AUEB at BioASQ 7: Document and Snippet Retrieval

Dimitris Pappas[1,2], Ryan McDonald[1,3], Georgios-Ioannis Brokos[1], and Ion Androutsopoulos[1]

[1] Department of Informatics, Athens University of Economics and Business, Greece
pappasd@aueb.gr, ion@aueb.gr, g.brokos@gmail.com
[2] Institute for Language and Speech Processing, Research Center 'Athena', Greece
dpappas@ilsp.gr
[3] Google Research
ryanmcd@google.com

Abstract. We present the submissions of AUEB to the BIOASQ 7 document and snippet retrieval tasks (parts of Task 7b, Phase A). Our systems build upon the methods we used in BIOASQ 6. This year we also experimented with models that jointly learn to retrieve documents and snippets, as opposed to using separate pipelined models for document and snippet retrieval. We also experimented with models based on BERT [5]. Our systems obtained the best document and snippet retrieval results for all batches of the challenge that we participated in.

Keywords: Information Retrieval · Document Retrieval · Document Reranking · Snippet Retrieval · Snippet Extraction · Sentence Selection · Biomedical Question Answering · Machine Learning · Deep Learning

1 Introduction

BIOASQ [26] is a biomedical document classification, retrieval, and question answering competition. It provides, among other information, tuples containing questions, gold relevant documents, and gold relevant snippets of relevant documents. All questions are expressed in natural language by human experts of the biomedical field. For the document and snippet retrieval tasks, the competitors receive a set of questions and must retrieve relevant documents and then extract relevant snippets from the retrieved documents. The available documents are abstracts from a collection of approx. 28 million MEDLINE/PUBMED biomedical articles. In this paper, we provide an overview of the submissions of AUEB to the document and snippet retrieval tasks (parts of Task 7b, Phase A) of BIOASQ 7.[4]

Most related research for biomedical document retrieval and snippet retrieval (or 'snippet extraction' or 'sentence selection') focuses mainly on one of the two tasks. When tackling both tasks, researchers usually follow a pipelined architecture where two models are trained separately and then run in sequence: document retrieval followed by snippet extraction from the retrieved documents. A

[4] See http://bioasq.org/participate/challenges.

major novel research direction of our participation this year is a new deep learning model which is jointly trained for both document and snippet retrieval. We build upon our BIOASQ 6 models for document retrieval [3, 17] and modify them to also yield a relevance score for each *sentence* of the documents; we treat each sentence as a snippet, hence we use the two terms as synonyms. Since neural document retrieval methods are computationally intensive, we rely on conventional information retrieval (IR) methods to pre-fetch a list of possibly relevant documents, and then rerank the top retrieved documents and their sentences using the neural models. To our knowledge this is the first work on deep learning for joint document and snippet reranking. We also experimented with pipelined and joint models for document and snippet retrieval that employ BERT [5].

Our systems scored at the top for all batches of the challenge we participated in. Although a plain BERT model outperformed the other methods we considered for document retrieval, our joint document and snippet retrieval model obtained substantially better snippet retrieval results, even without using BERT and even though it uses much fewer parameters than the corresponding pipelined models. We make publicly available the database, code, and trained models.[5]

2 Document Retrieval Models

BIOASQ requires competitors to return a list of 10 relevant documents and 10 relevant snippets (from the 10 documents) per query. As already noted, we use conventional (BM25-based) IR methods to pre-fetch possibly relevant documents, which we then rerank using neural models. We experimented with two neural models for document reranking, one based on PACRR [17] and one based on BERT [5]. PACRR was one of the best document retrieval methods in BIOASQ 6; and BERT has led to state of the art results in several tasks [5].

2.1 Term-PACRR

The first model we use for document retrieval is TERM-PACRR [3, 17], a modification of PACRR [9].[6] To train TERM-PACRR, we use mini-batches containing randomly selected relevant and irrelevant documents (in equal numbers) from the top N documents that the IR engine retrieves per training query, and we minimize binary cross-entropy. As in [3], we use a final linear layer that combines the TERM-PACRR score with traditional IR features like BM25, unigram and bigram overlap, and IDF-weighted unigram overlap. Consult [3] for details.

2.2 BERT based document retrieval

In the second document retrieval model, we employ BERT [5], which has recently led to state of the art results in several tasks, including document reranking

[5] See `https://github.com/nlpaueb/aueb-bioasq7`.
[6] TERM-PACRR is called PACRR-DRMM in [17].

on other datasets [23, 30, 20]. We add a task-specific logistic regression classifier on top of BERT, similar to other deep learning architectures [25, 19, 17]. We pre-trained our own BERT model on the PUBMED corpus using the titles and abstracts of the articles, and identical parameter settings to the uncased BERT LARGE model [5].[7] This is similar to BIOBERT [13], however unlike that work, we do not initialize the model with the public pre-trained BERT BASE instance, and we use a custom wordpiece model [29] also trained on PUBMED.

When fine-tuning BERT on BIOASQ data or when using it a test time, we feed it with the concatenation of a question and a (relevant or irrelevant) document. As standard, a special [CLS] token is added to the start of the concatenation, while a [SEP] token separates the question from the document (concatenated title and abstract), as illustrated in Fig. 1. The output vector of BERT for the [CLS] token is passed through a logistic regression layer (linear layer with sigmoid) to obtain a BERT-based score for the document. This score is then concatenated to extra features of the document (BM25 score and string overlap features), which are the same as in TERM-PACRR. Finally, another logistic regression layer is applied to the concatenated vector to get the final score of the document.

