημασία ανάπτηξης της λειτουργικότητας επιλογής προτάσεων σε ένα σύστημα ερωτοαπαντήσεων είναι προφανής.

Αυτή η διπλωματική εργασία έχει ως κύριο στόχο να διερευνήσει το συναρπαστικό έργο της επιλογής προτάσεων, μία απαιτητική διαδικασία που ανήκει στους ευρύτερους τομείς της επεξεργασίας φυσικής γλώσσας και της ανάκτησης πληροφοριών, χρησιμοποιώντας προηγμένες μεθόδους βαθιάς μάθησης. Επιλέξαμε τρία μοντέλα νευρωνικών δικτύων τελευταίας τεχνολογίας και τα αξιολογήσαμε σε πολύ γνωστά και αναγνωρισμένα σύνολα δεδομένων που χρησιμοποιούνται για διάφορες εργασίες ερωτοαπαντήσεων. Στη συνέχεια, εφαρμόσαμε αυτά τα μοντέλα σε δεδομένα που ανήκουν στον τομέα της βιοϊατρικής με σκοπό να δημιουργήσουμε ένα σύστημα ανάκτησης αποσπασμάτων με το οποίο θα μπορούσαμε να συμμετάσχουμε στον 6ο διαγωνισμό του BioASQ. Τα τελικά μας αποτελέσματα δείχνουν ότι οι αρχιτεκτονικές βαθιάς μάθησης για την επιλογή των προτάσεων είναι ανταγωνιστικές και μπορούν να ξεπεράσουν σε μεγάλο βαθμό τα παραδοσιακά συστήματα ανάκτησης πληροφοριών.

Contents

Chapter 1

Introduction

1.1 Sentence selection for Question Answering

Sentence selection is a challenging task and could play a significant role in the next generation of advanced web search and Question Answering (QA) systems. Sentence selection is the task of identifying sentences of a specific document that contain the answer to a given question. More specifically, given one or more documents, the system tries to answer a specific question by selecting sentences from each document that contain the desired information, similar to how humans look for specific information in documents or books. This is an important problem in its own right as well as in the larger context of open domain question answering [**1**].

An example we BioASQ [**2**], a biomedical question answering challenge that will be discussed below, is the following:

Question: Is Achondroplasia associated with hearing loss?

Correct Answer: We conclude that verbal comprehension is significantly impaired in children with achondroplasia.

Correct Answer: The AA report a clinical and radiological study performed in 18 achondroplastic patients in order to achieve a nosological settlement of the otological impairments.

Wrong Answer: Three patients had significant sensorineural hearing loss, two had conductive hearing loss and one patient had combined hearing loss.

Wrong Answer: Adults were more likely to fail hearing screening than children.

In the above example we can see a major challenge of the sentence selection task, which is the fact that the correct answer might not directly share many words with the question. Instead, the answers may only be semantically related and sometimes contain a significant amount of noise and unrelated information. Furthermore, there always exists the possibility of an irrelevant sentence sharing many words with the question [**3**].

In the last few years, due to the increase in computing power and the development of state-of-the-art open-source libraries, the popularity of neural networks has significantly increased. Subsequently, a variety of Deep Learning methods and model designs have blossomed in the context of Natural Language Processing [**4**; **5**]. Deep learning has been applied with success to various NLP tasks, like named entity recognition [**6**], text classification [**7**; **8**] and caption generation [**9**]. The task of sentence selection is not an exception to the above trend. Many scientists are trying to create their own Deep Learning models or improve existing ones, in order to face successfully the great challenge of sentence selection for question answering.

1.2 Goal of the Thesis

The main objective of this thesis is to investigate the challenging task of sentence selection for question answering, using deep learning methods. As we mentioned above, in the case of the snippet selection task, a candidate answer (which in our case is actually just a sentence as it is extracted from a sentence splitter) might not directly share lexical units with the given question, but it may actually contain synonyms of the lexical units. As a result, simple information retrieval (IR) techniques that use word matching cannot tackle the problems of synonymity and polysemy. On the other hand, word embeddings (dense vectors) produced by neural networks [**10**] can solve the above issue, as words with similar meaning have close representations. Deep Learning architectures can play a significant role here due to their ability to operate on top of the word embeddings of a question and a candidate answer. Consequently, useful patterns can be extracted that can be used in order to face the problems for which simple techniques are unable to find a solution.

The first goal was to select a number of different state-of-the-art neural network models, evaluate them on different datasets and improve them if possible. More specifically, we decided to use the following state-of-the-art deep learning models: the model of Severyn and Moschitti [**11**] and the ABCNN model of Yin et al. [**12**]. We should clarify that the second model is actually a set of models from which we chose two. Consequently, we end up with three innovative deep learning architectures. The datasets we used for the evaluation of the above models are TREC QA [**13**] and WikiQA [**14**].

The second goal was to apply these neural network models on data of the biomedical domain used in previous BioASQ competitions. More specifically, we focused on the dataset of the 5^{th} year of the BioASQ competition which contains a considerable number of questions and relevant documents. The final and most challenging goal was to participate, for the first time as Athens University of Economic and Business, in BioASQ's 6^{th} year competition (Task 6b-Phase A)[1] [2].

1.3 MEDLINE

MEDLINE[3] is a bibliographic database of life sciences and biomedical information. Started in the 1960s, it now provides more than 28 million references to biomedical and life sciences journal articles. In general it includes bibliographic information for articles from academic journals covering medicine, pharmacy and many other fields.

Compiled by the United States National Library of medicine (NLM), MEDLINE is freely available on the Internet and searchable via the PubMed search engine and NLM's National Center for Biotechnology Information's Entrez system.

1.4 PubMed Search Engine

PubMed is a free search engine accessing primarily the MEDLINE database of references and abstracts on life sciences and biomedical topics, developed by the United States National Institutes of the health (NIH) of the National Library of Medicine.

[1] For more information on Task6b-Phase A of 6^{th} year of BioASQ competition, see http://participants-area.bioasq.org/general_information/Task6b

[2] The entire code that was developed during this thesis is available at: https://github.com/polLiosis/Sentence_Selection_for_QA

[3] More information about MEDLINE: https://www.nlm.nih.gov/bsd/medline.html

NLM maintains the database as part of the Entrez system of information retrieval. PubMed was first released in January 1996 and was offered free to public in June 1997.

1.5 The BioASQ Challenge

BioASQ is a biomedical semantic indexing and QA challenge, which started as a project funded by the European Union (2012-14)[4], and is now supported by NLM[5]. BioASQ initiates a series of challenges on biomedical semantic indexing and question answering, with the purpose of helping those who are in need of knowledge scattered across a great number of biomedical documents.

The challenges include various tasks relevant to hierarchical text classification, machine learning, information retrieval, QA etc **[2]**. The BioASQ challenge comprises the following three tasks:

- *Large-Scale Online Biomedical Semantic Indexing:* The participants are asked to classify new PubMed documents, before PubMed curators annotate them manually.
- *Biomedical Semantic QA:* The participants are given English questions constructed by biomedical experts and they have to retrieve relevant documents, extract relevant snippets etc.
- *Funding Information Extraction from Biomedical Literature:* The participants are asked to extract grant ids and grant agencies from the full text of PubMed documents.

1.6 Overview of the Thesis

The rest of this thesis is organized as follows:

- *Chapter 2*: In this chapter we describe the architecture of the deep learning models we used for our experiments. These models are state-of-the-art deep learning models for Question Answering and their implementation has been made publicly available by the authors of the respective scientific papers. We also describe the BM25 ranking function which we use as a baseline and also as an additional feature in order to improve the performance of the deep learning models.
- *Chapter 3*: In this chapter we present crucial information and statistics of the three datasets we use for our experiments.
- *Chapter 4*: Here we present the basic evaluation measures we took into consideration and figures which illustrate the results of our experiments.
- *Chapter 5*: This chapter discusses related work.
- *Chapter 6*: In this final chapter we draw some useful conclusions and propose possible future directions.

[4]More information available at: https://cordis.europa.eu/project/rcn/105774_en.html
[5]Official website of NLM: https://www.nlm.nih.gov

Chapter 2

Deep Learning Methods and the BM25 Baseline

2.1 The Method of Severyn and Moschitti

This deep learning model which was proposed by Severyn and Moschitti [**11**], is based on a convolutional neural network architecture and was designed with the purpose of re-ranking pairs of short text. We learn the optimal representation of each text pair and a similarity function to relate them in a supervised way.

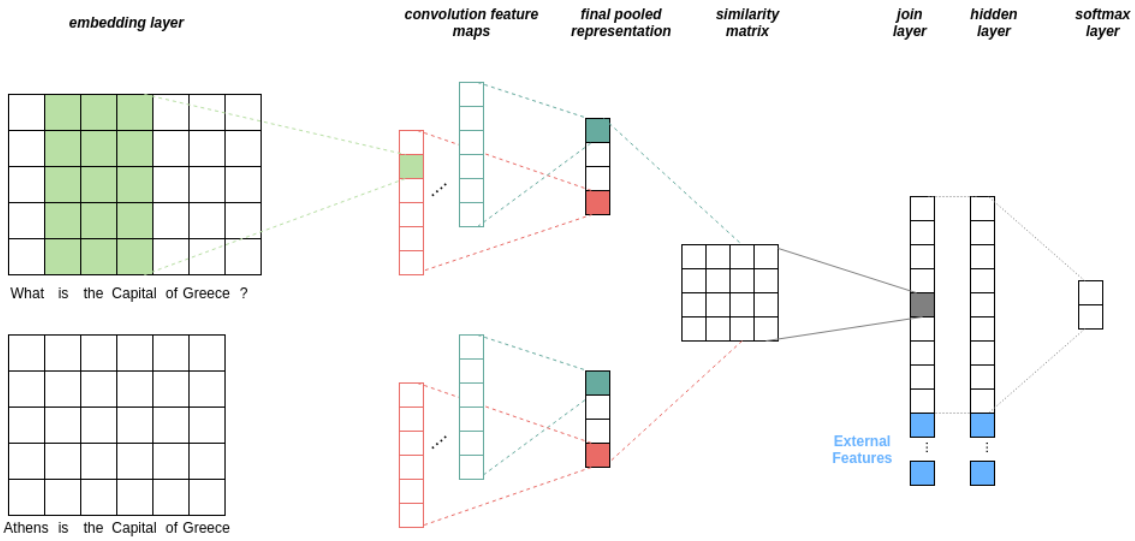

FIGURE (2.1) The method of Severyn and Moschitti [**11**].

2.1.1 Flow of Information

The information flow is quiet obvious from the basic diagram of the model's architecture (Figure 2.1). The model receives as input two sentences, one question and one candidate answer, and our main objective is to classify the candidate answer (1 if it is a correct answer and 0 if it is wrong).

The first step is to convert each sentence into a sentence matrix by retrieving the embedding vector of each word from our pre-trained word embeddings. Then we apply convolutions to each sentence matrix followed by a pooling layer in order to obtain the final representation of each sentence.

The second step is to obtain the similarity score between the two representations and concatenate it with the feature maps of the convolutions and an additional feature vector, which contains additional external information about our two sentences.

The final step is to feed the final question-answer representation into a softmax layer in order to obtain the probability distribution over the two classes.

2.1.2 Sentence Model

As we mentioned above the main purpose of the sentence model is to learn good representations of the queries and documents, which are then used for computing their semantic matching. The sentence model consists of the first three blocks of the basic architecture presented in figure 2.1, which are the embedding layer (contains the embedding matrix of each sentence), the convolutional feature maps and the pooling layer (final representation of each sentence).

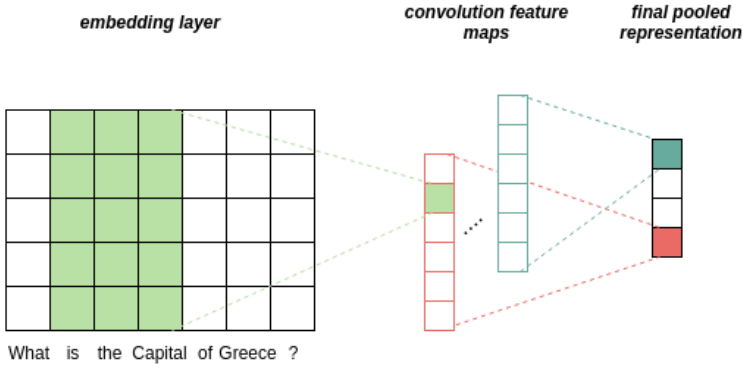

FIGURE (2.2) The sentence model of the architecture presented in Figure 2.1, the purpose of which is to extract the feature representation for a specific sentence.

