

AUEB at BioASQ 6: Document and Snippet Retrieval

Georgios-Ioannis Brokos¹, Polyvios Liosis¹, Ryan McDonald^{1,3},
Dimitris Pappas^{1,2} and Ion Androutsopoulos¹

¹Dept. of Informatics, Athens University of Economics and Business, Greece

²Institute for Language and Speech Processing, Research Center ‘Athena’, Greece

³Google AI

Abstract

We present AUEB’s submissions to the BioASQ 6 document and snippet retrieval tasks (parts of Task 6b, Phase A). Our models use novel extensions to deep learning architectures that operate solely over the text of the query and candidate document/snippets. Our systems scored at the top or near the top for all batches of the challenge, highlighting the effectiveness of deep learning for these tasks.

1 Introduction

BioASQ (Tsatsaronis et al., 2015) is a biomedical document classification, document retrieval, and question answering competition, currently in its sixth year.¹ We provide an overview of AUEB’s submissions to the document and snippet retrieval tasks (parts of Task 6b, Phase A) of BioASQ 6.² In these tasks, systems are provided with English biomedical questions and are required to retrieve relevant documents and document snippets from a collection of MEDLINE/PubMed articles.³

We used deep learning models for both document and snippet retrieval. For document retrieval, we focus on extensions to the Position-Aware Convolutional Recurrent Relevance (PACRR) model of Hui et al. (2017) and, mostly, the Deep Relevance Matching Model (DRMM) of Guo et al. (2016), whereas for snippet retrieval we based our work on the Basic Bi-CNN (BCNN) model of Yin et al. (2016). Little task-specific pre-processing is employed and the models operate solely over the text of the query and candidate document/snippets.

Overall, our systems scored at the top or near the top for all batches of the challenge. In previous

years of the BioASQ challenge, the top scoring systems used primarily traditional IR techniques (Jin et al., 2017). Thus, our work highlights that end-to-end deep learning models are an effective approach for retrieval in the biomedical domain.

2 Document Retrieval

For document retrieval, we investigate new deep learning architectures focusing on *term-based interaction models*, where query terms (*q-terms* for brevity) are scored relative to a document’s terms (*d-terms*) and their scores are aggregated to produce a relevance score for the document. All models use pre-trained embeddings for all q-terms and d-terms. Details on data resources and data pre-processing are given in Section 5.1.

2.1 PACRR-based Models

The first model we investigate is PACRR (Hui et al., 2017). In this model, a query-document term similarity matrix *sim* is first computed (Fig. 1, left). Each cell (i, j) of *sim* contains the cosine similarity between the embeddings of a q-term q_i and a d-term d_j . To keep the dimensions $l_q \times l_d$ of *sim* fixed across queries and documents of varying lengths, queries are padded to the maximum number of q-terms l_q , and only the first l_d terms per document are retained.⁴ Then, convolutions of different kernel sizes $n \times n$ ($n = 2, \dots, l_g$) are applied to *sim* to capture *n*-gram query-document similarities. For each size $n \times n$, multiple kernels (filters) are used. Max pooling is then applied along the dimension of the filters (max value of all filters of the same size), followed by *k*-max pooling along the dimension of d-terms to capture the strongest *k* signals between each q-term and all the d-terms. The resulting matrices (one per kernel

¹Consult <http://bioasq.org/>.

²For further information on the BioASQ 6 tasks, see <http://bioasq.org/participate/challenges>.

³<http://www.ncbi.nlm.nih.gov/pubmed/>.

⁴We use PACRR-*firstk*, which Hui et al. (2017) recommend when documents fit in memory, as in our experiments.

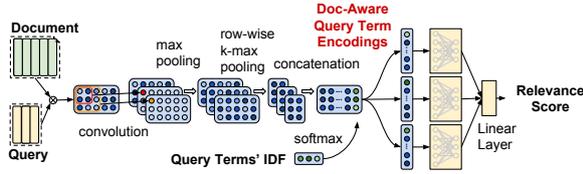


Figure 1: PACRR (Hui et al., 2017) and TERM-PACRR. In PACRR, an MLP is applied to the concatenation of the document-aware q-term encodings to produce the relevance score. In TERM-PACRR, the MLP is applied *separately* to each document-aware q-term encoding; the resulting scores are combined by a linear layer.

size) are concatenated into a single matrix where each row is a document-aware q-term encoding (Fig. 1); the IDF of the q-term is also appended, normalized by applying a softmax across the IDFs of all the q-terms. Following Hui et al. (2018), we concatenate the rows of the resulting matrix into a single vector, which is passed to an MLP that produces a query-document relevance score.⁵