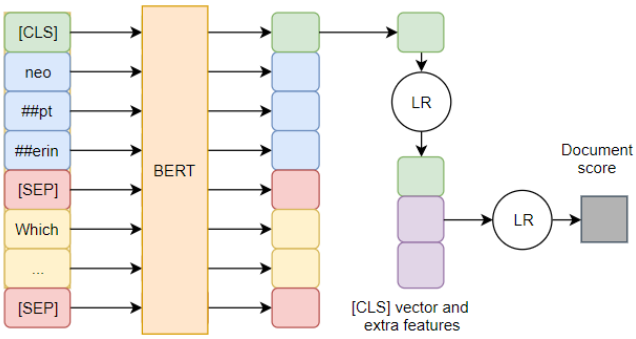

Fig. 1. BERT document ranking, with extra features added in a final layer.

During fine-tuning, negative samples (irrelevant documents to be concatenated with a training query) are drawn randomly from the non-relevant (according to the expert annotators) documents in the list of top N documents that the conventional IR system (the same as in the other methods) returned for the particular query. Critically, we found that incurring two losses per training instance helped accuracy. The first loss is standard, the binary cross-entropy of the final document score. The second loss is also binary cross-entropy, but computed on the BERT-based score, before concatenating it with the extra features. The two losses are summed. We found that this forced the model to use the BERT layers more effectively, otherwise the system tended to rely almost exclusively on the additional features during training.

Similarly to [3], we also experiment with a *high-confidence* version of BERT.[8] In this model, only documents with scores (probabilities of being relevant)

[7] Max length was set to 512. Instances exceeding this threshold where truncated.
[8] In [3], the high-confidence models were for another model, ABEL-DRMM.

greater than 0.01 were returned as relevant, hence fewer than 10 documents (the maximum allowed in BIOASQ) might be returned. This helped improve snippet retrieval as it focused that component only on the most relevant documents.

3 Snippet Retrieval Models

For snippet retrieval, we used two deep learning methods. The first one is the 'basic CNN' of [32], BCNN for short. It had the best snippet retrieval results in BIOASQ 6 [3]. The second method, POSIT-DRMM, PDRMM for short, had the best *document* retrieval results in the experiments of [17], but here we use it to score *snippets*, as a first step towards a joint model for document and snippet retrieval.

3.1 BCNN

BCNN [32] is a CNN-based snippet scoring method, which is fed with pairs consisting of a query and a snippet (relevant or irrelevant). For each pair, it returns an estimate of the probability that the snippet is relevant to the query. Following [3], we concatenate the score of BCNN to extra features of the snippet (sentence): the length of the sentence and the query (in tokens), the BM25 score of the sentence compared to the query, the number of tokens in the sentence excluding stopwords, the unigram and bigram token overlap of the sentence and the query, and finally the sum of the IDF scores of the overlapping tokens of the sentence and query divided by the sum of the IDF scores of the query's tokens. A logistic regression layer is then applied to obtain the final score of the snippet. As in [3], BCNN is trained on (relevant and irrelevant) snippets sampled (in equal numbers) from the relevant (gold) documents in the list of top N documents returned by the IR engine. Consult [3] for further details.

3.2 PDRMM

The second model we investigate is a modification of POSIT-DRMM [17], henceforth PDRMM. PDRMM was proposed for document scoring (reranking of documents retrieved by a conventional IR system), but here we use it for snippet scoring (reranking the sentences of retrieved documents). We first describe the original PDRMM, and then how we modified it to score snippets.

Given a query $q = \langle q_1, \ldots, q_n \rangle$ of n query terms (q-*terms*) and a document $d = \langle d_1, \ldots, d_m \rangle$ of m terms (d-*terms*), PDRMM computes context-sensitive term embeddings $c(q_i)$ and $c(d_i)$ from the static (e.g., WORD2VEC) embeddings $e(q_i)$ and $e(d_i)$ by applying two stacked convolutional layers with trigram filters, residuals [8], and zero padding to q and d, respectively.[9]

PDRMM then computes three similarity matrices S_1, S_2, S_3, each of dimensions $n \times m$ (Fig. 2). Each element $s_{i,j}$ of S_1 is the cosine similarity between

[9] In [17], a BILSTM is used instead of convolutions, but the latter are faster and do not degrade performance.

$c(q_i)$ and $c(d_j)$. S_2 is similar, but uses the static word embeddings $e(q_i), e(q_j)$. S_3 uses one-hot vectors for q_i, d_j, signaling exact matches. To each matrix (S_1, S_2, or S_3) we apply three row-wise pooling operators to extract 9 features for each q-term: max-pooling (to obtain the similarity of the best match between the q-term of the row and any of the d-terms), average pooling (to obtain the average match of each q-term to all d-terms), and average of k-max (to obtain the average similarity of the k best matches per q-term).[10] We concatenate the three features extracted from each row of the three similarity matrices (9 features in total) and concatenate them to obtain a new matrix S' of dimensions $n \times 9$ (Fig. 2, right). Each row of S' indicates the similarity of the corresponding q-term to any of the d-terms, through three different views of the terms (one-hot, static, context-aware embeddings). Each row of S' is then passed to a Multi-Layer Perceptron (MLP) to obtain a single match score per q-term.[11]