A brief explanation of the main components of the sentence model follows:

- *Embedding layer*: As we can see in Figures 2.1 and 2.2, the sentence model receives as input a sentence of length n (the question or the candidate answer), which is tokenized into a sequence of words: $[w_1, w_2, ..., w_n]$. Then for each word we find the respective vector $W \in \mathbb{R}^d$ from our pre-trained word embeddings, where d is their dimensionality. In the end we form the sentence matrix $S \in \mathbb{R}^{d \times |s|}$ (where d is the dimensionality of the pre-trained word embeddings and $|s|$ is the length of the sentence) by concatenating all the retrieved word embeddings.

- *Convolutional layer*: The main purpose of convolutional layers is to extract patterns. In this specific model a wide convolution is used creating vectors of size $c \in \mathbb{R}^{|s|+m-1}$, where m is the length of the filter. Each component c_i is the result of an element-wise product computation between a column slice of the sentence matrix S and the filter matrix F, which is then flattened and summed producing a single value. The formula of creating a component c_i is:

$$c_i = (s \star f)_i = s^T_{(i-m+1:i)} f = \sum_{k=i}^{i+m-1} s_k f_k \tag{2.1}$$

where s, f, m and $\star$ are the sentence matrix, the filter, the size of the filter and the convolution operation respectively.

- *Pooling layer*: The feature maps that were produced in the previous step (multiple filters are applied, thus multiple feature maps are produced) are now forwarded to a pooling layer, whose goal is to aggregate the information and produce the final representation of a sentence. There are many types of pooling, but in this case we use max pooling. More specifically, we read each feature map and we pick only the maximum value. As a result, in the end we just unify all the maximum values in order to produce the final representation vector.

2.1.3 Matching Text Pairs

The next part of the architecture is the matching between the two sentences. After the calculation of the final representations of the two sentences from the sentence model, a similarity function is used in order to calculate the similarity score between the question and the candidate answer. Then we concatenate this score together with the representations and various features in order to produce the probabilities.

The matching model consists of the following components:

- *Similarity Matrix (M)*: It is just a parameter and is optimized during the training. We use is to compute the similarity score as follows:

$$sim(x_q, x_d) = x_q^T M x_d \tag{2.2}$$

where $M \in \mathbb{R}^{d \times d}$, x_q is the representation of the question and x_d is the representation of the candidate answer.

- *Additional features*: A vector which consists of additional important information we can obtain during the parsing of the dataset. In particular, this information can be word overlaps of stopwords and non stopwords, similarity scores we obtain using other functions, and in general extra information that can improve the functionality of the model.

- *Join layer*: In this layer we concatenate the similarity score with the two final representation feature maps of the sentences and the external features vector.

- *Hidden layer*: The model contains only one hidden layer right before the softmax layer. The input of this layer is the concatenation of the question representation, the representation of the candidate answer, the similarity score and the additional features vector (output of the previous step which is the join layer).

- *Softmax Layer*: It computes the probability distribution over the two classes.

2.2 ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs

ABCNN **[12]**, which stands for Attention-Based Convolutional Neural Network, is a model which can achieve state-of-the-art performance on various tasks like paraphrase identification, answer selection etc. and it consists of various parts and mechanisms. More specifically, we have the basic convolutional neural network model

and two different kinds of attention that can be applied before the convolution and pooling phase. As a result, we end up with the following four different model architectures:

- *BCNN*: The basic Convolutional Neural Network, the architecture of which can be seen in Figure 2.3.
- *ABCNN1*: A model which uses BCNN as a core and then it applies an attention mechanism on the input of the convolution layer.
- *ABCNN2*: A model which uses BCNN as a core and then it applies an attention mechanism on the output of the convolution layer.
- *ABCNN3*: A model with BCNN as its main core, in which two different attention mechanisms are applied, one on the input and one on the output of the convolution layer.

In this thesis, we focus on the BCNN and ABCNN3 models. We chose ABCNN3 over ABCNN1 and ABCNN2, because the authors of the specific paper suggest that the ABCNN3 model achieves better results for QA tasks.

2.2.1 Flow of Information

The model receives as input two sentences, one question and one candidate answer, and our main objective is to classify the candidate answer (1 if it is a correct answer and 0 if it is wrong).

The first step is to convert each sentence into a matrix by retrieving the embedding vector of each word from our pre-trained word embeddings. Then we feed each sentence matrix into a convolution layer in order to obtain a new feature map representation which can be fed into another convolution layer and so on. This way we can stack an arbitrary number of convolution-pooling blocks in order to extract increasingly abstract features each time.

After the extraction of the final representation feature maps of the two sentences, we feed them into a logistic regression layer together with various helpful features, with the purpose of predicting the probability distribution of the two classes.

2.2.2 Basic Convolutional Neural Network

In this subsection, we describe the convolutional neural network (CNN) which is used as a base for the other three models. The BCNN architecture is illustrated in Figure 2.3.

The following layers exist in the BCNN architecture:

- *Input Layer*: This layer receives as input two sentences of length $|s0|$ and $|s1|$ (the query and the extracted snippet) which are tokenized into sequences of words. Then for each word we find the respective vector from our pre-trained word embeddings. In the end we apply zero padding and we form two matrices, of equal size $\mathbb{R}^{d \times max(|s0|,|s1|)}$ by concatenating all the retrieved word embeddings for each sentence. In Figure 2.3 we have $|s0| = 4$ and $|s1| = 6$ and the padding is not shown for simplicity.
- *Convolution Layer*: The main purpose of this layer is to extract patterns, by using a wide convolution with filter of width w. It is vital to mention the fact that for filter width w, a wide convolution layer transforms an input feature map of s columns into a new feature map of $(s + w - 1)$ columns.

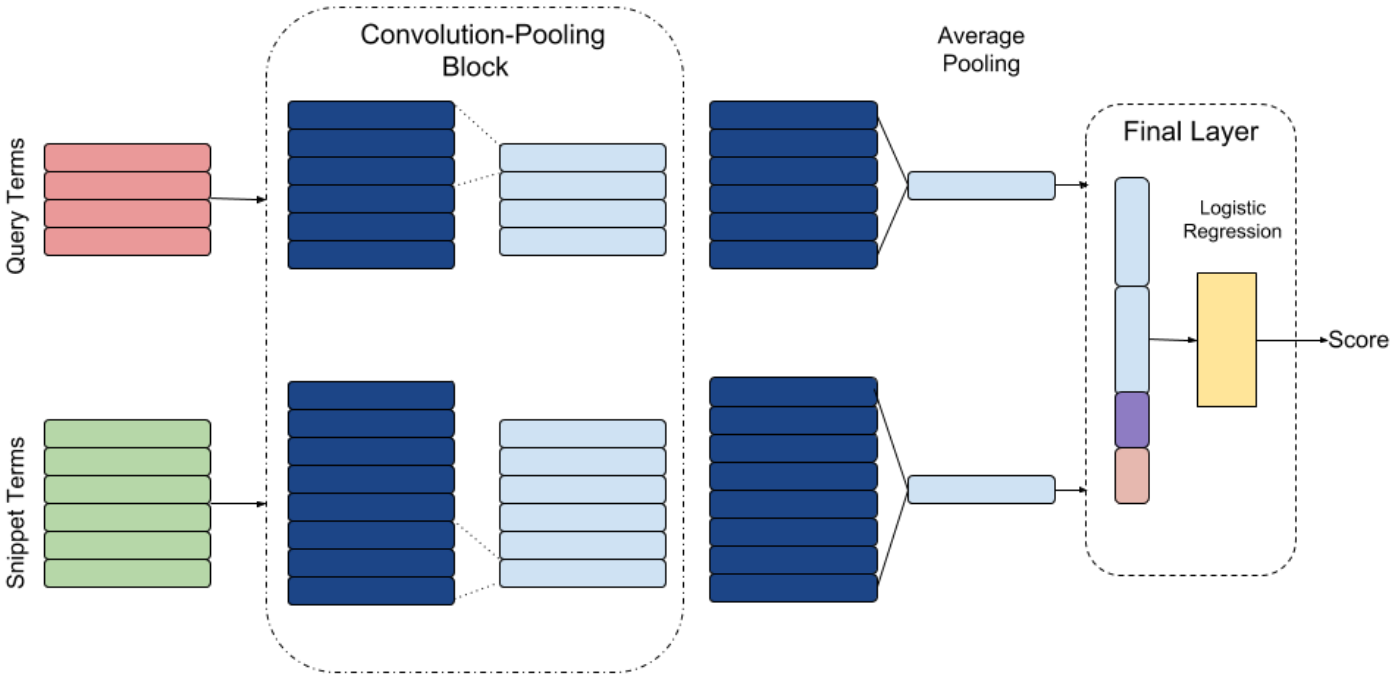

FIGURE (2.3) The architecture of the Basic Convolutional Neural Network model (BCNN). Various attention mechanisms can subsequently be applied to this model to improve its performance.

- *Pooling Layer*: The main purpose of this layer is to transform the output of the convolution layer back to a feature map of length s. As a result, we can stack an arbitrary number of convolution-pooling blocks in order to extract increasingly abstract features. In this layer two kinds of average pooling are applied, average pooling over all columns (all-ap) and average polling over a window of columns (w-ap). With the use of all-ap we obtain a single feature map for the question and candidate answer for each convolution-pooling block. Then cosine similarity scores are computed at each level and are fed into the output layer.

- *Output Layer*: This layer is a logistic regression classifier, and is used in order to produce the probabilities of each class. Here we can also use various additional features, which include possible matching scores, the length of each sentence, number of stopwords, bigram overlaps etc. The default additional features of the BCNN model are the length of the question, the length of the candidate answer and the common words between these two sentences.

2.2.3 Attention Mechanisms

As we have already mentioned, there are two different types of attention mechanisms, each used on different layers of the ABCNN architecture.

The first attention type is used before the convolutional layer's operation and its aim is to improve the features computed by the convolution. The vital role is played by an attention matrix A, which is generated by comparing units of the left representation (question) with units of the right representation (candidate answer). The formula to calculate each cell of the attention matrix A, is the following:

$$A_{i,j} = score(F_{0,r}[:, i], F_{1,r}[:, j]) \tag{2.3}$$

where $F_{0,r}$ stands for the representation of the left sentence (question) and $F_{1,r}$ stands for the representation of the right sentence (candidate answer). The score function can be defined in a variety of ways, but the authors suggest to use the formula:

$$score(x, y) = \frac{1}{1 + d(x, y)} \tag{2.4}$$

where $d(x, y)$ is the Euclidean distance.The next step is to create a new feature map (attention feature map), using matrix A. We generate the attention feature map of each sentence as follows:

$$F_{0,a} = W_0 \cdot A^T \tag{2.5}$$

$$F_{1,a} = W_1 \cdot A \tag{2.6}$$

where $F_{0,a}$ stands for the attention feature map of the left representation, $F_{1,a}$ stands for the attention feature map of the right representation and W_0, W_1 are parameters. Finally, we stack the representation and attention feature map of each sentence as an order 3 tensor and feed it into convolution to generate a higher level representation feature map (one for each sentence). The whole process can be seen in Figure 2.4.

The second type of attention is used on the output of the convolution and computes an attention matrix A on this output in order to re-weight it.

After computing A, we sum all attention values for a unit to derive a attention weight for it. This corresponds to summing all values in a row or column for left or right representation feature map respectively. So the j-th column of the final feature map which is produced by the attention-based windowed average pooling operation, can be computed by the following formula:

$$F^p_{i,r}[:, j] = \sum_{k=j:j+w} a_{i,k} F^c_{i,r}[:, k] \tag{2.7}$$

where $a_{i,k}$ is the attention weight (sum of the corresponding column or row of A) of unit k in representation i (where i is 0 or 1) and $F^c_{i,r}$ is the output feature map of the convolutional layer.

Based on the above analysis, it is obvious that we can combine the two attention mechanisms with the purpose of creating a more rich and improved model, which is ABCNN3, as we have already mentioned. The final architecture of the new Attention-Based Convolutional Neural Network is illustrated in Figure 2.6.