Instead of using an MLP to score the *concatenation* of all the (document-aware) q-term encodings, a simple extension we found effective was to use an MLP to *independently* score each q-term encoding (the same MLP for all q-terms, Fig. 1); the resulting scores are aggregated via a linear layer. This version, TERM-PACRR, performs better than PACRR, using the same number of hidden layers in the MLPs. Likely this is due to the fewer parameters of TERM-PACRR’s MLP, which is shared across the q-term representations and operates on shorter input vectors. Indeed, in our early experiments TERM-PACRR was less prone to overfitting.⁶

2.2 DRMM-based Models

The second model we investigate is DRMM (Guo et al., 2016) (Fig. 2). The original DRMM uses pre-trained word embeddings for q-terms and d-terms, and (bucketed) cosine similarity histograms (outputs of \otimes nodes in Fig. 2). Each histogram captures the similarity of a q-term to all the d-terms of a particular document. The histograms, which in this model are the document-aware q-term encodings, are fed to an MLP (dense layers of Fig. 2) that produces the (document-aware) score of each q-term. Each q-term score is then weighted using

⁵Hui et al. (2017) used an additional LSTM, which was later replaced by the final concatenation (Hui et al., 2018).

⁶In the related publication of McDonald et al. (2018) TERM-PACRR is identical to the PACRR-DRMM model.

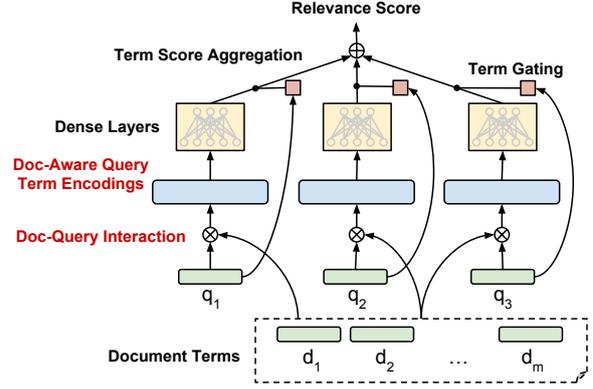


Figure 2: Illustration of DRMM (Guo et al., 2016) for three q-terms and m d-terms. The \otimes nodes produce (bucketed) cosine similarity histograms, each capturing the similarity between a q-term and all the d-terms.

a gating mechanism (topmost box nodes in Fig. 2) that examines properties of the q-term to assess its importance for ranking (e.g., common words are less important). The sum of the weighted q-term scores is the relevance score of the document.

For gating (topmost box nodes of Fig. 2), Guo et al. (2016) use a linear self-attention:

$$g_i = \text{softmax}\left(w_g^T \phi_g(q_i); q_1, \dots, q_n\right)$$

$\phi_g(q_i)$ is the embedding $e(q_i)$ of the i -th q-term, or its IDF, $\text{idf}(q_i)$; w_g is a weights vector. We found that $\phi_g(q_i) = [e(q_i); \text{idf}(q_i)]$, where ‘;’ is concatenation, was optimal for all DRMM-based models.

2.2.1 ABEL-DRMM

The original DRMM (Guo et al., 2016) has two shortcomings. The first one is that it ignores entirely the contexts where the terms occur, in contrast to position-aware models such as PACRR (Section 2.1) or those based on recurrent representations (Palangi et al., 2016). Secondly, the histogram representation for document-aware q-term encodings is not differentiable, so it is not possible to train the network end-to-end, if one wished to backpropagate all the way to word embeddings.

To address the first shortcoming, we add an encoder (Fig. 3) to produce the *context-sensitive encoding* of each q-term or d-term from the pre-trained embeddings of the previous, current, and next term in a particular query or document. A single dense layer with residuals is used, in effect a one-layer Temporal Convolutional Network (TCN) (Bai et al., 2018) without pooling or dilation. The number of convolutional filters equals the dimen-

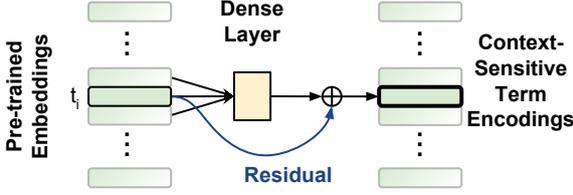


Figure 3: Producing *context-sensitive term encodings*.

sions of the pre-trained embedding, for residuals to be summed without transformation.