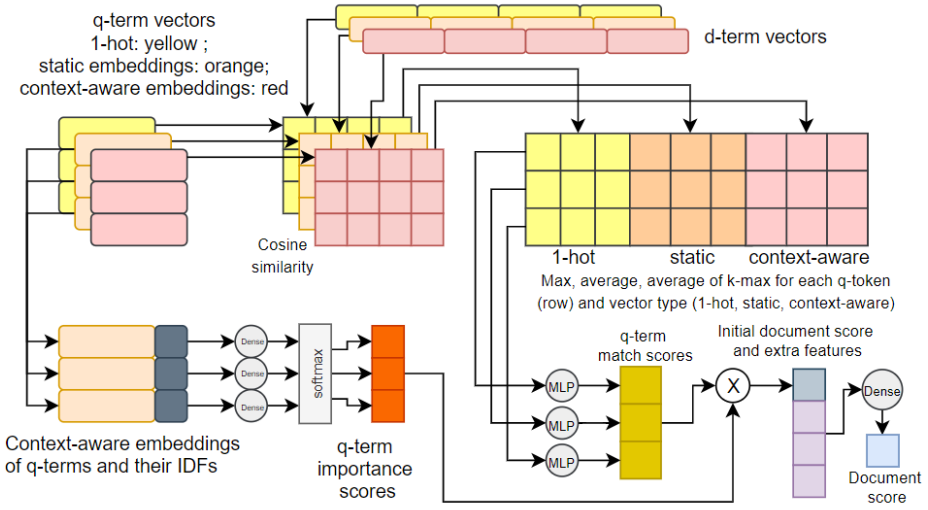

Fig. 2. PDRMM scoring documents with respect to a query. The same model can be used to score individual sentences with respect to a query, with different extra features.

Each context-aware q-term embedding is also concatenated with the corresponding IDF score (Fig. 2, bottom left) and passed to another linear layer that computes a score for each q-term. A SoftMax activation function is then applied across all the q-term scores to compute the importance of each q-term (e.g., words with low IDFs may not be helpful to answer the question).[12] Let v be the vector containing the n match scores of the q-terms, and u the vector of the corresponding n importance scores (Fig. 2, bottom). We extract an initial relevance

[10] In our experiments, $k = 5$. We added the average pooling to PDRMM to balance the other two pooling operators that favor long documents.

[11] This MLP consists of one dense layer with 8 neurons and leaky RELU activation function, followed by a second dense layer with 1 output and no activation function.

[12] The importance scores of the q-terms can also be viewed as self-attention scores.

score for the document as $\hat{r}(q,d) = v^T u$, which is then concatenated with four extra features: z-score normalized BM25 [24]; percentage of q-terms with exact match in d (regular and IDF weighted); percentage of q-term bigrams matched in d. An MLP computes the final relevance $r(q,d)$ from the five features.[13]

In its original form, PDRMM is trained on triples $\langle q, d, d' \rangle$, where d is a relevant document from the top N that the IR engine (the same as in the other methods) returned for query q, and d' a randomly sampled irrelevant document among the top N. Hinge loss is used, requiring $r(q,d)$ to exceed $r(q,d')$ by a margin.

In this work, we also use PDRMM to score snippets (sentences) by feeding it with query-snippet pairs, instead of query-document pairs, i.e., the d-terms (top of Fig. 2) are now the tokens of a particular snippet s from a retrieved document (relevant or irrelevant), and the output (bottom right of Fig. 2) is now the relevance score $r(q,s)$ of s. We also use different extra features when scoring snippets with PDRMM, which are the same as in BCNN (Section 3.1), instead of the extra features that are used when PDRMM scores documents. PDRMM is again trained on triples $\langle q, d, d' \rangle$, where q is a query, while d and d' are relevant and irrelevant documents, respectively, sampled from the top N documents that the IR engine returned for query q. Unlike the original PDRMM that scores documents, we use binary cross-entropy loss when training PDRMM to score snippets, treating the snippets of d that were selected by BIOASQ's human annotators as relevant, and all the other snippets from d and d' as irrelevant.

4 Joint Document and Snippet Retrieval Models

4.1 JPDRMM

As PDRMM can be used for both tasks, we create a joint PDRMM-based model, called JPDRMM, which given a query and a document, outputs relevance scores for each sentence (snippet) of the document, along with a relevance score for the entire document. JPDRMM applies the same process described in Section 3.2 to compute a score for each sentence in the document (Fig. 2, now operating on sentences). Then the maximum score of all the sentences is selected and concatenated to the extra features of the document (left part of Fig. 3), which are the same as when PDRMM scores documents.[14] The score of the document is computed by applying an MLP to the concatenated features.[15] The scores of the sentences are then revised to take into account the score of the entire document; the intuition is that snippets from relevant documents are more likely to be relevant. To do so, we concatenate the score of each sentence to the document

[13] This MLP also consists of one dense layer with 8 neurons and leaky RELU activation function, followed by a second dense layer with no activation function.

[14] We also experimented with other pooling operators to obtain the document score from the sentence scores, including combinations of max-pooling, average pooling, average of top k pooling, but they did not improve performance.

[15] This MLP consists of one dense layer with 8 neurons and leaky RELU activation function, followed by a second dense layer with no activation function.

score (Fig. 3, right part), and pass each pair of sentence-document scores through a logistic regression layer to obtain the final sentence score.

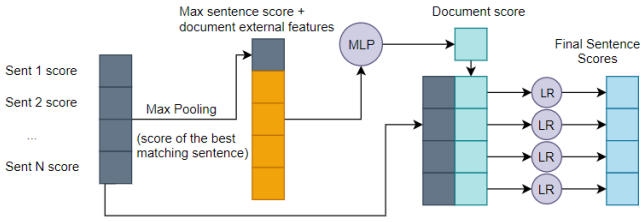

Fig. 3. The final layers of JPDRMM. The scores of the sentences (left) are generated by PDRMM (Fig. 2) operating on sentences. The maximum sentence score is concatenated with the external features of the document. An MLP produces the document score. A logistic regression layer then revises the score of each sentence, taking into account the original score of the sentence and the score of the document.