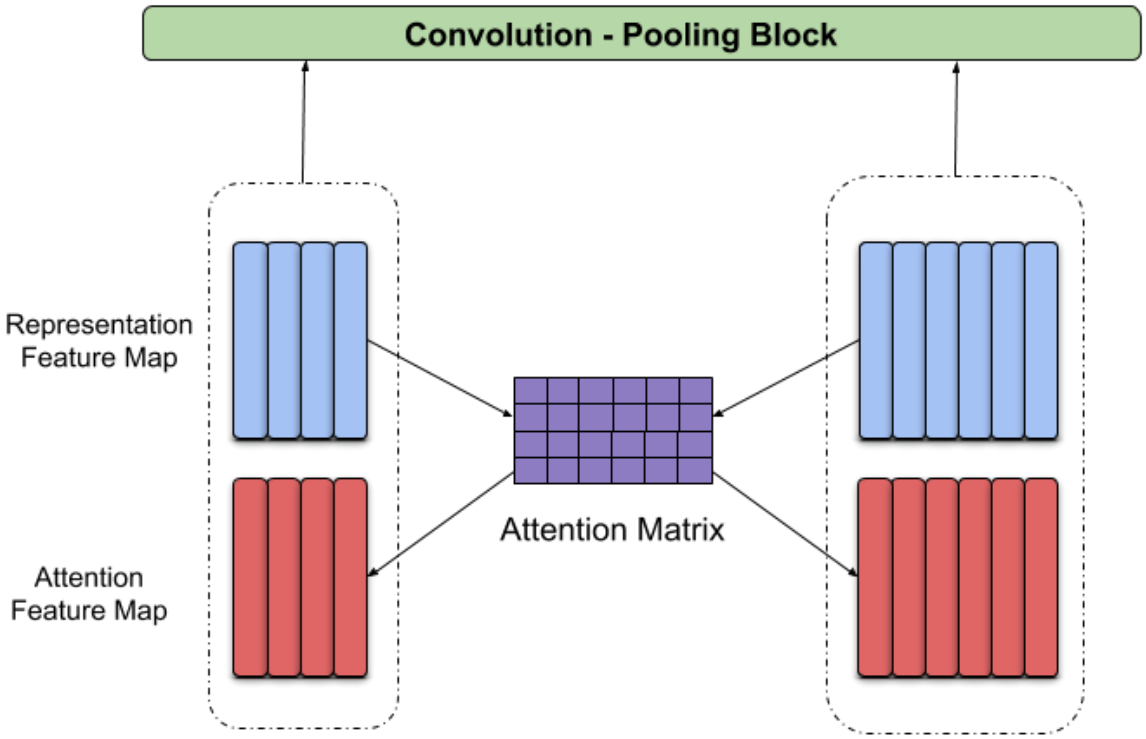

FIGURE (2.4) The first attention mechanism provides improved feature maps to the convolution layer. With the help of an attention matrix two new attention feature maps are calculated for each representation, which are then passed together with the starting representation feature map to the next convolution layer.

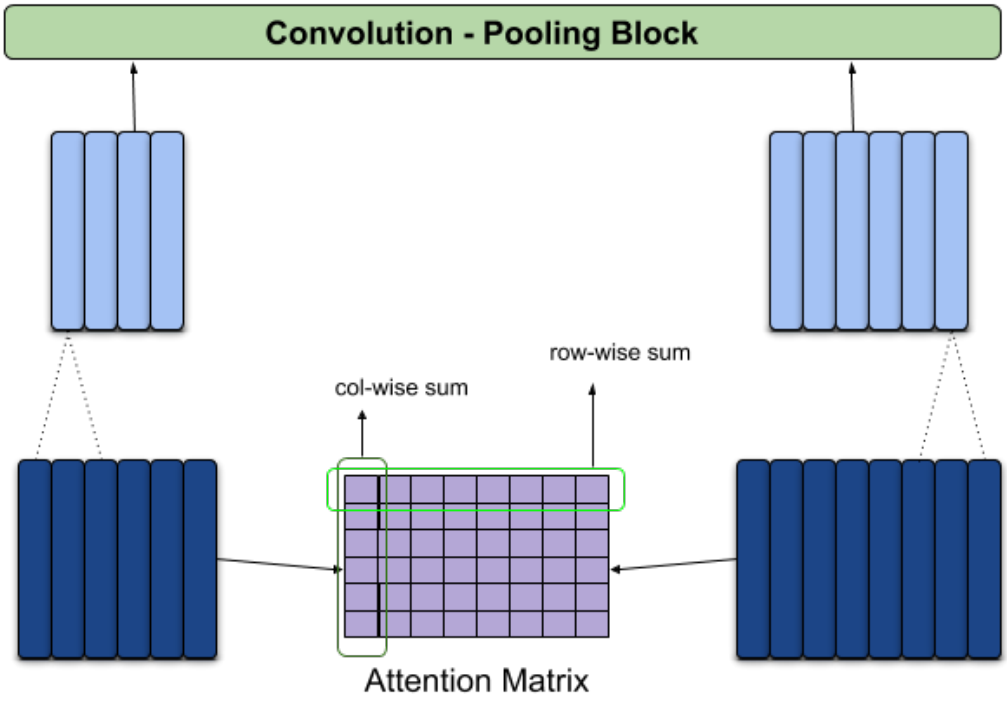

FIGURE (2.5) The second attention mechanism re-weights the output of the convolution. With the help of the attention matrix, an attention-based average pooling is used in order to obtain a new feature map, on which we can apply again the first attention mechanism in order to feed it into a new convolutional layer.

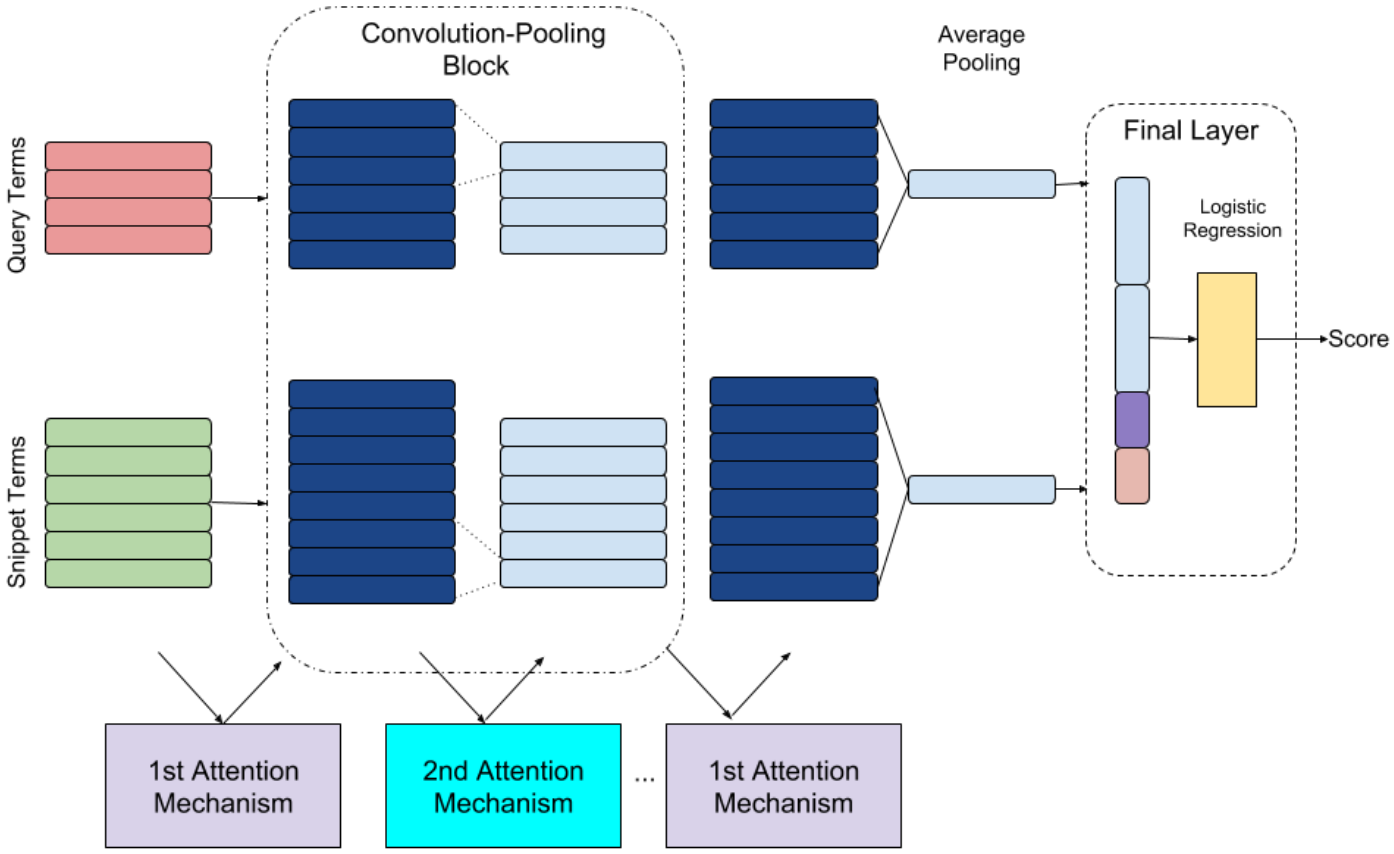

FIGURE (2.6) ABCNN3 combines the two attention mechanisms in order to improve the performance of the BCNN model. With the help of the first attention mechanism it creates better representations as input to the convolutional layer, while with the help of the second one the output of each convolutional layer is re-weighted for an improved performance.

2.3 Okapi BM25

Okapi BM25 is a ranking function used by search engines to rank documents according to their relevance to a given search query and it is based on the probabilistic retrieval framework [**15**; **16**].

More specifically, it is a bag-of-words retrieval function that ranks documents based on the query terms appearing in each document. As a result, documents which contain a lot of words that appear also in the given query, are more likely to get a high BM25 score than documents with fewer words similar to the query. A more formal definition of the BM25 ranking function is the following one:

Given a query Q with n words $q_1, q_2, ..., q_n$ the BM25 score of a document D is:

$$score(D,Q) = \sum_{i=1}^{n} idf(q_i) \frac{tf(q_i,D)(k+1)}{tf(q_i,D) + k(1 - b + b\frac{|D|}{avgdl})} \tag{2.8}$$

where $tf(q_i, D)$ is the term frequency of query term q_i in document D, $idf(q_i)$ is the inverse document frequency of the same term, $|D|$ is the length of the document D in words, $avgdl$ is the average document length in the collection of documents from which we retrieved the document D. Finally, k and b are constants. The idf score of a term q_i, can be computed as follows:

$$idf(q_i) = log\frac{N}{df(q_i) + a} \tag{2.9}$$

where N is the total number of documents in our collection, $df(q_i)$ is the document frequency of the word q_i and a is a smoothing factor.

We have to mention that there also many modified versions of the BM25 ranking function (BM11, BM15, BM25F, BM25+), but for the experiments we use the standard version of BM25. To use BM25 for sentence selection, we treat each extracted snippet as a document and we use the above formula in order to compute each snippet's BM25 score given a query. During our experiments we use BM25 as a stand alone ranking system and we also include it as an external feature to our neural network models.

Chapter 3

Datasets

3.1 TREC QA

This dataset is one of the most widely used benchmarks for question answering tasks [**17**]. It was first used by Wang et al. [**13**]. Since then, it became the standard benchmark for the answer selection task. The data used in the TREC tracks, are available on the TREC website[1].

For our experiments we used the additional training set (TRAIN-ALL) which contains a greater number of QA pairs. Statistics of the TREC QA dataset are shown in the following table and figures:

	QA pairs	**Questions**	**Positives**	**Negatives**
Train	53.417	1.227	6.403	47.014
Validation	1.148	81	222	926
Test	1.515	94	284	1.231

TABLE (3.1) Statistics for the train, validation and test set of TREC QA dataset. The first two columns refer to the number of QA pairs and unique questions in each set respectively, while the last two contain the number of positive and negative answers for each set.