Specifically, let $e(t_i)$ be the pre-trained embedding for a q-term or d-term term t_i . We compute the context-sensitive encoding of t_i as:

$$c(t_i) = \varphi(W_c \phi_c(t_i) + b_c) + e(t_i) \quad (1)$$

W_c and b_c are the weights matrix and bias vector of the dense layer, φ is the activation function, $\phi_c(t_i) = [e(t_{i-1}); e(t_i); e(t_{i+1})]$, t_{i-1}, t_{i+1} are the tokens surrounding t_i in the query or document. This is an orthogonal way to incorporate context into the model relative to PACRR. PACRR creates a query-document similarity matrix and computes n -gram convolutions over the matrix. Here we incorporate context directly into the term encodings; hence similarities in this space are already context-sensitive. One way to view this difference is the point at which context enters the model – directly during term encoding (Fig. 3) or after term similarity scores have been computed (PACRR, Fig. 1).

To make DRMM trainable end-to-end, we replace its histogram-based document-aware q-term encodings (\otimes nodes of Fig. 2) by q-term encodings that consider d-terms via an attention-mechanism. Figure 4 shows the new sub-network that computes the document-aware encoding of a q-term q_i , given a document $d = \langle d_1, \dots, d_m \rangle$ of m d-terms. We first compute a dot-product attention score $a_{i,j}$ for each d_j relative to q_i :

$$a_{i,j} = \text{softmax}(c(q_i)^T c(d_j); d_1, \dots, d_m) \quad (2)$$

where $c(t)$ is the context-sensitive encoding of t (Eq. 1). We then sum the context-sensitive encodings of the d-terms, weighted by their attention scores, to produce an attention-based representation d_{q_i} of document d from the viewpoint of q_i :

$$d_{q_i} = \sum_j a_{i,j} c(d_j) \quad (3)$$

The Hadamard product (element-wise multiplication, \odot) between the document representation d_{q_i}

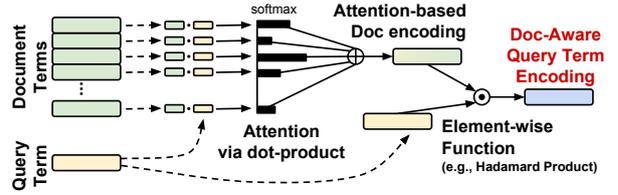


Figure 4: ABEL-DRMM sub-net. From context-aware q-term and d-term encodings (Fig. 3), it generates fixed-dimension *document-aware q-term encodings* to be used in DRMM (Fig. 2, replacing \otimes nodes).

and the q-term encoding $c(q_i)$ is then computed and used as the fixed-dimension document-aware encoding $\phi_H(q_i)$ of q_i (Fig. 4):

$$\phi_H(q_i) = d_{q_i} \odot c(q_i) \quad (4)$$

The \otimes nodes and lower parts of the DRMM network of Fig. 2 are now replaced by (multiple copies of) the sub-network of Fig. 4 (one copy per q-term), with the \odot nodes replacing the \otimes nodes. We call the resulting model Attention-Based Element-wise DRMM (ABEL-DRMM).

Intuitively, if the document contains one or more terms d_j that are similar to q_i , the attention mechanism will have emphasized mostly those terms and, hence, d_{q_i} will be similar to $c(q_i)$, otherwise not. This similarity could have been measured by the cosine similarity between d_{q_i} and $c(q_i)$, but the cosine similarity assigns the same weight to all the dimensions, i.e., to all the element-wise products in $\phi_H(q_i)$. By using the Hadamard product, we pass on to the upper layers of DRMM (the dense layers of Fig. 2), which score each q-term with respect to the document, all the element-wise products of $\phi_H(q_i)$, allowing the upper layers to learn which element-wise products (or combinations of them) are important when matching a q-term to the document.

2.2.2 ABEL-DRMM extensions

We experimented with two extensions to ABEL-DRMM. The first is a *density-based extension* that considers all the windows of l_w consecutive tokens of the document and computes the ABEL-DRMM relevance score per window. The final relevance score of a document is the sum of the original ABEL-DRMM score computed over the entire document plus the maximum ABEL-DRMM score over all the document's windows. The intuition is to reward not only documents that match the query, but also those that match it in a dense window.

The second extension is to compute a *confidence* score per document and only return documents with scores above a threshold. We apply a softmax over the ABEL-DRMM scores of the top t_d documents and return only documents from the top t_d with normalized scores exceeding a threshold t_c . While this will always hurt metrics like Mean Average Precision (MAP) when evaluating document retrieval, it has the potential to improve the precision of downstream components, in our case snippet retrieval, which in fact we observe.