Like the original PDRMM, JPDRMM is trained on triples $\langle q, d, d' \rangle$, where q is a query, and d, d' are relevant and irrelevant documents, respectively, sampled from the top N documents returned by the IR engine for q. In this case, however, we apply a sentence splitter to d and d', and use JPDRMM to obtain relevance scores for d, d', and each one of their sentences. We compute a document hinge loss from the scores of d and d' as when PDRMM scores documents, and a binary cross-entropy loss for each sentence (relevant or irrelevant) of d and d' as when PDRMM scores sentences. The document hinge loss is added to the average sentence cross-entropy loss (averaged over all the sentences of d and d'), and their sum is used to train the entire model via backpropagation.

We create two versions of JPDRMM: one using pre-trained WORD2VEC embeddings, and one using pre-trained embeddings obtained from the top layer of the publicly available BERT BASE instance [5].[16] We call W2V-JPDRMM and BERT-JPDRMM the two versions, respectively. BERT's tokenizer splits words into subword units (wordpieces) [29]. In BERT-JPDRMM, in order to use IDF scores of entire words and compute exact matches across entire words, as in W2V-JPDRMM, we reconstruct the words from the subword units before feeding them to the rest of the model. Also, we use BERT's top-level embedding for the first wordpiece of each reconstructed word as the pretrained embedding of that word.

5 Overall System Architecture

Figure 4 presents the architecture of our pipelined systems. The first step is retrieving N documents using a conventional BM25-based IR engine given a user question; see Section 6.1 below for details. Then a neural document retrieval model reranks the N documents and selects the top K_d. The K_d documents are reranked by a neural snippet retrieval model, which returns the top K_s snippets. BIOASQ requires $K_d = K_s = 10$, and we set $N = 100$.[17]

[16] We also experimented with BIOBERT [13], but there was no notable improvement.
[17] Setting N to larger values had no impact on the final results.

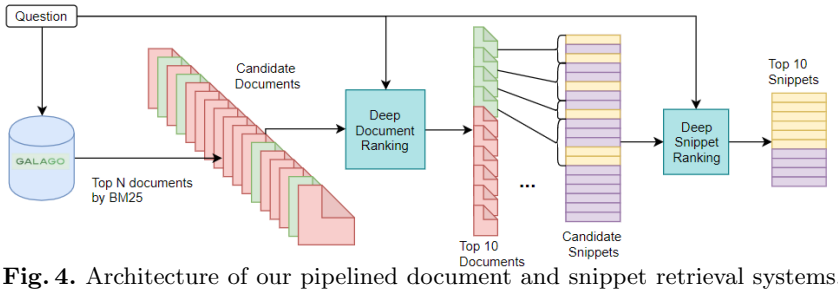

Fig. 4. Architecture of our pipelined document and snippet retrieval systems. The IR engine retrieves candidate relevant documents (left). A neural document retrieval model ranks the retrieved documents and returns the top 10. Then a neural snippet retrieval model ranks the snippets from the 10 documents and returns the top 10 snippets.

Figure 5 illustrates the architecture of our joint document and snippet retrieval models. The same IR engine is used to retrieve N documents. Then a joint model assigns relevance scores to the N documents and their snippets. We return the K_d documents with the highest relevance scores, and the K_s snippets with the highest relevance scores among all the snippets of the K_d documents. We use the same N, K_d, K_s values as in the pipelined models.

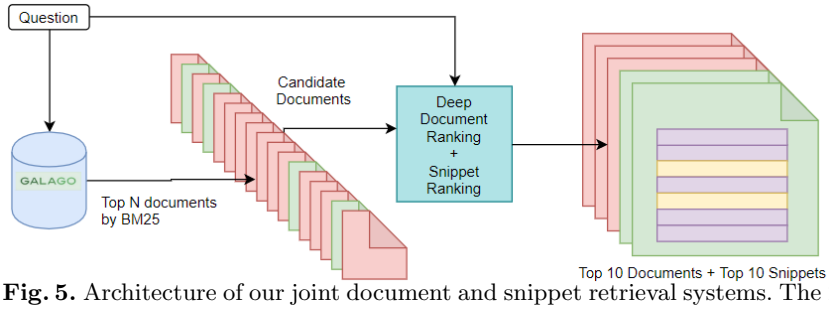

Fig. 5. Architecture of our joint document and snippet retrieval systems. The IR engine retrieves candidate relevant documents. The joint neural model assigns scores to the sentences of the retrieved documents and their snippets. We return the 10 documents with the highest scores, and the 10 snippets of those documents with the highest scores.

6 Experiments

6.1 Data and Experimental Setup

The document collection consists of approx. 29M 'articles' (titles and abstracts) from the 'MEDLINE/PUBMED baseline 2019' collection.[18] We discarded approx. 10M 'articles' that contained only titles, since very few of them had been judged as relevant by the expert annotators for any question. We created an index of the remaining approx. 19M articles using Galago.[19] For indexing purposes, we removed stopwords and applied Krovetz's stemmer [12]. To train the neural models (or to fine-tune the BERT-based ones), we used years 1–6 of the BIOASQ data (2,647 questions), using batch 5 of year 6 as development set (100 questions).

[18] See `https://www.nlm.nih.gov/databases/download/pubmed_medline.html`.
[19] We used Galago version 3.10. Consult `http://www.lemurproject.org/galago.php`.