[1] All TREC QA datasets are available on the TREC website: https://trec.nist.gov/data/qa.html

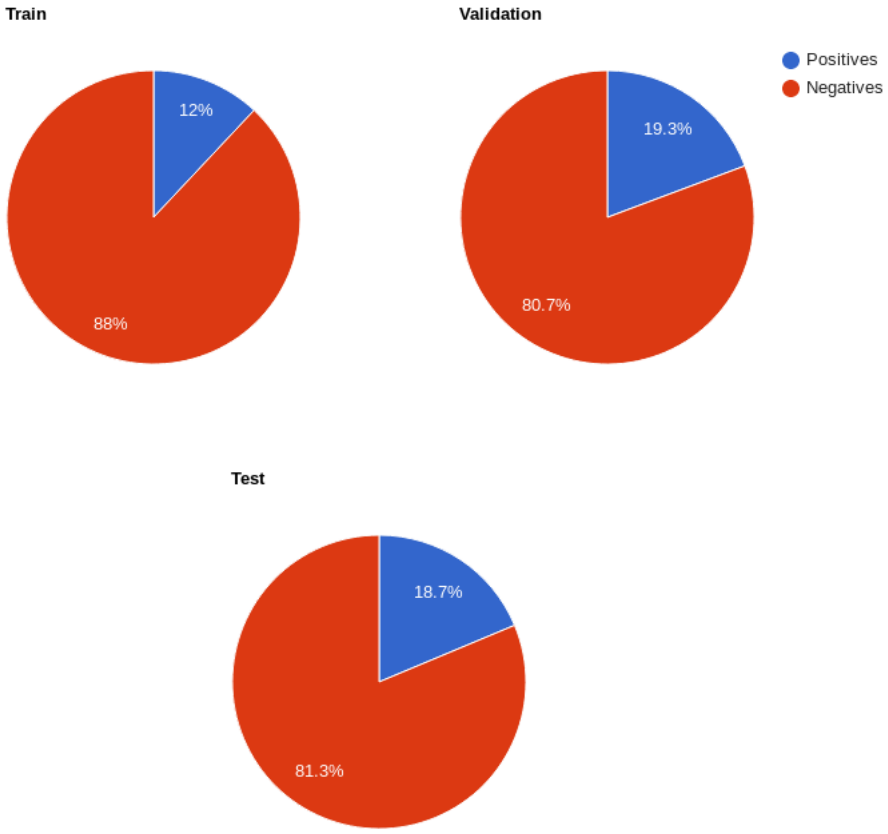

FIGURE (3.1) Percentage of relevant (positive) and irrelevant (negative) candidate answers in each set. The TREC QA dataset is imbalanced, since the negative class outnumbers the positive one.

3.2 Wiki QA

WikiQA[2] **[14]** is a publicly available set of question-answer sentence pairs, collected and annotated for research on open-domain question answering. It is worth mentioning that this dataset includes a great number of questions for which we do not have correct sentences. That means that a specific question could have many irrelevant candidate answers and zero relevant ones. In Table 3.2 and Figure 3.2 we can observe the statistics of WikiQA dataset.

Summing up the information we can extract, it is obvious that the WikiQA dataset is imbalanced since the number of negative QA pairs outnumbers by far the number of the positive ones. Additionally, the train set is the only set that contains questions with not even one relevant answer.

[2]The data and evaluation script can be downloaded at `http://aka.ms/WikiQA`.

	QA pairs	Questions	Positives	Negatives	Without Answer
Train	20.360	2.118	1.040	19.320	1.245
Validation	8.672	873	1.040	7.632	0
Test	2.351	243	293	2.058	0

TABLE (3.2) Statistics for WikiQA dataset. Worth mentioning is the fact that from the 2.118 available questions in the training set, 1.245 do not have a relevant candidate answer.

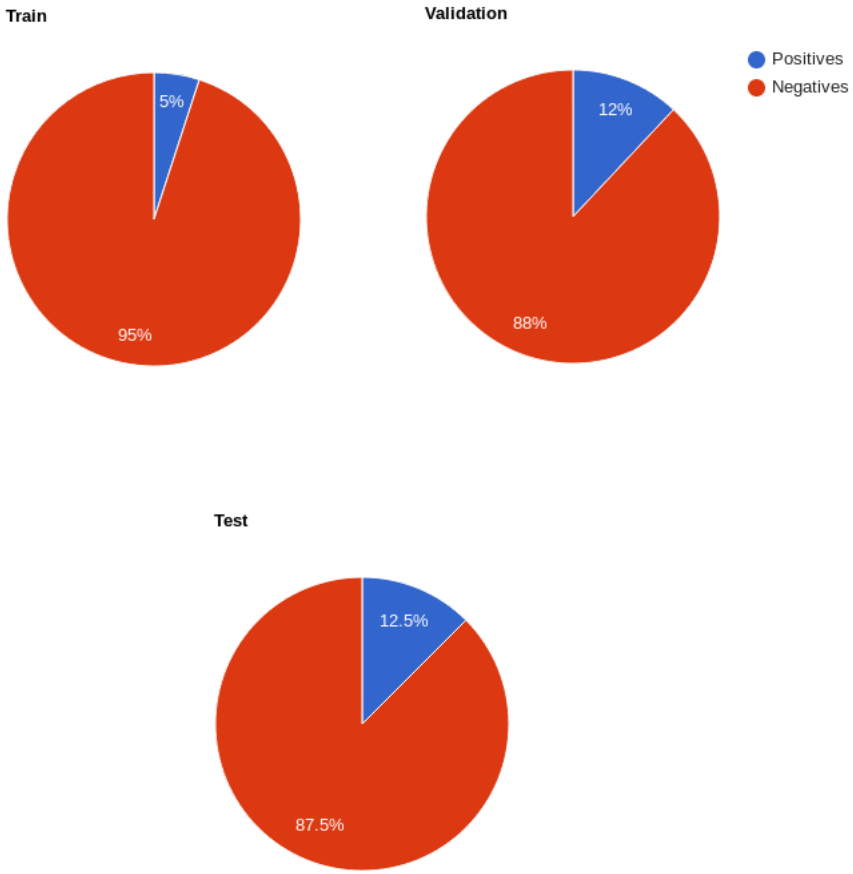

FIGURE (3.2) Percentage of relevant (positive) and irrelevant (negative) candidate answers in each set. The WikiQA dataset is imbalanced, as the negative class outnumbers the positive one.

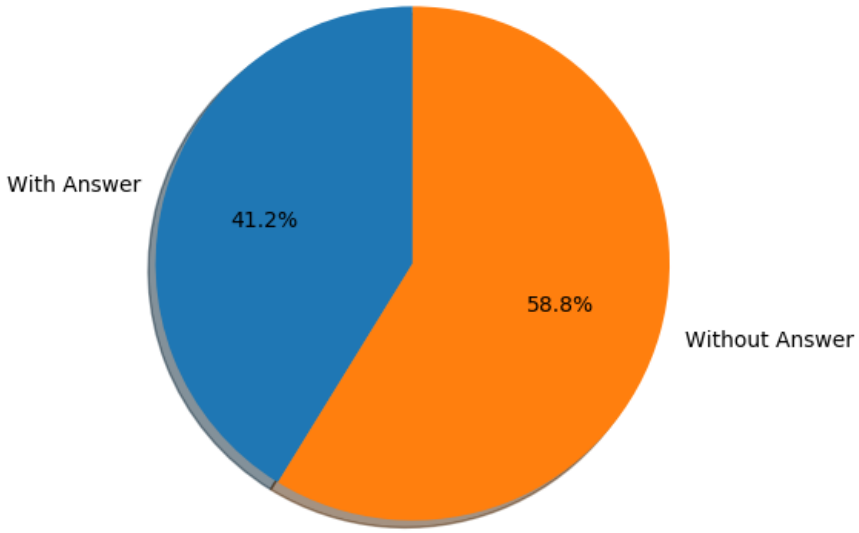

FIGURE (3.3) The training set of WikiQA contains a greater number of questions that do not have a relevant candidate answer.

3.3 BioASQ

For our experiments we focused on the dataset of the 5^{th} year of the BioASQ challenge (Task5b-phaseB) which contains a great number of questions [**2**]. For each question, the ID of the relevant documents and the sentences that answer the respective question in each document are also provided. In order to create the train, validation and test sets, we implemented a Python script that extracted the abstracts and the titles of the relevant documents and we saved both the relevant sentences and the irrelevant ones. All abstracts and titles were split into sentences using the sentence splitter of the NLTK[3] library. Sentences and titles were then preprocessed and tokenized using a modified version of the *"bioclean"*[4] function. More specifically, we modified the *"bioclean"* function so that text is also split on dashes ("-") which proved useful due to the fact that we observed an improvement to the functionality of the BM25 ranking function. Also our modified version of the *"bioclean"* function does not remove punctuation marks like its default version.

From the above method of extraction, the dataset will contain a greater number of irrelevant QA pairs than the number of the relevant ones. Consequently, we decided that it would be prudent to balance the train set. Again we used a script, which for each question's positive QA pairs randomly selects an equal number of unique negative pairs from our starting imbalanced dataset.

	QA pairs	**Questions**	**Positives**	**Negatives**
Train	43.750	1.615	21.875	21.875
Validation	12.689	120	*overlaps*	*overlaps*
Test	17.933	399	*overlaps*	*overlaps*

TABLE (3.3) Statistics for train, validation and test set of our BioASQ dataset. The first two columns refer to the number of QA pairs and unique questions in each set, while the last two contain the number of positive and negative QA pairs for each set.

The statistics for our final BioASQ dataset are reported in Table 3.3. We can see that in the field of positives and negatives we have the word *"overlaps"*. That means that in the case of BioASQ dataset we do not use the golden snippets for the validation and test set. Instead, we compute the overlap of bytes between the candidate answer and the golden snippets for the specific question. The overlap of bytes is actually the intersection of characters of the candidate answer with the total characters of the golden snippets (the relevant sentences for the given query). In order to compute the overlap of bytes, we use the official BioASQ script[5]. More information for this overlap-based computation method is outlined in the next chapter.

[3]The NLTK library: https://www.nltk.org/

[4]Standard version of the *"bioclean"* function. Available at: https://github.com/nlpaueb/BioIR/blob/master/reranking.py

[5]The official BioASQ script we used for the evaluation of the three models on the BioASQ dataset is available at: https://github.com/BioASQ/Evaluation-Measures

Chapter 4

Experiments

4.1 Basic Evaluation Measures

For the purpose of evaluating the performance of the neural network models and BM25 baseline, described in Chapters 2, on TREC QA and WikiQA, we used the Mean Average Precision (MAP) [**18**] and Mean Reciprocal Rank (MRR) evaluation measures. We chose MAP and MRR because the majority of the state of the art models in the QA domain are evaluated based on these two specific metrics. For the reader's convenience, we provide a simple description of the above evaluation measures:

- *Mean Average Precision (MAP)* is the mean average precision score for each question, defined as follows; the definition is based on the corresponding definition of Brokos [**19**]:

$$MAP = \frac{1}{|Q|} \cdot \sum_{i=1}^{|Q|} AP_i \tag{4.1}$$

  where $|Q|$ is the number of the questions and AP is the Average Precision. Average Precision (AP) takes into account the position of the returned sentence (candidate answer) in the list of results, to score the performance on a single question:

$$AP = \frac{\sum_{r=1}^{|L|} P(r) \cdot rel(r)}{|L_R|} \tag{4.2}$$

  where $|L|$ is the number of sentences returned, $P(r)$ is the precision when only the top r returned sentences are considered, L_R is the number of sentences in the golden snippets (set of all relevant sentences) and $rel(r)$ is 1 if the sentence positioned $r - th$ is annotated as relevant, else 0.

  Precision (P) is a basic measure and is defined as:

$$P = \frac{TP}{TP + FP} \tag{4.3}$$

  where TP (True Positives) is the number of returned relevant sentences and FP (False Positives) is the number of returned irrelevant sentences.

- *Mean Reciprocal Rank (MRR)* is defined as follows:

$$MRR = \frac{1}{|Q|} \cdot \sum_{i=1}^{|Q|} \frac{1}{rank_i} \tag{4.4}$$

where $rank_i$ refers to the ranking position of the first relevant document for the ith question. If no (truly) relevant document is returned for the ith question, then $\frac{1}{rank_i}$ is set to 0.

4.2 Evaluation Measures for BioASQ's Snippet Extraction Task

For the evaluation on the BioASQ dataset we use a greater variety of evaluation measures, which are used by BioASQ itself in order to evaluate and rank the participating models. These measures are Mean Precision, Mean Recall, Mean F-Measure, MAP and GMAP [**20**].

In contrast to basic snippet extraction evaluation methods, BioASQ uses a completely different method to evaluate its candidate answer by modifying the basic formulas of Recall and Precision. More specifically a returned snippet may overlap with one or more golden snippets without being identical to any of them. A snippet is determined by the document it comes from and by the offsets (positions) in the article of the first and last characters of the snippet.