3 Snippet Retrieval

For the snippet retrieval task, we used the ‘basic CNN’ (BCNN) network of the broader ABCNN model (Yin et al., 2016), which we combined with a post-processing stage, as discussed below. The input of snippet retrieval is an English question and text snippets (e.g., sentences) from documents that the document retrieval component returned as relevant to the question. The goal is to rank the snippets, so that snippets that human experts selected as relevant to the question will be ranked higher than others. In BioASQ, human experts are instructed to select relevant snippets consisting of one or more consecutive sentences.⁷ For training purposes, we split the relevant documents into sentences, and consider sentences that overlap the gold snippets (the ones selected by the human experts) as relevant snippets, and the remaining ones as irrelevant. At inference time, documents returned by the document retrieval model as relevant are split into sentences, and these sentences are ranked by the system. For details on sentence splitting, tokenization, etc., see Section 5.1.

3.1 BCNN Model

BCNN receives as input two sequences of terms (tokens), in our case a question (query) and a sentence from a document. All terms are represented by pre-trained embeddings (Section 5.1). Snippet sequences were truncated (or zero padded) to be of uniform length. A convolution layer with multiple filters, each of the same width w , is applied to each one of the two input sequences, followed by a windowed-average pooling layer over the same filter width to produce a feature map (per filter) of the same dimensionality as the input to the con-

⁷This was not actually the case in BioASQ year 1. Hence, some of our training data do not adhere to this rule.

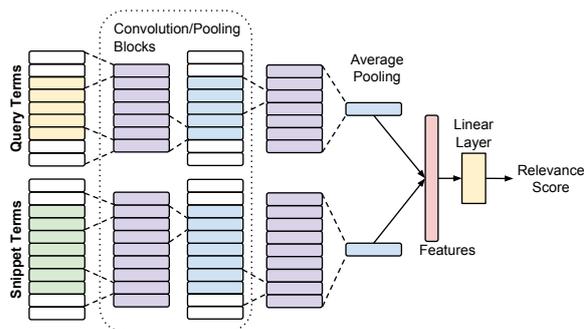


Figure 5: BCNN (Yin et al., 2016) scoring snippets relative to a query. The example illustrates a query of 5 terms, a snippet of 7 terms, and a single convolution filter of width $w = 3$. Zero-padding shown as empty boxes. In each convolution/pooling block, the convolution layer is followed by a windowed-average pooling of the same width w to preserve the dimensionality of the input to the block. Thus convolution/pooling blocks can be repeated, making the model arbitrarily deep.

volution layer.⁸ Consequently, we can stack an arbitrary number of convolution/pooling blocks in order to extract increasingly abstract features.

An average pooling layer is then applied to the entire output of the last convolution/pooling block (Fig. 5) to obtain a feature vector of the query and snippet, respectively. When multiple convolution filters are used (Fig. 5 illustrates only one), we obtain a different feature vector from each filter (for the query and snippet, respectively), and the feature vectors from the different filters are concatenated, again obtaining a single feature vector for the query and snippet, respectively. Similarity scores are then computed from the query and snippet feature vectors, and these are fed into a linear logistic regression layer. One critical implementation detail from the original BCNN paper is that when computing the query-snippet similarity scores, average pooling is actually applied to the output of each one of the convolution/pooling blocks, i.e., we obtain a different query and snippet feature vector from the output of each block. Different similarity scores are computed based on the query and snippet feature vectors obtained from the output of each block, and all the similarity scores are passed to the final layer. Thus the number of inputs to the final layer is proportional to the number of blocks.

⁸The same filters are applied to both queries and snippets.

3.2 Post-processing

A technique that seems to improve our results in snippet retrieval is to retain only the top K_s snippets with the best BCNN scores for each query, and then re-rank the K_s snippets by the relevance scores of the documents they came from; if two snippets came from the same document, they are subsequently ranked by their BCNN score. This is a proxy for more sophisticated models that would jointly consider document and snippet retrieval. This is important as the snippet retrieval model is trained under the condition that it only sees relevant documents. So accounting for the rank/score of the document itself helps to correctly bias the snippet model.