In TERM-PACRR, BCNN, PDRMM (as sentence ranker), and W2V-JPDRMM, we use the biomedical WORD2VEC embeddings and the pre-computed IDF scores of [17]. We do not update the word embeddings when training these models. Similarly, when training BERT-JPDRMM, we use the same IDF scores and we do not update the BERT instance that provides the wordpiece embeddings; recall that we reconstruct the tokens from the wordpieces and use the first wordpiece embedding for each reconstructed token in BERT-JPDRMM. For tokenization, in methods that do not use wordpieces, we rely on the 'bioclean' tool provided by BIOASQ. Even for models using wordpieces, the features for the final logistic regression layer are derived using 'bioclean' in order to have tokens consistent with the same IDF table used in the other models. A slightly modified version of 'bioclean' is also used when constructing the index of the IR engine. The same 'bioclean' version (plus stopword removal) is also applied to the question when passing it as a query to the IR engine. In snippet retrieval, we use NLTK's English sentence splitter.[20]

TERM-PACRR was trained using default settings from the public release.[21] The BERT based document ranker (Section 2.2) was also trained using default settings from the public release.[22] The only exception was that we used a learning rate of $5e$-6 for fine-tuning, based on development set performance.

For BCNN, we used the publicly available code.[23] BCNN was trained using the settings that won last year's snippet extraction task [3], using Adagrad [6], with learning rate 0.08, and batch size 200. The model was trained for a maximum of 50 epochs. We keep for testing the parameters of BCNN from the epoch with the best snippet Mean Average Precision (MAP) score [16] on the development set. Document and snippet MAP are the official scores for document and snippet retrieval, respectively, in BIOASQ.

We re-implemented PDRMM (as a sentence ranker) in PYTORCH [21] replicating the code of [3]; we also implemented JPDRMM in PYTORCH.[24] We trained PDRMM and W2V-JPDRMM for a maximum of 20 epochs, and BERT-JPDRMM for a maximum of 4 epochs, selecting the parameters from the epoch with the best development snippet MAP for PDRMM and document MAP the two JPDRMM versions. We also applied early stopping and stopped training when development performance (snippet or document MAP) stopped improving for 4 consecutive epochs. For the two JPDRMM versions, one could also monitor *snippet* MAP on development data, instead of document MAP, or a combination of the two. We plan to examine how this affects the performance of JPDRMM in future work. W2V-JPDRMM, BERT-JPDRMM, and PDRMM were trained using Adam [11] with a learning rate of 0.01, $\beta_1/\beta_2 = 0.9/0.999$, and a batch size of 32.

[20] We used NLTK v3.2.4. See `https://www.nltk.org/api/nltk.tokenize.html`.
[21] See `https://github.com/nlpaueb/aueb-bioasq6`.
[22] See `https://github.com/google-research/bert`.
[23] BCNN's code is also available from `https://github.com/nlpaueb/aueb-bioasq6`.
[24] The original code of PDRMM is also available from `https://github.com/nlpaueb/aueb-bioasq6`. All the additional code of this paper will also be made available.

6.2 Official Submissions

We submitted five different systems to BIOASQ 7 (Task 7b, Phase A), all of which consist of components described above.

AUEB-NLP-1: W2V-JPDRMM for both document retrieval and snippet extraction.
AUEB-NLP-2: BERT-JPDRMM for both document retrieval and snippet extraction.
AUEB-NLP-3: pipeline consisting of TERM-PACRR for document retrieval, followed by BCNN for snippet retrieval in batches 2 and 3, or PDRMM in batches 4 and 5.
AUEB-NLP-4: pipeline consisting of BERT for document retrieval, followed by BCNN for snippet retrieval in batches 2 and 3, or PDRMM in batches 4 and 5.
AUEB-NLP-5: pipeline of BERT high confidence (Section 2.2, last paragraph) for document retrieval, followed by BCNN for snippet retrieval in batches 2 and 3, or PDRMM in batches 4 and 5.

In all five systems, after obtaining the top 10 documents and top 10 snippets, we reranked the top 10 snippets by the scores of the documents they came from. The goal was to promote snippets coming from highly relevant documents. In the first two systems, which use JPDRMM versions, this final reranking of the snippets made almost no difference, since JPDRMM internally revises the scores of the snippets taking into account the scores of the documents they come from.

6.3 Results

Table 1 reports our test results (F1, MAP) for batches 2–5 of BIOASQ 7. We did not participate in batch 1. We observe that the BERT document ranker (used in AUEB-NLP-4) has the best document MAP scores in all batches; recall that document MAP is the official document retrieval measure of BIOASQ and also the measure we monitored on the development data to select the best training epoch. The BERT high confidence document ranker (used in AUEB-NLP-5) has the second best document MAP overall, but with greatly improved F1.

Interestingly, the joint model (used in AUEB-NLP-1/2) outperformed comparable pipelined systems (AUEB-NLP-1 vs. AUEB-NLP-3, AUEB-NLP-2 vs. AUEB-NLP-4) by a wide margin in snippet MAP. It obtained very competitive results in snippet MAP even without using BERT embeddings (AUEB-NLP-1) and against pipelines that used BERT for document retrieval (AUEB-NLP-4) and additional reranking heuristics (AUEB-NLP-5). Recall, also, that in the joint model we selected the best training epoch by monitoring the *document* MAP on development data, whereas for the snippet retrieval components of the pipelined models (AUEB-NLP-3/4/5) *snippet* MAP was monitored; hence, the snippet MAP scores of the joint model might improve further by monitoring *snippet* MAP. We also note that the joint models use much fewer trainable parameters than the pipeline models (Table 1); and they outperform AUEB-NLP-3, which was one of the best systems of BIOASQ 6. It is also interesting that in both document and snippet retrieval, there is no clear difference between AUEB-NLP-1, which does not rely on BERT at all, and AUEB-NLP-2, which uses BERT to obtain word embeddings.