Following [**20**], let us call A the set of all the article-offset pairs of all the characters in the snippets returned by a system for a particular question and G the set of all the article-offset pairs of all the characters in the golden snippets of the question. Also let $|x|$ denote the cardinality of a set x. The definitions of the modified Precision and Recall for snippets are:

$$P_{snip} = \frac{|A \bigcap G|}{|A|} \tag{4.5}$$

$$R_{snip} = \frac{|A \bigcap G|}{|G|} \tag{4.6}$$

As a result, in order to calculate the BioASQ evaluation measures, we use the above formulas instead of the standard formulas of Precision and Recall. The remaining evaluation measures of BioASQ are as follows:

- *F-Measure*: the traditional F-Measure or F1 score is the harmonic mean of the precision and recall. In the case of snippets we use the modified formulas of precision and recall. As a result the new formula for F-Measure is the following one:

$$F_1 = 2 \cdot \frac{Psnip \cdot Rsnip}{Psnip + Rsnip} \tag{4.7}$$

- *MAP*: mean average precision was described in Section 4.1 and is the most important of all the evaluation measures as the models are ranked based on it. The new modified formula for snippet extraction is the same as Eq. in Section 4.1, except that we now use P_{snip}.

- *GMAP*: the geometric mean average precision is very similar to MAP, but it uses the geometric instead of the arithmetic mean, which places more emphasis on improvements in low performing queries. Using the following equation we can compute the GMAP for a set of queries:

$$GMAP = \sqrt[n]{\prod_{i=1}^{n} AP_i + b} \tag{4.8}$$

where b is actually a really small number added in order to handle cases where the respective average precision is equal to zero.

4.3 Results on Development Data

In this subsection we present and compare the performance of the three neural network models and the BM25 baseline. In order to evaluate each model on the validation set we trained each model ten times, using different random seeds and picked the one that achieved the highest MAP score (in any epoch). This process is repeated for each one of our three datasets. We used the selected trained models (of the above process) for the evaluation of the test sets (more information in the next subsection). For the experiment we used Word2Vec [**10**] embeddings, pre-trained by Brokos [**19**] on the PubMed corpus[1], while for TREC QA[2] and WikiQA[3] we used the pre-trained embeddings that the authors [**11**; **12**] used for their own experiments.

The following figures illustrate the performance of the three models on each dataset. For convenience we named the model of Severyn and Moschitti [**5**] "SM-CNN", while the names BCNN and ABCNN3 remain the same. For every model there is a figure for MAP (left) and one for MRR (right). The meaning of each curve is as follows:

- *model*: The performance of the model without any modifications or additional external features.
- *model+BM25*: The performance of the system when we include the BM25 score as an additional feature.
- *BM25*: The performance of the BM25 ranking function on its own.

The models were trained for 50 epochs. We use interpolation for the illustration of each curve.

[1]The pre-trained embeddings used for the BioASQ dataset are available at: http://nlp.cs.aueb.gr/software.html

[2]The embeddings used for TREC QA dataset are available at: https://drive.google.com/drive/folders/0B-yipfgecoSBfkZlY2FFWEpDR3M4Qkw5U055MWJrenE5MTBFVXlpRnd0QjZaMDQxejh1cWs

[3]The embeddings used for WikiQA dataset are available at: https://drive.google.com/uc?id=0B7XkCwpI5KDYNlNUTTlSS21pQmM&export=download

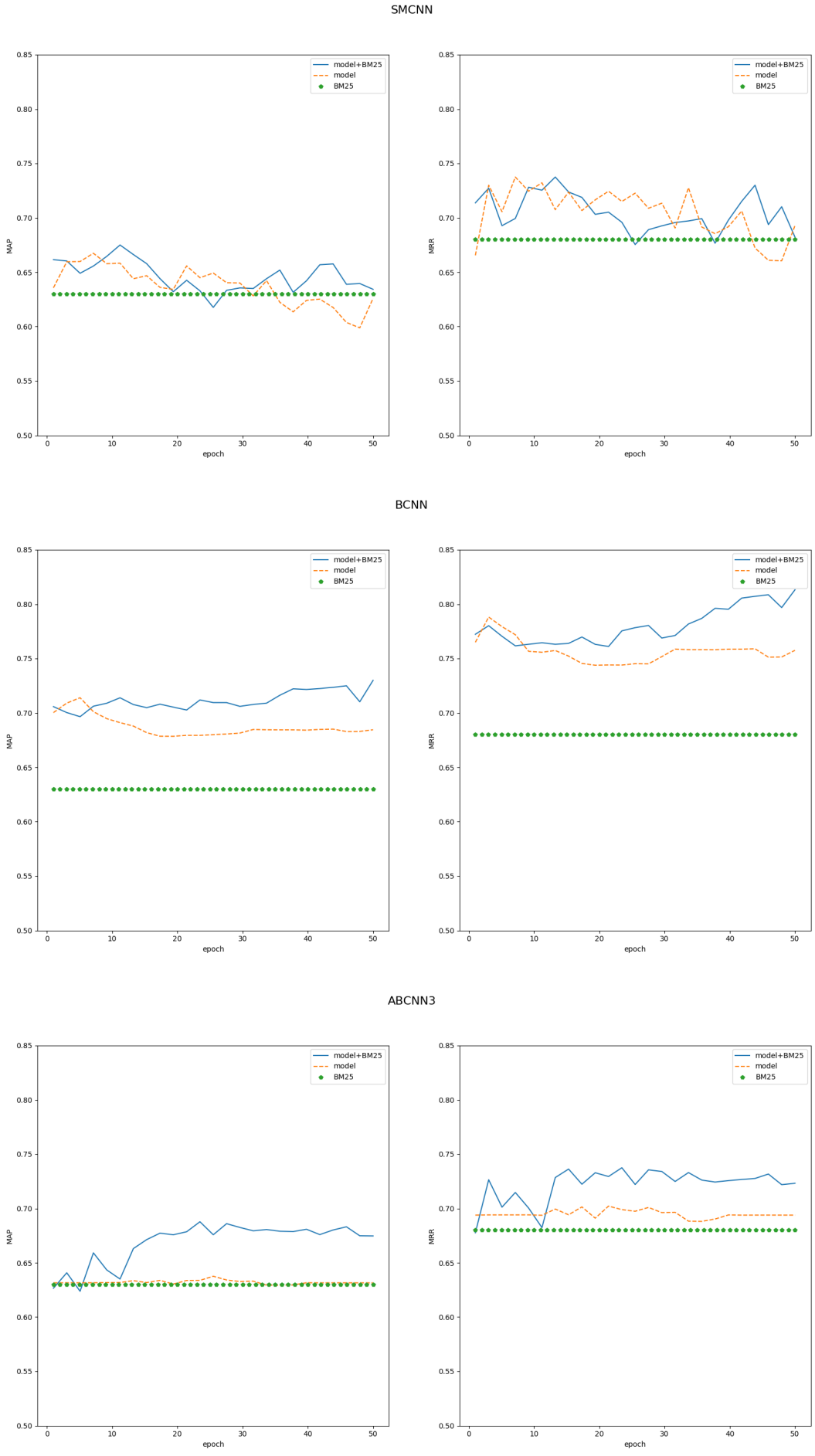

FIGURE (4.1) Performance of SMCNN, BCNN, ABCNN3 and BM25 on TREC QA dataset.

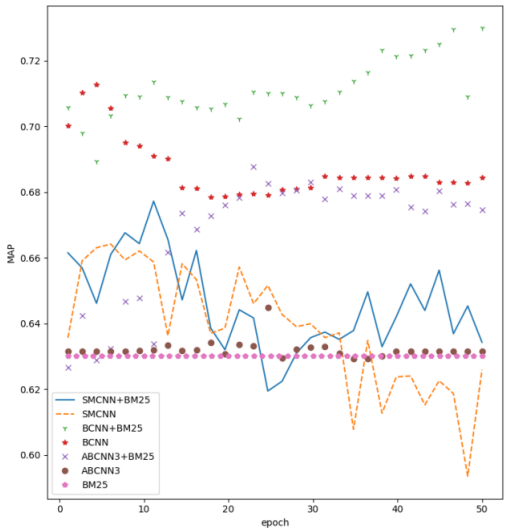

FIGURE (4.2) MAP scores for TREC QA dataset.

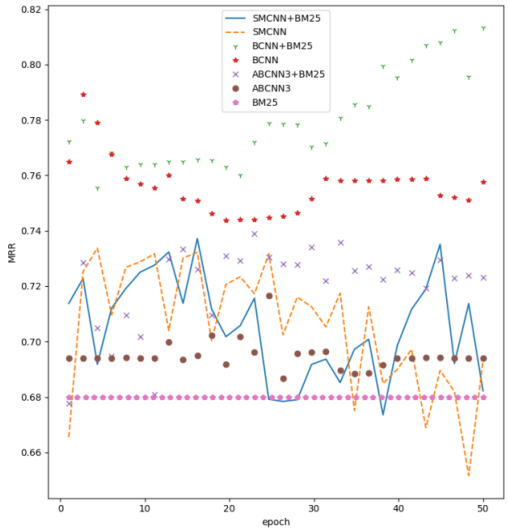

FIGURE (4.3) MRR scores for TREC QA dataset.

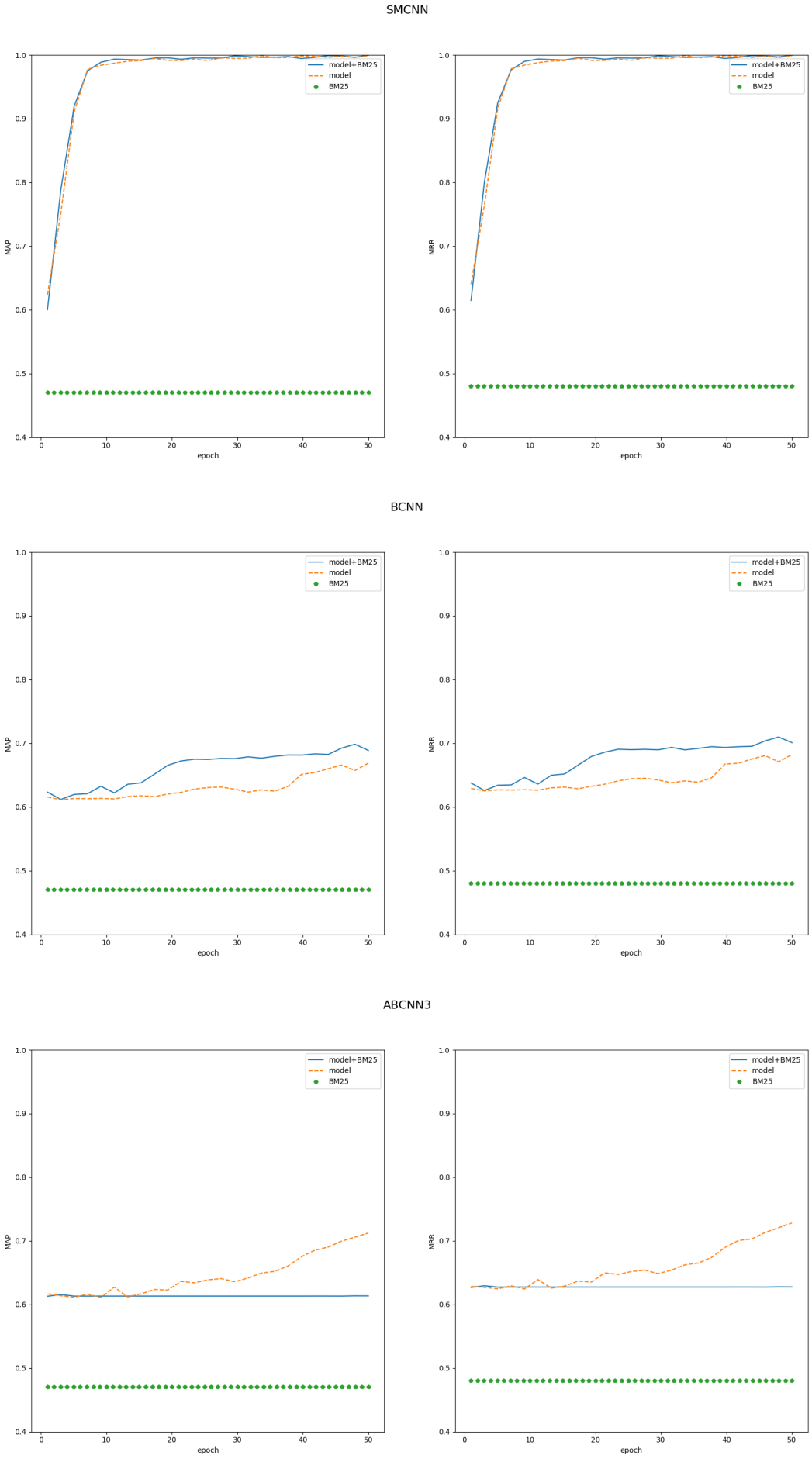

FIGURE (4.4) Performance of SMCNN, BCNN, ABCNN3 and BM25 on WikiQA dataset.