4 Overall System Architecture

Figure 6 outlines the general architecture that we used to piece together the various components. It consists of retrieving the top N documents per query using BM25 (Robertson et al., 1995); re-ranking the top N documents using one of the document retrieval models (Section 2) and retaining (up to) the top K_d documents; scoring all candidate snippets of the top K_d documents via a snippet retrieval model (BCNN, Section 3.1) and retaining (up to) the top K_s snippets; re-ranking the K_s snippets by the relevance scores of the documents they came from (Section 3.2).⁹

We set $K_d = K_s = 10$ as it was dictated by the BioASQ challenge. We set $N = 100$ as we found that with this value, BM25 returned the majority of the relevant documents from the training/development data sets. Setting N to larger values had no impact on the final results. The reason for using a pre-retrieval model based on BM25 is that the deep document retrieval models we use here are computationally expensive. Thus, running them on every document in the index for every query is prohibitive, whereas running them on the top $N = 100$ documents from a pre-retrieval system is easily achieved.

5 Experiments

All retrieval components (PACRR-, DRMM-, BCNN-based) were augmented to combine the scores of the corresponding deep model with a number of traditional IR features, which is a common technique (Severyn and Moschitti, 2015). In

⁹The last step was used only in batches 3–5.

TERM-PACRR, the additional features are fed to the linear layer that combines the q-term scores (Fig. 1). In ABEL-DRMM, an additional linear layer is used that concatenates the deep learning document relevance score with the traditional IR features. In BCNN, the additional features are included in the final linear layer (Fig. 5). The additional features we used were the BM25 score of the document (the document the snippet came from, in snippet retrieval), word overlap (binary and IDF weighted) between the query and the document or snippet; bigram overlap between the query and the document or snippet. The latter features were taken from Mohan et al. (2017). The additional features improved the performance of all models.

5.1 Data Resources and Pre-processing

The document collection consists of approx. 28M ‘articles’ (titles and abstracts only) from the ‘MEDLINE/PubMed Baseline 2018’ collection.¹⁰ We discarded the approx. 10M articles that contained only titles, since very few of these were annotated as relevant. For the remaining 18M articles, a document was the concatenation of each title and abstract. These documents were then indexed using Galago, removing stop words and applying Krovetz’s stemmer (Krovetz, 1993).¹¹ This served as our pre-retrieval model.

Word embeddings were pre-trained by applying word2vec (Mikolov et al., 2013) to the 28M ‘articles’ of the MEDLINE/PubMed collection. IDF values were computed over the 18M articles that contained both titles and abstracts. We used the GenSim implementation of word2vec (skip-gram model), with negative sampling, window size set to 5, default other hyper-parameter values, to produce word embeddings of 200 dimensions.¹² The word embeddings were not updated when training the document relevance ranking models. For tokenization, we used the ‘bioclean’ tool provided by BioASQ.¹³ In snippet retrieval, we used NLTK’s

¹⁰Available from https://www.nlm.nih.gov/databases/download/pubmed_medline.html.

¹¹We used Galago version 3.10. Consult <http://www.lemurproject.org/galago.php>.

¹²Consult <https://radimrehurek.com/gensim/models/word2vec.html>. We used Gensim v. 3.3.0. The word embeddings and code of our experiments are available at <https://github.com/nlpaueb/aueb-bioasq6>.

¹³The tool accompanies an older set of embeddings provided by BioASQ. See <http://participants-area.bioasq.org/tools/BioASQword2vec/>.

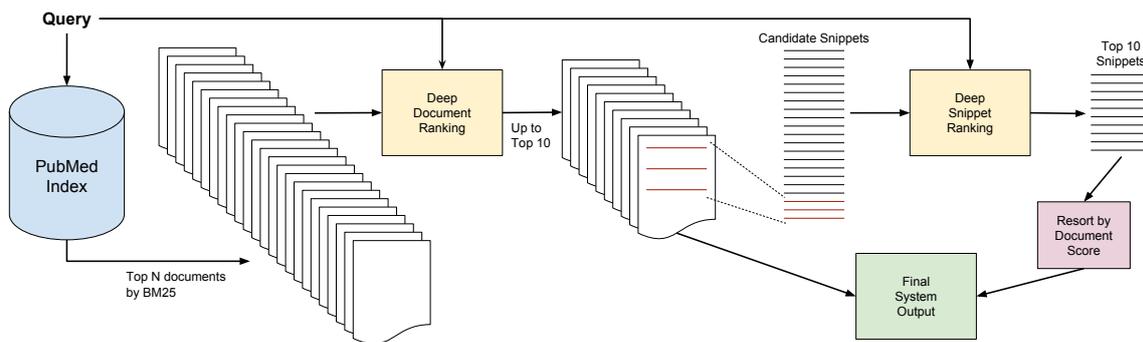


Figure 6: Overall architecture of document and snippet retrieval systems.