It is particularly interesting is that the joint model (AUEB-NLP-1/2) outperforms the BERT based high-confidence model (AUEB-NLP-5). Similarly to [3], we

DOCUMENT RETRIEVAL					SNIPPET RETRIEVAL				
System	**Rank**	**F-M.**	**MAP**	**GMAP**	**System**	**Rank**	**F-M.**	**MAP**	**GMAP**
Batch 2					Batch 2				
AUEB-NLP-1	9	17.84	7.41	0.66	**AUEB-NLP-1**	1	18.55	**14.38**	0.19
AUEB-NLP-2	10	18.23	7.41	0.62	AUEB-NLP-2	3	17.64	12.90	**0.28**
AUEB-NLP-3	5	19.05	7.71	**0.75**	AUEB-NLP-3	6	11.11	6.25	0.13
AUEB-NLP-4	1	19.11	**8.49**	0.67	AUEB-NLP-4	5	10.96	6.43	0.14
AUEB-NLP-5	2	**34.43**	8.30	0.49	AUEB-NLP-5	2	**19.01**	13.62	0.23
Top Comp.	3	18.77	7.91	0.48	Top Comp.	4	12.12	8.93	0.04
Batch 3					Batch 3				
AUEB-NLP-1	4	23.80	10.41	1.18	**AUEB-NLP-1**	1	24.72	**22.06**	0.81
AUEB-NLP-2	2	24.49	11.21	1.56	AUEB-NLP-2	2	**25.63**	21.97	**0.89**
AUEB-NLP-3	6	22.66	9.86	1.04	AUEB-NLP-3	7	14.43	9.90	0.28
AUEB-NLP-4	1	24.71	**11.99**	1.51	AUEB-NLP-4	5	15.44	11.26	0.37
AUEB-NLP-5	3	**40.34**	11.02	**1.64**	AUEB-NLP-5	3	24.56	19.21	0.85
Top Comp.	5	28.94	10.33	0.18	Top Comp.	4	16.17	14.04	0.09
Batch 4					Batch 4				
AUEB-NLP-1	4	20.56	9.51	1.01	AUEB-NLP-1	2	24.40	20.86	0.65
AUEB-NLP-2	3	20.51	9.68	0.83	**AUEB-NLP-2**	1	23.65	**21.14**	0.75
AUEB-NLP-3	5	19.42	9.09	0.83	AUEB-NLP-3	7	17.79	11.49	0.53
AUEB-NLP-4	1	21.48	**10.34**	1.12	AUEB-NLP-4	9	17.91	11.16	0.56
AUEB-NLP-5	2	**37.83**	10.15	**1.16**	AUEB-NLP-5	3	**24.67**	18.21	**0.98**
Top Comp.	6	18.53	8.35	0.51	Top Comp.	4	17.23	15.27	0.13
Batch 5					Batch 5				
AUEB-NLP-1	3	9.90	3.68	0.06	AUEB-NLP-1	3	8.04	5.81	0.02
AUEB-NLP-2	6	9.00	3.55	0.06	**AUEB-NLP-2**	1	8.18	**6.31**	**0.03**
AUEB-NLP-3	5	9.91	3.66	0.07	AUEB-NLP-3	8	5.81	3.87	0.02
AUEB-NLP-4	1	11.20	**4.25**	**0.10**	AUEB-NLP-4	6	6.53	4.16	0.02
AUEB-NLP-5	2	**20.12**	3.99	0.08	AUEB-NLP-5	2	**9.89**	6.17	**0.03**
Top Comp.	4	9.27	3.68	0.05	Top Comp.	4	6.56	4.99	0.01

Table 1. Performance on BIOASQ Task 7b, Phase A (batches 2–5) for document and snippet retrieval. Top Comp. is the top scoring submission from other teams.

observed that passing only high-confidence retrieved documents to the snippet ranking component in pipeline systems improved snippet retrieval greatly (compare the snippet scores of AUEB-NLP-4 vs. AUEB-NLP-5), because it allowed the snippet retrieval component to operate only on documents that were likely to be relevant. However, JPDRMM did not require such heuristics. Instead, since it models the fact that good snippets come from good documents and vice-versa, it naturally selected snippets mostly from high confidence documents. Thus the empirical results validate the hypothesis that joint modeling is beneficial. An open question is why the joint models do worse on document ranking compared to the pipelined models (AUEB-NLP-4/5). This is likely due to BERT (the document scorer of AUEB-NLP-4/5) being such a powerful model. A future line of investigation is to build joint models that integrate BERT to a larger extent, instead of just providing word embeddings to JPDRMM as in BERT-JPDRMM.

Model	Number of Parameters
AUEB-NLP-1	5,793
AUEB-NLP-2	3,541,551
AUEB-NLP-3	16,519
AUEB-NLP-4/5 (with BCNN for snippets)	109,499,902
AUEB-NLP-4/5 (with PDRMM for snippets)	109,489,455

Table 2. Number of trainable parameters for systems submitted.

7 Related Work

7.1 Document Retrieval

Neural document ranking models [7, 9, 10, 17, 18] have only recently managed to improve upon the rankings of traditional IR systems (e.g., rankings based on BM25). See also [14] for caveats.

PACRR [9] uses a matrix containing the cosine similarities between each query term embedding and each document term embedding; the multiple similarity matrices of PDRMM [17] (Section 3.2, Fig. 3.2) are an extension of PACRR's similarity matrix. PACRR applies convolutions with multiple filters of kernel size 2 and 3 to its similarity matrix to capture bigram and trigram matches, respectively; PDRMM skips these convolutions, since one of its similarity matrices already contains similarities between context-aware embeddings. PACRR then employs max-pooling (over the outputs of kernels of the same size) followed by row-wise k-max pooling to obtain the k-best unigram, bigram, and trigram matches between each query term and the entire document, producing $3k$ document-aware features per query term; these pooling operations are again very similar to the ones of PDRMM. The IDF score of each query term is then appended to its $3k$ features, and the features of all the query terms are then concatenated into a single vector, which is passed to an MLP that produces the relevance score of the document. The only difference between PACRR and TERM-PACRR [3, 17] (Section 2.1) is that the latter passes the features of each query term separately to the MLP, obtaining a separate relevance score per query term, and then uses a linear layer to combine the relevance scores.[25] By contrast, PDRMM computes a weighted sum of the feature vectors of the query terms (weighted by their importance scores, bottom right of Fig. 1) and passes the weighted sum to the MLP that produces the document's relevance score.