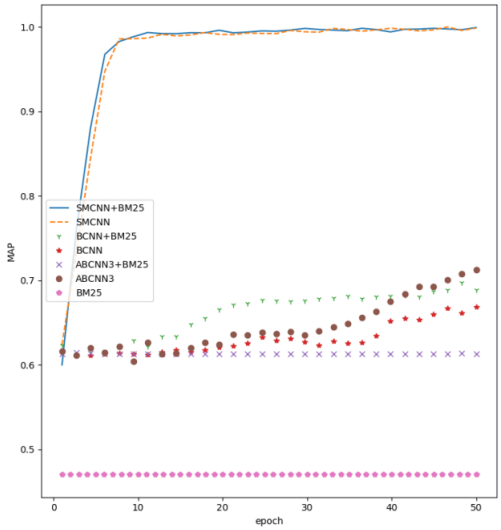

FIGURE (4.5) Performance of all models on MAP score for WikiQA dataset.

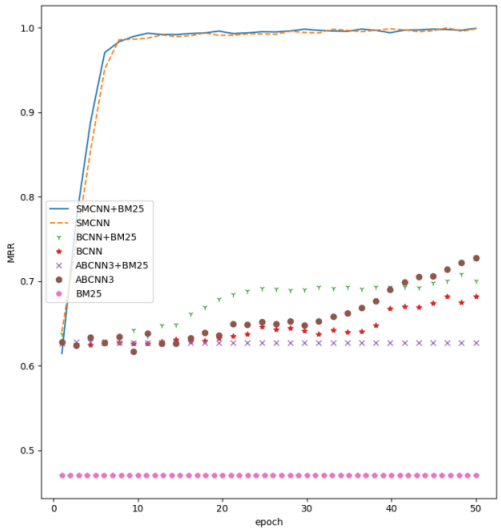

FIGURE (4.6) Performance of all models on MRR score for WikiQA dataset.

On the TREC QA dataset, all neural network models outperform BM25, and they all benefit when BM25 is integrated as an external feature. The best system is BCNN+BM25.

Without a doubt the results of the WikiQA dataset are intriguing. First of all we observe that the BM25 ranking function is outperformed from all perspectives. SMCNN tends to show a strange behavior in the validation set as it achieves a MAP and MRR score of almost 1 (100%). On the other hand, its performance on the training and test set of the respective dataset is normal (the corresponding diagrams are not shown to save space). BCNN seems to have a more reasonable behavior. Furthermore when we include BM25 as an external feature the performance of BCNN model is improved. Finally simple ABCNN3 model seems to perform better as it achieves the highest MAP and MRR score, but we can see a strange behavior when we include BM25. As a result, MAP and MRR curves seems to be horizontal lines. Hence, BM25 actually deteriorates the performance of the ABCNN3 model. Further experimental results (Precision, Recall, F1, MAP, GMAP) for all methods are provided in Appendix A.

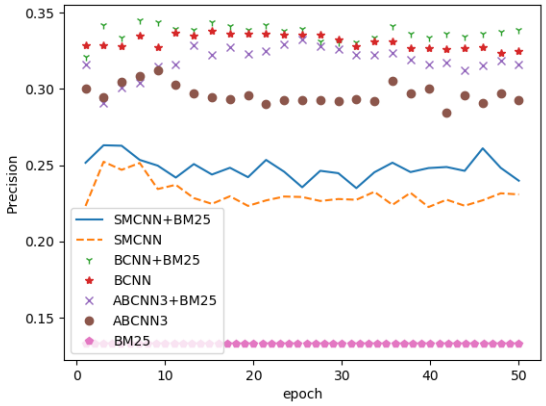

FIGURE (4.7) Precision scores for BioASQ dataset.

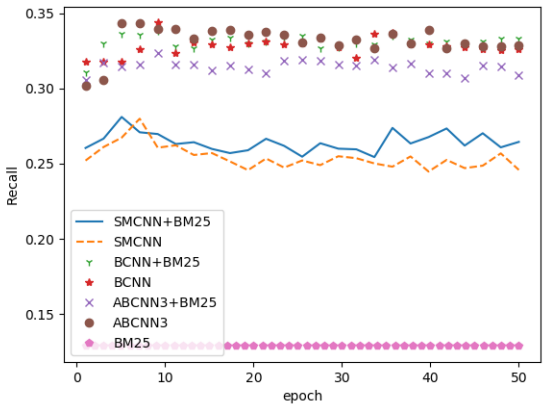

FIGURE (4.8) Recall scores for BioASQ dataset.

BioASQ ranks the participing models based on the MAP score, so in this thesis we will do exactly the same. As a result, it is obvious that the BM25 baseline and

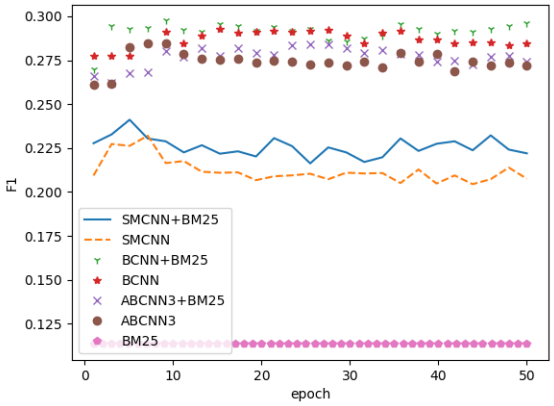

FIGURE (4.9) F1 scores for BioASQ dataset.

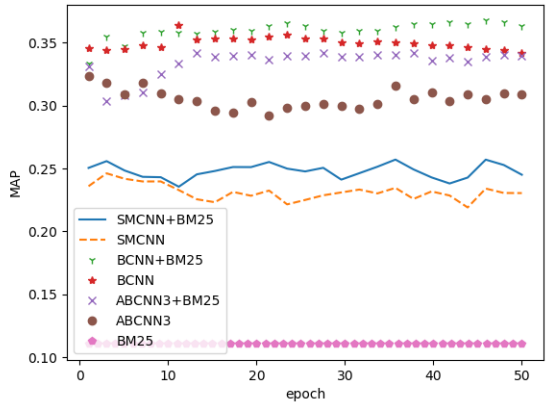

FIGURE (4.10) MAP scores for BioASQ dataset.

SMCNN are completely outperformed by BCNN and ABCNN3 models. BCNN performs clearly better than ABCNN3 on BioASQ, which makes BCNN a better model to use for the 6th year of the BioASQ competition. An important final observation is the fact that the integration of BM25 score improves the performance of all three models.

Summarizing the knowledge we acquired from the above figures, we present the following tables in which we can see the highest scores achieved from each model (over the 10 repetitions with different random seeds) on the validation set (Table 4.1) and the average score over all 10 repetitions (Table 4.2):

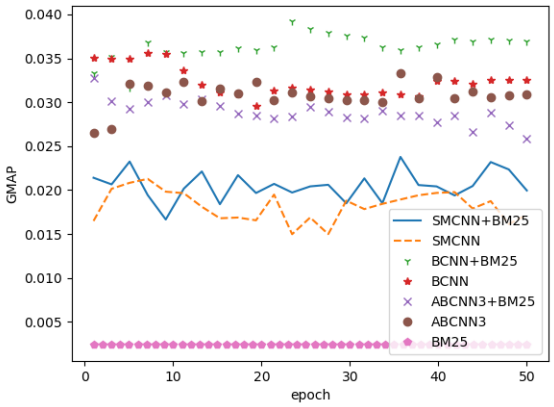

FIGURE (4.11) GMAP scores for BioASQ dataset.

		TREC QA		WikiQA		BioASQ				
		MAP	MRR	MAP	MRR	Precision	Recall	F1	MAP	GMAP
BM25		0.63	0.68	0.47	0.48	0.13	0.12	0.11	0.11	0.01
SMCNN	model	0.66	0.73	0.99	1.0	0.25	0.27	0.23	0.24	0.02
	model+BM25	0.68	0.73	0.99	0.99	0.27	0.28	0.24	0.25	0.02
BCNN	model	0.71	0.78	0.67	0.69	0.34	***0.34***	***0.30***	0.36	0.03
	model+BM25	***0.72***	***0.81***	0.69	0.70	***0.34***	0.34	0.29	***0.36***	***0.03***
ABCNN3	model	0.64	0.71	***0.71***	***0.72***	0.31	0.34	0.28	0.32	0.03
	model+BM25	0.69	0.78	0.61	0.62	0.33	0.32	0.28	0.34	0.03

TABLE (4.1) Best performance (over 10 repetitions with random seeds) of SMCNN, BCNN, ABCNN3 and BM25 ranking function on the validation set of each dataset.

		TREC QA		WikiQA		BioASQ				
		MAP	MRR	MAP	MRR	Precision	Recall	F1	MAP	GMAP
BM25		0.63	0.68	0.47	0.48	0.13	0.12	0.11	0.11	0.01
SMCNN	model	0.66	0.73	0.99	0.99	0.23	0.26	0.22	0.23	0.02
	model+BM25	0.65	0.72	0.99	0.99	0.25	0.26	0.23	0.24	0.02
BCNN	model	0.70	0.77	0.66	0.67	0.33	0.32	0.28	0.35	0.03
	model+BM25	***0.72***	***0.80***	0.68	0.69	***0.33***	**0.33**	**0.29**	***0.36***	***0.03***
ABCNN3	model	0.63	0.69	***0.70***	***0.72***	0.30	0.32	0.27	0.31	0.03
	model+BM25	0.67	0.74	0.61	0.62	0.32	0.31	0.28	0.33	0.02

TABLE (4.2) Average scores (over 10 repetitions with random seeds) for SMCNN, BCNN and ABCNN3 on the validation set of each dataset.

4.4 Results on Test Data

		TREC QA		WikiQA		BioASQ				
		MAP	MRR	MAP	MRR	Precision	Recall	F1	MAP	GMAP
BM25		0.60	0.67	0.51	0.51	0.24	0.41	0.26	0.16	0.04
SMCNN	model	0.64	0.72	0.56	0.58	0.34	0.53	0.36	0.25	0.13
	model+BM25	0.64	0.71	0.60	0.61	0.35	0.55	0.37	0.27	0.15
BCNN	model	**0.71**	**0.78**	**0.66**	**0.67**	0.42	0.65	0.45	0.38	0.27
	model+BM25	0.68	0.74	0.62	0.63	**0.44**	**0.66**	**0.46**	**0.39**	**0.29**
ABCNN3	model	0.70	0.76	0.62	0.63	0.38	0.61	0.41	0.34	0.22
	model+BM25	0.71	0.75	0.63	0.63	0.41	0.63	0.43	0.37	0.25

TABLE (4.3) Performance of SMCNN, BCNN, ABCNN3 and BM25 ranking function on the test set of each dataset.

In order to evaluate the models on each dataset's test data, we picked the epoch for which we achieved the maximum MAP performance. More specifically, during the training process we save a model for each epoch. So if for example we achieve the maximum MAP score in epoch E, we will load and use for the evaluation on the test set the model of epoch E. In Table 4.3, we can observe the scores on test data for each dataset.

BCNN seems to perform very well on all datasets and outperforms SMCNN, ABCNN3 and the BM25 baseline. It is also remarkable that the integration of BM25 improves the results for the BioASQ dataset, but on the other hand it deteriorates the performance of BCNN for TREC QA and WikiQA. ABCNN3 seems to be outperformed by its simple version (BCNN) on all three datasets. Also in the case of the BioASQ dataset the simple BCNN (even without BM25) seems to achieve better results (Precision: 0.42, Recall: 0.65, F1: 0.45, MAP: 0.38, GMAP: 0.27) than ABCNN3 even if we use the "ABCNN3+BM25" version (Precision: 0.41, Recall: 0.63, F1: 0.43, MAP: 0.37, GMAP: 0.25). Finally, SMCNN and BM25 ranking function are completely outperformed.

4.5 Results on the 6th Year of the BioASQ Challenge

The final goal of the Thesis was to participate in BioASQ's 6^{th} year competition (Task 6b-Phase A). In order to do that we used a combination of models for the document retrieval task together with the BCNN model. We chose to use BCNN because during experiments on data from the previous years of the BioASQ challenge it seemed to outperform the other neural network models. In order to retrieve the relevant documents for each question we used modified versions of two state-of-the-art models; the Position-Aware Convolutional-Recurrent Relevance matching model (PACRR) **[21]**, a neural network model which aims to capture position-dependent interactions between a query and a document, and the Deep Relevance Matching Model (DRMM) **[22]** which calculates matching signals between query-document pairs which are then passed to neural operations to produce a relevance score. See Brokos et al. **[23]** for further details.