English sentence splitter.¹⁴

To train and tune the models we used years 1–5 of the BioASQ data, using batch 5 of year 5 as development for the final submitted models, specifically when selecting optimal model epoch. We report test results (F1, MAP, GMAP) on batches 1–5 of year 6 from the official results table.¹⁵ Details on the three evaluation metrics are provided by Tsatsaronis et al. (2015). They are standard, with the exception that MAP here always assumes 10 relevant documents/snippets, which is the maximum number of documents/snippets the participating systems were allowed to return per query.

5.2 Hyperparameters

All DRMM-based models were trained with Adam (Kingma and Ba, 2014) with a learning rate of 0.01 and $\beta_1/\beta_2 = 0.9/0.999$. Batch sizes were set to 32. We used a hinge-loss with a margin of 1.0 over pairs of a single positive and a single negative document of the same query. All models used a two-layer MLP to score q-terms (dense layers of Fig. 2), with leaky-RELU activation functions and 8 dimensions per hidden layer. For context-sensitive term encodings (Fig. 3), a single layer was used, again with leaky-RELU as activation. For the density-based extension of ABEL-DRMM (Section 2.2.2), $l_w = 20$. For the confidence extension of ABEL-DRMM, $t_d = 100$, $t_c = 0.01$.

TERM-PACRR was also trained with Adam, with a learning rate of 0.001 and $\beta_1/\beta_2 = 0.9/0.999$ with batch size equal to 32. Following Hui et al. (2018), we used binary log-loss over pairs of a single positive and a single negative document of the

same query. Maximum query length l_q was set to 30 and maximum document length l_d was set to 300. Maximum kernel size ($l_g \times l_g$) was set to (3×3) with 16 filters per size. Row-wise k -max pooling used $k = 2$. TERM-PACRR used a two-layer MLP with RELU activations and hidden layers with 7 dimensions to independently score each document-aware query-term encoding.

BCNN was trained using binary log-loss and AdaGrad (Duchi et al., 2011), with a learning rate of 0.08 and $L2$ regularization with $\lambda = 0.0004$. We used 50 convolution kernels (filters) of width $w = 4$ in each convolution layer, and two convolution/pooling blocks. Finally, batch sizes were set to 200. Snippets were truncated to 40 tokens. Questions were never truncated.

5.3 Official Submissions

We submitted 5 different systems to the BioASQ challenge, all of which consist of components described above.

- **AUEB-NLP-1:** Combo of 10 runs of TERM-PACRR for document retrieval (§2.1) followed by BCNN for snippet retrieval (§3).
- **AUEB-NLP-2:** Combo of 10 runs of ABEL-DRMM (§2.2) for document retrieval followed by BCNN for snippet retrieval.
- **AUEB-NLP-3:** Combo of 10 runs of TERM-PACRR and 10 runs of ABEL-DRMM followed by BCNN for snippet retrieval.
- **AUEB-NLP-4:** ABEL-DRMM with density extension (§2.2.2) for document retrieval followed by BCNN for snippet retrieval.
- **AUEB-NLP-5:** ABEL-DRMM with both density and confidence extensions (§2.2.2) for document retrieval followed by BCNN for

¹⁴We used NLTK v3.2.3. See <https://www.nltk.org/api/nltk.tokenize.html>.

¹⁵Available at <http://participants-area.bioasq.org/results/6b/phaseA/>. The names of our systems have been modified for the blind review.

snippet retrieval. This system was submitted for batches 2-5 only.

In combination (combo) systems, we obtained 10 versions of the corresponding model by retraining it 10 times with different random seeds, and we then used a simple voting scheme. If a document was ranked at position 1 by a model it got 10 votes, position 2 was 9 votes, until position 10 where it got 1 vote. Votes were then aggregated over all models in the combination. While voting did not improve upon the best single model, it made the results more stable across different runs.

5.4 Results

Results are given in Table 1. There are a number of things to note. First, for document retrieval, there is very little difference between our submitted models. Both PACRR- and DRMM-based models perform well (usually at the top or near the top) with less than 1 MAP point separating them. These systems were all competitive and for 4 of the 5 batches one was the top scoring system in the competition. On average the experimental ABEL-DRMM system (AUEB-NLP-4) scored best amongst AUEB submissions and in aggregate over all submissions, but by a small margin (0.1053 average MAP versus 0.1016 for TERM-PACRR). The exception was the high precision system (AUEB-NLP-5) which did worse in all metrics except F1, where it was easily the best system for the 4 batches it participated in. This is not particularly surprising, but impacted snippet selection, as we will see.