TERM-PACRR's final layers, which apply an MLP separately to document-aware features of each query term and then combine the resulting relevance scores of the query terms using a linear layer, are very similar to the corresponding layers of DRMM [7]. In DRMM, however, the document-aware features of each query term represent a histogram of (frequencies of buckets of) the cosine similarities between the embedding of the query term and all the terms of the document. These histogram representations are non-differentiable, hindering the end-to-end training of the model via backpropagation. By contrast, all the models used in our work are fully differentiable, following [17].

[25] We note again that TERM-PACRR is called PACRR-DRMM in [17].

The document retrieval model of Zhu et al. [33] was designed to handle medical questions. It uses a BIGRU with self-attention [4, 2] to produce a single query embedding from the query's word embeddings; and a hierarchical BIGRU [31] to produce a single document embedding. The word-level BIGRU of the hierarchical BIGRU reads the word embeddings of a single sentence of the document at a time, turning each sentence into a sentence embedding; it also employs a cross-attention mechanism between the word embeddings of the query and those of the sentence. The sentence-level BIGRU reads the sentence embeddings and produces the document embedding using a self-attention mechanism. The relevance score of the document is then computed by taking the element-wise product of the query and document embeddings and feeding it to an MLP. Although we hope to compare to the model of Zhu et al. in future work, we note that their experiments were conducted on a dataset much smaller than BIOASQ's, containing only 7.5k documents and 7.5k queries with only one relevant document per query. Furthermore, the documents of Zhu et al.'s dataset were article sections from a healthcare portal and the queries were produced by annotators looking at a particular article. The annotators were not biomedical experts, hence their queries and terminology were much simpler compared to BIOASQ's, where the annotators are biomedical experts and queries reflect real needs.

BERT based models have recently been explored for document ranking. Most approaches train shallow task-specific layers on top of BERT [30, 20], much as in our BERT based document retrieval model (Section 2.2, Fig. 1). MacAvaney et al. [15] explored ways to combine ELMO [22] and BERT [5] with complex neural IR models such as DRMM [7] and PACRR [9]. It would be interesting to explore similar ways to improve BERT-JPDRMM (Section 4.1), e.g., by using cosine similarity matrices (Fig. 2) computed on wordpiece embeddings coming from different layers of BERT, or by concatenating the embedding of BERT's [CLS] token (Fig. 1) with the extra document features in the final layers of JPDRMM (Fig. 3).

7.2 Snippet Extraction

BCNN (Section 3.1) is one of the several CNN-based models explored by Yin et al. [32]. We used BCNN in our pipeline systems (Section 6.2), because it had the best snippet retrieval results in BIOASQ 6 [3]. As we demonstrated with PDRMM, however, neural document retrieval models can also be used to rank snippets, and in our experiments PDRMM performed better than BCNN for snippet retrieval, which is why it replaced BCNN in our pipeline systems in batches 4 and 5.

Amiri et al. [1] use context-sensitive autoencoders to create question and sentence vectors. They compute the cosine similarity between the question and sentence vectors and rank the sentences in the dataset. They experiment on three datasets, including TREQ QA [27], which includes biomedical data. Their method is unsupervised and performed competitively compared to former state-of-the-art supervised models. However, it does not take into account the relevance of the documents when ranking sentences.

Other neural models have also been proposed for snippet extraction in biomedical question answering. Wang et al. [28] use a stacked BILSTM that reads the

concatenation of the question and a candidate sentence. In each timestep, the model produces a relevance score for the sentence, taking into account the tokens read so far. Then a mean pooling operation extracts the final relevance score of the sentence. Wang et al. combine the relevance score of the neural model with a keyword matching score, in order to distinguish tokens with similar embeddings and to favor exact token matches. In PDRMM and JPDRMM, this effect is achieved by external overlap features in the final linear layer, but also by including in the neural model a similarity matrix (view) with one-hot token embeddings (Fig. 2). As in the model of Amiri et al., discussed above, the model of Wang et al. does not take into account the relevance of the documents when ranking sentences.

8 Discussion and Future Work

We presented the models, experiments, and results of the submissions of AUEB for the document and snippet retrieval tasks of BIOASQ 7. Our systems obtained the best document and snippet retrieval results in the four batches we participated in.

We introduced a new jointly trained model for document and snippet retrieval. The joint model outperformed comparable pipelined architectures by a wide margin in snippet retrieval. It obtained very competitive results in snippet retrieval even without using BERT at all, and against pipelines that used BERT for document retrieval and additional reranking heuristics. On the other hand, a BERT based document ranker performed better at the document retrieval level than the joint model. We aim to investigate if tuning the weights of the document and snippet losses of the joint model could help it perform better in document retrieval too. We also aim to integrate more tightly BERT into our joint model, e.g., by using similarity matrices based on embeddings coming from different levels of BERT, instead of using only the top-level BERT embeddings (as in one version of our joint model), and by adding the embedding of BERT's [CLS] token to the extra features of the joint model. Finally, we aim to extend the joint model to also perform exact answer extraction (part of BIOASQ Task 7b, Phase B).