Using combinations of the above two modified document retrieval systems with the BCNN snippet extraction system, we end up with five (5) models in total:

- *aueb-nlp-1*: A combo of 10 runs (each trained with a different random seed) of a modified PACRR model for document retrieval followed by BCNN for snippet extraction.
- *aueb-nlp-2*: A combo of 10 runs of modified DRMM model for document retrieval followed by BCNN for snippet extraction.
- *aueb-nlp-3*: A combo of 10+10 runs of modified PACRR and DRMM models for document retrieval followed by BCNN for snippet extraction.
- *aueb-nlp-4*: Here we use a modified DRMM model with "density extension" for document retrieval followed by BCNN for snippet retrieval. The density extension considers all the windows of w consecutive tokens of the document and computes the relevance score per window. The final relevance score of a document is the sum of the original score computed over the entire document plus the maximum score over all the document's windows.
- *aueb-nlp-5*: Similar to aueb-nlp-4, however, for document ranking we threshold results and we do not necessarily return 10 documents (BioASQ demands to return up to 10 relevant documents for the document retrieval task and up to 10 snippets for the snippet extraction task).

An interesting technique that seems to improve our results in snippet retrieval task is to retain the top returned snippets (based on BCNN scores) for each query, and then re-rank them by the relevance scores of the documents they came from (based on the scores of the document retrieval model). This can be seen as an early step towards models that would learn to jointly retrieve documents and relevant sentences from them.

System	Mean Precision	Mean Recall	Mean F-Measure	MAP	GMAP
aueb-nlp-1	0.1160	0.2066	0.1296	0.0687	0.0029
aueb-nlp-2	0.1190	0.2182	0.1347	0.0665	0.0026
aueb-nlp-3	0.1180	0.2141	0.1329	0.0661	0.0028
aueb-nlp-4	0.1162	0.2009	0.1297	0.0694	0.0024
aueb-nlp-5	-	-	-	-	-

TABLE (4.4) Performance on BIOASQ6-TEST BATCH 1.

System	Mean Precision	Mean Recall	Mean F-Measure	MAP	GMAP
aueb-nlp-1	0.1088	0.2408	0.1329	0.0717	0.0034
aueb-nlp-2	0.1195	0.2492	0.1434	0.0750	0.0044
aueb-nlp-3	0.1125	0.2430	0.1355	0.0734	0.0033
aueb-nlp-4	0.1158	0.2498	0.1397	0.0713	0.0037
aueb-nlp-5	0.1679	0.3077	0.1939	0.1368	0.0045

TABLE (4.5) Performance on BIOASQ6-TEST BATCH 2.

System	Mean Precision	Mean Recall	Mean F-Measure	MAP	GMAP
aueb-nlp-1	0.1384	0.2288	0.1563	0.1331	0.0046
aueb-nlp-2	0.1308	0.2204	0.1494	0.1262	0.0034
aueb-nlp-3	0.1341	0.2263	0.1526	0.1294	0.0038
aueb-nlp-4	0.1325	0.2252	0.1519	0.1293	0.0038
aueb-nlp-5	0.2551	0.3412	0.2744	0.2314	0.0068

TABLE (4.6) Performance on BIOASQ6-TEST BATCH 3.

System	Mean Precision	Mean Recall	Mean F-Measure	MAP	GMAP
aueb-nlp-1	0.1010	0.2117	0.1211	0.0716	0.0009
aueb-nlp-2	0.1061	0.2327	0.1307	0.0821	0.0011
aueb-nlp-3	0.1018	0.2214	0.1251	0.0747	0.0009
aueb-nlp-4	0.0970	0.2091	0.1180	0.0750	0.0009
aueb-nlp-5	0.1671	0.2996	0.1940	0.1425	0.0017

TABLE (4.7) Performance on BIOASQ6-TEST BATCH 4.

System	Mean Precision	Mean Recall	Mean F-Measure	MAP	GMAP
aueb-nlp-1	0.0601	0.1571	0.0768	0.0357	0.0003
aueb-nlp-2	0.0560	0.1539	0.0728	0.0405	0.0004
aueb-nlp-3	0.0581	0.1560	0.0747	0.0377	0.0004
aueb-nlp-4	0.0617	0.1631	0.0790	0.0403	0.0004
aueb-nlp-5	0.0616	0.1480	0.0778	0.0526	0.0003

TABLE (4.8) Performance on BIOASQ6-TEST BATCH 5.

As we can observe in Tables 4.4-4.8, the performance of each system varies among the batches. The system "aueb-nlp-5" seems to achieve the best MAP scores in comparison not only to our other four systems but also to the systems of the other participants. As a result we had the top system for four of the five batches in total[4].

[4]See http://participants-area.bioasq.org/results/6b/phaseA/

Chapter 5

Related Work

There are several studies on deep learning systems for snippet selection which aim to improve the classification and the ranking of snippets extracted from a document based on a specific query. In [**24**] a system which uses stacked bidirectional LSTMs [**25**] is proposed. This system gets as input a question Q and a sentence S, it concatenates them in a single string and then forwards that string to the input layer of the BiLSTM neural architecture. The advantage of this architecture is the fact that all the contextual information across the entire sequence has been taken into consideration in order for the model to classify a given candidate answer. This approach does not require any syntactic parsing or external knowledge resources and also BM25 is used for the improvement of the model's performance.

Two other systems are proposed in [**26**] and [**27**]. In the first case [**26**] Noise-Contrastive Estimation is used with the purpose to learn joint representations of triplets of the form *(question, relevant sentence, irrelevant sentence)*. More specifically, the architecture consists of two neural network models, each of which takes a pair of a question and a candidate answer. Then one of the two neural networks processes the *(question, relevant sentence)* pair, while the other one will do the same for the *(question, irrelevant sentence)* pair. The system is expected to output a higher score for the relevant pair. Then a triplet ranking loss function is stacked on top in order to learn nonlinear feature correlations from the joint representations. The second model [**27**] calculates a semantic matching vector S for each word of a sentence based on all words in the other sentence (the input is always a question-answer pair) and then based on S, each word vector is decomposed into a similarity and a dissimilarity component. Then a two-channel CNN operation is performed with the purpose to compose these components into a single feature vector which is used for the final prediction of the model.

In another study [**28**], a method called Bilateral Multi-Perspective Matching is used. Given a question Q and a candidate answer A, the model encodes them using a bidirectional LSTM and then it matches the encoded sentences in two directions ($Q \rightarrow A$ and $A \rightarrow Q$) from multiple perspectives, by comparing each time-step of each sentence against all time-steps of the other sentence. Then another bidirectional LSTM layer is used in order to aggregate the results produced by the matching layer into a fixed-length matching vector. The later is forwarded as input into a softmax layer in order to retrieve the final prediction. This model can be used for various NLP tasks, like paraphrase identification, natural language inference and answer sentence selection, and achieves state-of-the-art performance.

Finally, in [**29**] the authors use auto-encoders to learn to encode texts, and use the resulting encodings to compute similarity between text pairs in semantic similarity texts, including answer ranking.

Chapter 6

Conclusions and Future Work

The main objective of this thesis was to study and scrutinize various possible ways to use neural network architectures for the task of sentence selection for question answering. We selected three state of the art models to run experiments for three different datasets. We observed the performance of each model on TREC QA and WikiQA, two datasets that are well known standard benchmarks for the sentence selection task, and then we continued with the 5^{th} year dataset of the BioASQ challenge.

The architecture of the three systems (SMCNN, BCNN, ABCNN3) was actually similar, with small but important variations. The core functionality of the three architectures is the following one: A question-answer pair is given as input and then each sentence is converted into an embedding matrix by retrieving the embedding vectors for each word. Then using convolution and pooling layers we retrieve a final representation vector for each sentence which we combine with a number of additional features, like BM25 ranking score, in order to predict if the QA pair is actually a valid question-answer pair.

The experimental results verify the performance of SMCNN, BCNN and ABCNN3 on TREC QA and WikiQA datasets. The only deviation during our experiments was the fact that ABCNN3 seemed to be outperformed by its simple version, BCNN. On the other hand, based on our own experiments on the BioASQ dataset we reached the conclusion that BCNN seems to perform better than the other neural network architectures and the BM25 ranking function. In all three deep learning models, if we include the BM25 score as an additional feature, the performance of the neural network models is significantly improved.

There are a lot of interesting approaches that could be considered for future work, in order to improve our results. First of all, we could re-run the experiments using embeddings of higher dimensionality. Our pre-trained embeddings are produced by Word2Vec, so for this specific approach we could also use GloVe or fastText embeddings [**30**; **31**]. Another idea would be to use various data augmentation techniques in order to train our models, as the training set of BioASQ is small and imbalanced. Finally, we could create tf-idf scores especially for the WikiQA and TREC QA datasets (the tf-idf scores we use have been produced from PubMed abstracts).

Appendix A

Performance on Development Data

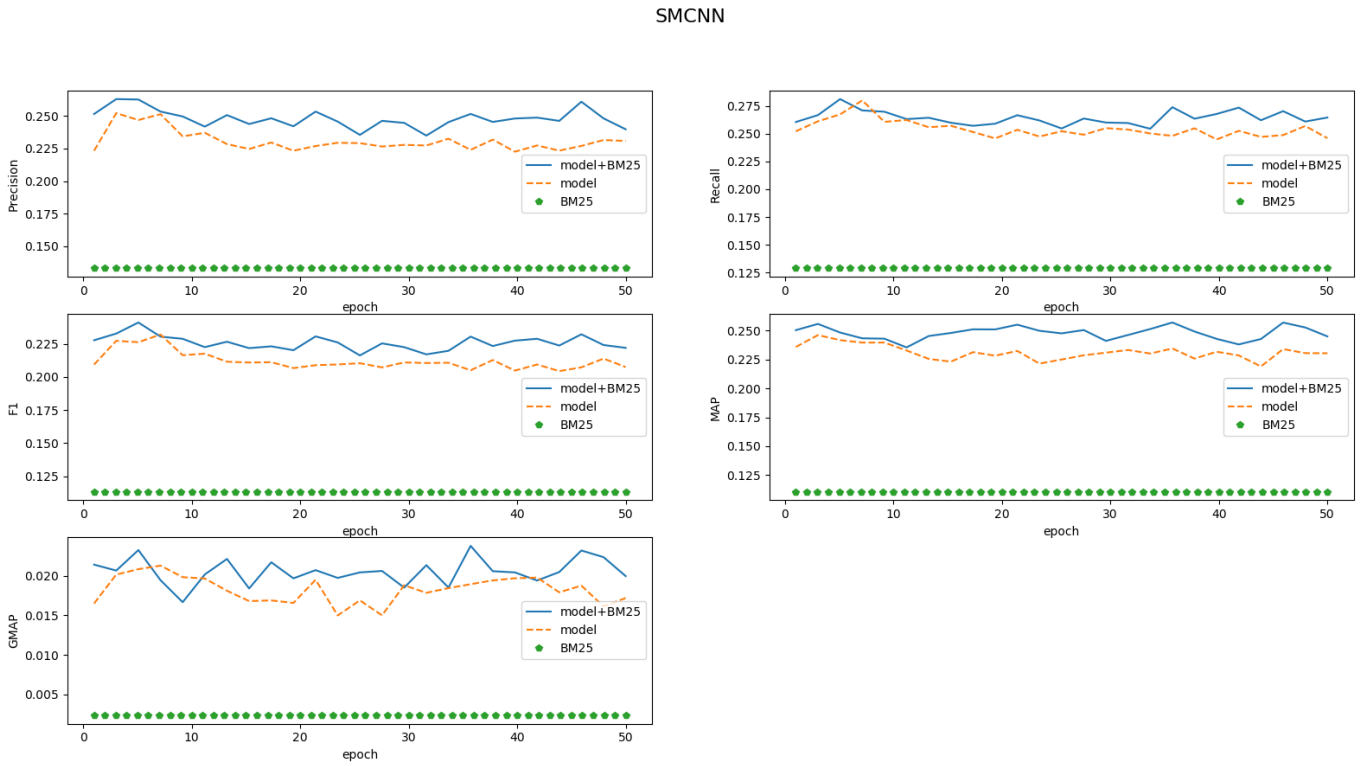

FIGURE (A.1) Performance of SMCNN on BioASQ dataset.