For snippet selection, all systems did well (AUEB-NLP-[1-4]) and it is hard to form a pattern that a base document retrieval model's results are more conducive to snippet selection. The exception is the high-precision document retrieval model of AUEB-NLP-5, which had by far the best scores for AUEB submissions and the challenge as a whole. The main reason for this is that the snippet retrieval component was trained assuming only relevant documents as input. Thus, if we fed it all 10 documents, even when some were not relevant, it could theoretically still rank a snippet from an irrelevant document high since it is not trained to combat this. By sending the snippet retrieval model only high precision document sets it focused on finding good snippets at the expense of potentially missing some relevant documents.

6 Related Work

Document ranking has been studied since the dawn of IR; classic term-weighting schemes were designed for this problem (Sparck Jones, 1972; Robertson and Sparck Jones, 1976). With the advent of statistical NLP and statistical IR, probabilistic language and topic modeling were explored (Zhai and Lafferty, 2001; Wei and Croft, 2006), followed recently by deep learning IR methods (Lu and Li, 2013; Hu et al., 2014; Palangi et al., 2016; Guo et al., 2016; Hui et al., 2017).

Most document relevance ranking methods fall within two categories: representation-based, e.g., Palangi et al. (2016), or interaction-based, e.g., Lu and Li (2013). In the former, representations of the query and document are generated independently. Interaction between the two only happens at the final stage, where a score is generated indicating relevance. End-to-end learning and backpropagation through the network tie the two representations together. In the interaction-based paradigm – which is where the models studied here fall – explicit encodings between pairs of queries and documents are induced. This allows direct modeling of exact or near-matching terms (e.g., synonyms), which is crucial for relevance ranking. Indeed, Guo et al. (2016) showed that the interaction-based DRMM outperforms previous representation-based methods. On the other hand, interaction-based models are less efficient, since one cannot index a document representation independently of the query. This is less important, though, when relevance ranking methods rerank the top documents returned by a conventional IR engine, which is the scenario we consider here.

In terms of biomedical document and snippet retrieval, several methods have been proposed for BioASQ (Tsatsaronis et al., 2015), mostly based on traditional IR and ML techniques. For example, the system of Jin et al. (2017), which is the top scoring one for previous incarnations of BioASQ (UTSB team), uses an underlying graphical model for scoring coupled with a number of traditional IR techniques like pseudo-relevance feedback.

The most related work from the biomedical domain is that of Mohan et al. (2017), who use a deep learning architecture for document ranking. Like our systems they use interaction-based models to score and aggregate q-term matches relative to a document, however using different document-aware q-term representations – namely best match

DOCUMENT RETRIEVAL			
System	F1	MAP	GMAP
Batch 1			
AUEB-NLP-1	0.2546	0.1246	0.0282
AUEB-NLP-2	0.2462	0.1229	0.0293
AUEB-NLP-3	0.2564	0.1271	0.0280
AUEB-NLP-4	0.2515	0.1255	0.0235
Top Competitor	0.2216	0.1058	0.0113
Batch 2			
AUEB-NLP-1	0.2264	0.1096	0.0148
AUEB-NLP-2	0.2473	0.1207	0.0200
AUEB-NLP-3	0.2364	0.1178	0.0161
AUEB-NLP-4	0.2350	0.1182	0.0161
AUEB-NLP-5	0.3609	0.1014	0.0112
Top Competitor	0.2265	0.1201	0.0183
Batch 3			
AUEB-NLP-1	0.2345	0.1122	0.0101
AUEB-NLP-2	0.2345	0.1147	0.0108
AUEB-NLP-3	0.2350	0.1135	0.0109
AUEB-NLP-4	0.2345	0.1137	0.0106
AUEB-NLP-5	0.4093	0.0973	0.0062
Top Competitor	0.2186	0.1281	0.0113
Batch 4			
AUEB-NLP-1	0.2136	0.0971	0.0070
AUEB-NLP-2	0.2148	0.0996	0.0069
AUEB-NLP-3	0.2134	0.1000	0.0068
AUEB-NLP-4	0.2094	0.0995	0.0064
AUEB-NLP-5	0.3509	0.0875	0.0044
Top Competitor	0.2044	0.0967	0.0073
Batch 5			
AUEB-NLP-1	0.1541	0.0646	0.0009
AUEB-NLP-2	0.1522	0.0678	0.0013
AUEB-NLP-3	0.1513	0.0663	0.0010
AUEB-NLP-4	0.1590	0.0695	0.0012
AUEB-NLP-5	0.1780	0.0594	0.0008
Top Competitor	0.1513	0.0680	0.0009