References

1. Amiri, H., Resnik, P., Boyd-Graber, J., Daumé III, H.: Learning text pair similarity with context-sensitive autoencoders. In: Proc. of the 54th Annual Meeting of the Association for Computational Linguistics (Vol. 1: Long Papers). pp. 1882–1892. Berlin, Germany (2016)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: 3rd International Conf. on Learning Representations. San Diego, California (2015)
3. Brokos, G., Liosis, P., McDonald, R., Pappas, D., Androutsopoulos, I.: AUEB at BioASQ 6: Document and Snippet Retrieval. In: Proc. of the 6th BioASQ Workshop A challenge on large-scale biomedical semantic indexing and question answering. Brussels, Belgium (2018)

4. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proc. of the Conf. on Empirical Methods in Natural Language Processing. pp. 1724–1734. Doha, Qatar (2014)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 (2018)
6. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. J. of Mach. Learn. Res. **12**, 2121–2159 (07 2011)
7. Guo, J., Fan, Y., Ai, Q., Croft, W.B.: A Deep Relevance Matching Model for Ad-hoc Retrieval. In: Proc. of the 25th ACM International on Conf. on Information and Knowledge Management. pp. 55–64. Indianapolis, Indiana, USA (2016)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: IEEE Conf. on Comp. Vision and Pattern Recog. pp. 770–778 (2016)
9. Hui, K., Yates, A., Berberich, K., de Melo, G.: PACRR: A position-aware neural IR model for relevance matching. In: Proc. of the Conf. on Empirical Methods in Natural Language Processing. pp. 1049–1058. Copenhagen, Denmark (2017)
10. Hui, K., Yates, A., Berberich, K., de Melo, G.: Co-PACRR: A context-aware neural IR model for ad-hoc retrieval. In: Proc. of the 11th ACM International Conf. on Web Search and Data Mining. pp. 279–287. Marina Del Rey, CA (2018)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2015)
12. Krovetz, R.: Viewing morphology as an inference process. In: Proc. of the 16th Annual International ACM SIGIR Conf. on Research and Development in Information Retrieval. pp. 191–202. Pittsburgh, PA (1993)
13. Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C.H., Kang, J.: Biobert: a pre-trained biomedical language representation model for biomedical text mining. arXiv preprint arXiv:1901.08746 (2019)
14. Lin, J.: The neural hype and comparisons against weak baselines. SIGIR Forum **52**(2), 40–51 (2019)
15. MacAvaney, S., Yates, A., Cohan, A., Goharian, N.: Cedr: Contextualized embeddings for document ranking. CoRR **abs/1904.07094** (2019)
16. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
17. McDonald, R., Brokos, G.I., Androutsopoulos, I.: Deep Relevance Ranking Using Enhanced Document-Query Interactions. In: Proc. of the Conf. on Empirical Methods in Natural Language Processing. Brussels, Belgium (2018)
18. Mitra, B., Craswell, N.: An Introduction to Neural Information Retrieval. Now Publishers (2018)
19. Mohan, S., Fiorini, N., Kim, S., Lu, Z.: Deep learning for biomedical ir: Learning textual relevance from click logs. In: BioNLP 2017. pp. 222–231 (2017)
20. Nogueira, R., Cho, K.: Passage re-ranking with BERT. CoRR **abs/1901.04085** (2019)
21. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. In: NIPS-W (2017)
22. Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proc. of the 2018 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1. pp. 2227–2237. New Orleans, Louisiana (2018)
23. Qiao, Y., Xiong, C., Liu, Z.H., Liu, Z.: Understanding the behaviors of BERT in ranking. CoRR **abs/1904.07531** (2019)

24. Robertson, S., Zaragoza, H.: The probabilistic relevance framework: BM25 and beyond. Foundations and Trends in Information Retrieval **3**(4), 333–389 (2009)
25. Severyn, A., Moschitti, A.: Learning to rank short text pairs with convolutional deep neural networks. In: Proc. of the 38th international ACM SIGIR Conf. on Research and Development in Information Retrieval. pp. 373–382. ACM (2015)
26. Tsatsaronis, G., Balikas, G., Malakasiotis, P., Partalas, I., Zschunke, M., Alvers, M., Weissenborn, D., Krithara, A., Petridis, S., Polychronopoulos, D., Almirantis, Y., Pavlopoulos, J., Baskiotis, N., Gallinari, P., Artieres, T., Ngonga, A., Heino, N., Gaussier, E., Barrio-Alvers, L., Schroeder, M., Androutsopoulos, I., Paliouras, G.: An overview of the BioASQ Large-Scale Biomedical Semantic Indexing and Question Answering Competition. BMC Bioinformatics **16**(138) (2015)
27. Voorhees, E.M.: Question answering in trec. In: Proceedings of the Tenth International Conference on Information and Knowledge Management. pp. 535–537. New York, NY, USA (2001)
28. Wang, D., Nyberg, E.: A long short-term memory model for answer sentence selection in question answering. In: Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conf. on Natural Language Processing (Volume 2: Short Papers). pp. 707–712. Beijing, China (2015)
29. Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G.S., Hughes, M., Dean, J.: Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. CoRR **abs/1609.08144** (2016)
30. Yang, W., Zhang, H., Lin, J.: Simple applications of bert for ad hoc document retrieval. CoRR **abs/1903.10972** (2019)
31. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical Attention Networks for Document Classification. In: Proc. of the 2016 Conf. of the NA Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 1480–1489 (2016)
32. Yin, W., Schütze, H., Xiang, B., Zhou, B.: ABCNN: Attention-based convolutional neural network for modeling sentence pairs. Transactions of the Association for Computational Linguistics **4** (2016)
33. Zhu, M., Ahuja, A., Wei, W., Reddy, C.K.: A hierarchical attention retrieval model for healthcare question answering. In: The World Wide Web Conf. pp. 2472–2482. San Francisco, CA, USA (2019)