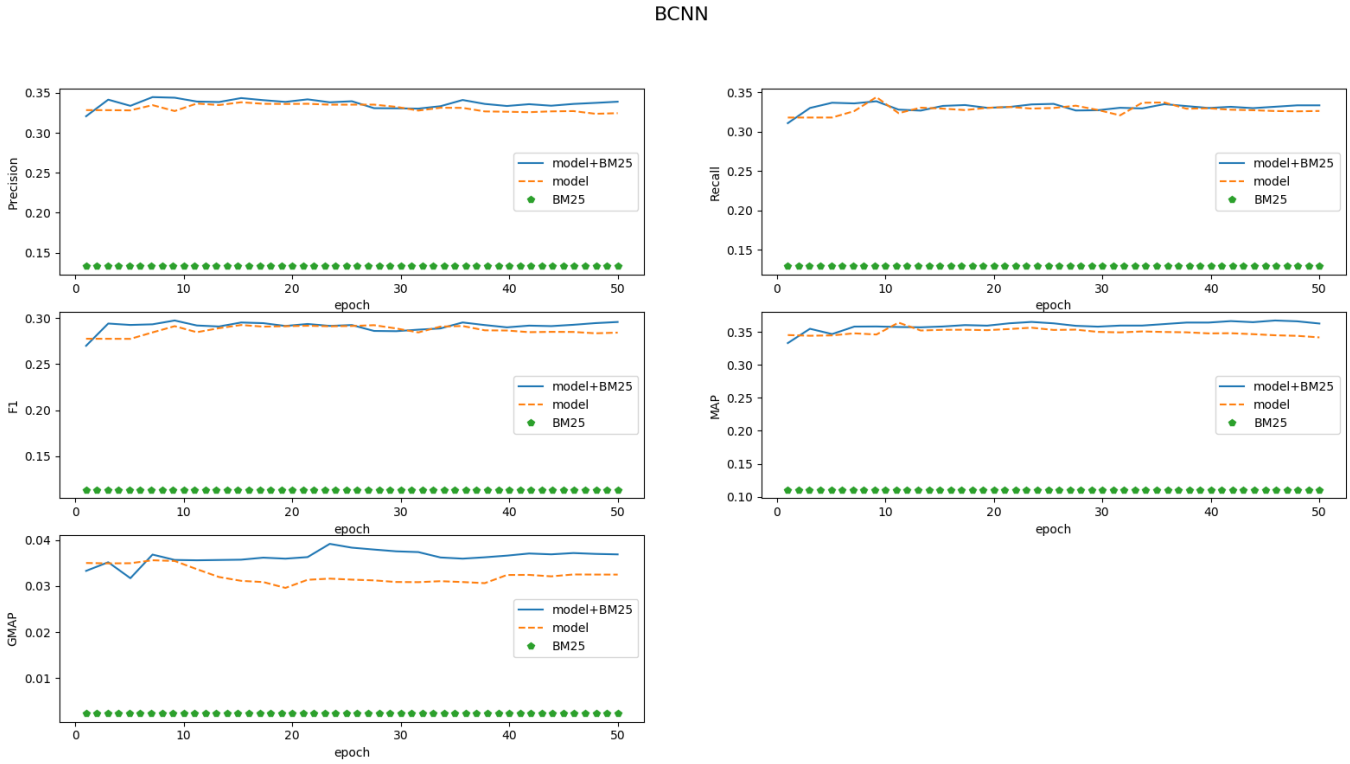

FIGURE (A.2) Performance of BCNN on BioASQ dataset.

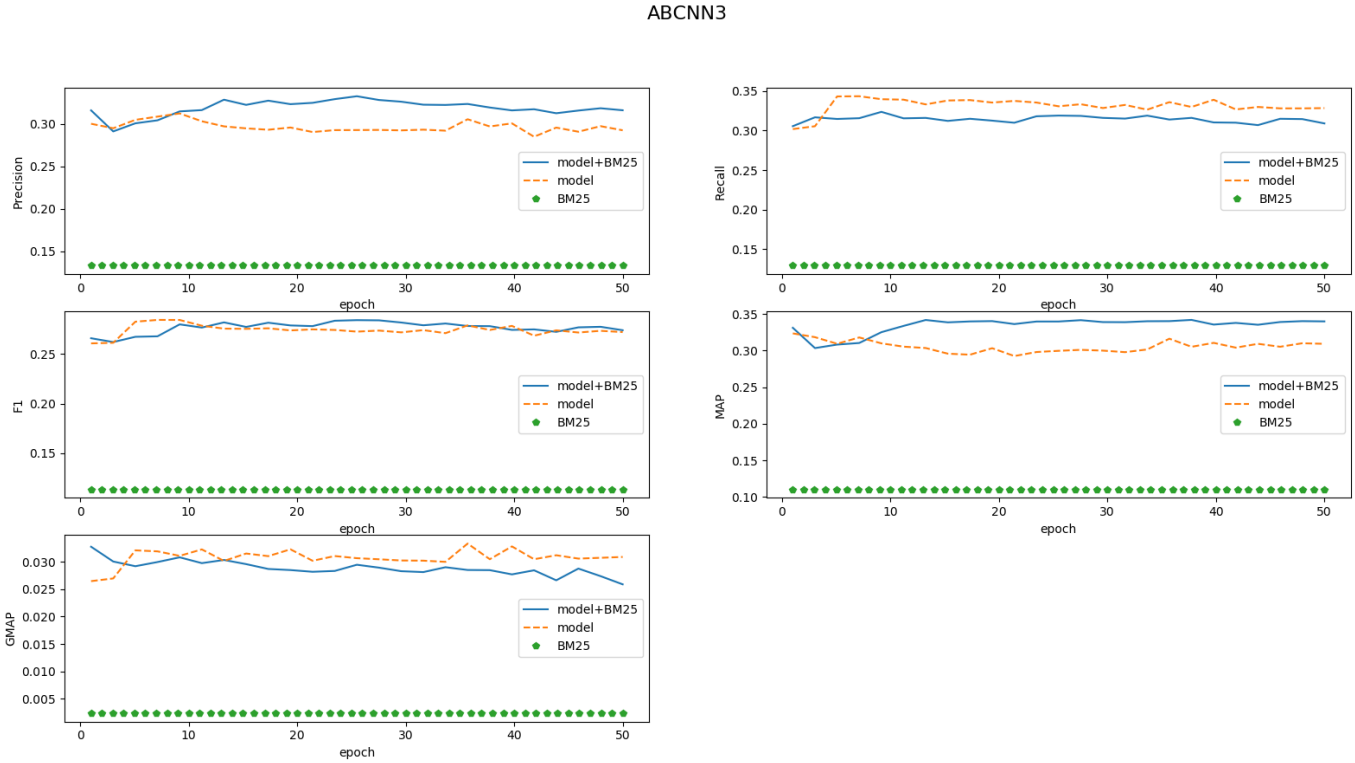

FIGURE (A.3) Performance of ABCNN3 on BioASQ dataset.

Bibliography

[1] Lei Yu, Karl Moritz Hermann, Phil Blunsom, Stephen Pulman. Deep Learning for Answer Sentence Selection. In Proceedings of the Deep Learning and Representation Learning Workshop: NIPS, 2014.

[2] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artiéres, Axel-Cyrille Ngonga Ngomo, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, Georgios Paliouras. An overview of the BioASQ large-scale biomedical semantic indexing and question answering competition. BMC Bioinformatics, 2015.

[3] Marc Moreno Lopez, Jugal Kalita. Deep Learning applied to NLP. arXiv preprint arXiv:1703.03091 [cs.CL], 2017

[4] Tom Young, Devamanyu Hazarika, Soujanya Poria, Erik Cambria. Recent Trends in Deep Learning Based Natural Language Processing. arXiv preprint arXiv:1708.02709 [cs.CL], 2017.

[5] Yoav Goldberg, Graeme Hirst. Neural Network Methods in Natural Language Processing (Synthesis Lectures on Human Language Technologies). Morgan and Claypool Publishers, 2017.

[6] Yanyao Shen, Hyokun Yun, Zachary C. Lipton, Yakov Kronrod, Animashree Anandkumar. Deep Active Learning for Named Entity Recognition. arXiv preprint arXiv:1707.05928 [cs.CL], 2017.

[7] Zhiguo Wang, Wael Hamza, Linfeng Song. k-Nearest Neighbor Augmented Neural Networks for Text Classification. arXiv preprint arXiv:1708.07863 [cs.CL], 2017.

[8] Spiros V. Georgakopoulos, Sotiris K. Tasoulis, Aristidis G. Vrahatis, Vassilis P. Plagianakos. Convolutional Neural Networks for Toxic Comment Classification. arXiv preprint arXiv:1802.09957 [cs.CL], 2018.

[9] Xu Jia, Efstratios Gavves, Basura Fernando, Tinne Tuytelaars. Guiding Long-Short Term Memory for Image Caption Generation. arXiv preprint arXiv:1509.04942 [cs.CV], 2015.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781 [cs.CL], 2013.

[11] Aliaksei Severyn, Alessandro Moschitti. Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 373–382, 2015.

[12] Wenpeng Yin, Hinrich Schutze, Bing Xiang, Bowen Zhou. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. arXiv preprint arXiv:1512.05193 [cs.CL], 2015.

[13] Mengqiu Wang, Noah A. Smith, Teruko Mitamura. What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 22-32, 2007.

[14] Yi Yang, Wen-tau Yih, Christopher Meek. WIKIQA: A Challenge Dataset for Open-Domain Question Answering. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2013–2018, 2015.

[15] Stephen Robertson, Karen Sparck Jones. Relevance Weighting of Search Terms. Journal of the American Society for Information Science, Volume 27 number 3, pages 129-148, 1976.

[16] Stephen Robertson, Hugo Zaragoza. The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends in Information Retrieval, Volume 3 number 4, pages 333-389, 2009.

[17] Ellen M. Voorhees. The TREC-8 question answering track report. In Proceedings of the 8th Text Retrieval Conference, pages 77-82, 1999.

[18] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze. Introduction to Information Retrieval, Cambridge University Press., 2008.

[19] George Brokos. Document Reranking with Deep Learning in Information Retrieval. MSc thesis, Department of Informatics, Athens University of Economics and Business, 2018.

[20] Prodromos Malakasiotis, Ioannis Pavlopoulos, Ion Androutsopoulos, Anastasios Nentidis. Evaluation Measures for Task B. ICT, 2018.

[21] Kai Hui, Andrew Yates, Klaus Berberich, Gerard de Melo. PACRR: A Position-Aware Neural IR Model for Relevance Matching. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1060–1069, 2017.

[22] Jiafeng Guo, Yixing Fan, Qingyao Ai, W. Bruce Croft. A Deep Relevance Matching Model for Ad-hoc Retrieval. In Proceedings of the 25th ACM International Conference on Information and Knowledge Management, pages 55–64, 2016.

[23] George Brokos, Polyvios Liosis, Ryan McDonald, Dimitris Pappas, Ion Androutsopoulos. AUEB at BioASQ 6: Document and Snippet Retrieval. Under review, 2018.

[24] Di Wang, Eric Nyberg. A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 707-712, 2015.

[25] Mike Schuster, Kuldip K. Paliwal. Bidirectional Recurrent Neural Networks. Signal Processing, IEEE Transactions, pages 2673–2681, 1997.

[26] Jinfeng Rao, Hua He, Jimmy Lin. Noise-Contrastive Estimation for Answer Selection with Deep Neural Networks. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pages1913–1916, 2016.

[27] Zhiguo Wang, Haitao Mi, Abraham Ittycheriah. Sentence Similarity Learning by Lexical Decomposition and Composition. arXiv preprint arXiv:1602.07019 [cs.CL], 2016.

[28] Zhiguo Wang, Wael Hamza, Radu Florian. Bilateral Multi-Perspective Matching for Natural Language Sentences. arXiv preprint arXiv:1702.03814 [cs.AI], 2017.

[29] Hadi Amiri, Philip Resnik, Jordan Boyd-Graber, Hal Daume III. Learning Text Pair Similarity with Context-sensitive Autoencoders. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1882–1892, 2016.

[30] Jeffrey Pennington, Richard Socher, Christopher D. Manning. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pages 1532–1543, 2014.

[31] Yanshan Wang, Sijia Liu, Naveed Afzal, Majid Rastegar-Mojarad, Liwei Wang, Feichen Shen, Paul Kingsbury, Hongfang Liu. A Comparison of Word Embeddings for the Biomedical Natural Language Processing. arXiv preprint arXiv:1802.00400 [cs.IR], 2018.