SNIPPET RETRIEVAL			
System	F1	MAP	GMAP
Batch 1			
AUEB-NLP-1	0.1296	0.0687	0.0029
AUEB-NLP-2	0.1347	0.0665	0.0026
AUEB-NLP-3	0.1329	0.0661	0.0028
AUEB-NLP-4	0.1297	0.0694	0.0024
Top Competitor	0.1028	0.0710	0.0002
Batch 2			
AUEB-NLP-1	0.1329	0.0717	0.0034
AUEB-NLP-2	0.1434	0.0750	0.0044
AUEB-NLP-3	0.1355	0.0734	0.0033
AUEB-NLP-4	0.1397	0.0713	0.0037
AUEB-NLP-5	0.1939	0.1368	0.0045
Top Competitor	0.1416	0.0938	0.0011
Batch 3			
AUEB-NLP-1	0.1563	0.1331	0.0046
AUEB-NLP-2	0.1494	0.1262	0.0034
AUEB-NLP-3	0.1526	0.1294	0.0038
AUEB-NLP-4	0.1519	0.1293	0.0038
AUEB-NLP-5	0.2744	0.2314	0.0068
Top Competitor	0.1877	0.1344	0.0014
Batch 4			
AUEB-NLP-1	0.1211	0.0716	0.0009
AUEB-NLP-2	0.1307	0.0821	0.0011
AUEB-NLP-3	0.1251	0.0747	0.0009
AUEB-NLP-4	0.1180	0.0750	0.0009
AUEB-NLP-5	0.1940	0.1425	0.0017
Top Competitor	0.1306	0.0980	0.0006
Batch 5			
AUEB-NLP-1	0.0768	0.0357	0.0003
AUEB-NLP-2	0.0728	0.0405	0.0004
AUEB-NLP-3	0.0747	0.0377	0.0004
AUEB-NLP-4	0.0790	0.0403	0.0004
AUEB-NLP-5	0.0778	0.0526	0.0003
Top Competitor	0.0542	0.0475	0.0001

Table 1: Performance on BioASQ Task 6b, Phase A (batches 1–5) for document and snippet retrieval (left and right tables, respectively). Systems described in Section 5.3. The italicised system is the top scoring system from AUEB’s entries and if also in bold, is the top from all official entries in that batch. *Top* is by MAP, the official metric of BioASQ. *Top Competitor* is the top scoring entry – by MAP – that is not among AUEB’s submissions.

d-term distance scores. Also unlike our work, they focus on user click data as a supervised signal, and they use context-insensitive representations of document-query term interactions.

There are several studies on deep learning systems for snippet selection which aim to improve the classification and ranking of snippets extracted from a document based on a specific query. Wang and Nyberg (2015) use a stacked bidirectional LSTM (BILSTM); their system gets as input a question and a sentence, it concatenates them in a single string and then forwards that string to the input layer of the BILSTM. Rao et al. (2016) employ a neural architecture to produce representations of pairs of the form (*question, sentence*) and to learn to rank pairs of the form (*question, relevant sentence*) higher than pairs of the form (*question, irrelevant sentence*) using Noise-Contrastive Estimation. Finally, Amiri et al. (2016) use autoen-

coders to learn to encode input texts and use the resulting encodings to compute similarity between text pairs. This is similar in nature to BCNN, the main difference being the encoding mechanism.

7 Conclusions

We presented the models, experimental set-up, and results of AUEB’s submissions to the document and snippet retrieval tasks of the sixth year of the BioASQ challenge. Our results show that deep learning models are not only competitive in both tasks, but in aggregate were the top scoring systems. This is in contrast to previous years where traditional IR systems tended to dominate. In future years, as deep ranking models improve and training data sets get larger, we expect to see bigger gains from deep learning models.

References

- Hadi Amiri, Philip Resnik, Jordan Boyd-Graber, and Hal Daumé III. 2016. Learning text pair similarity with context-sensitive autoencoders. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1882–1892, Berlin, Germany.
- Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.
- John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.
- Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. 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, Indianapolis, IN.
- Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems 27*, pages 2042–2050.
- Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. PACRR: A position-aware neural IR model for relevance matching. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1049–1058, Copenhagen, Denmark.
- Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2018. Co-PACRR: A context-aware neural IR model for ad-hoc retrieval. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, pages 279–287, Marina Del Rey, CA.
- Zan-Xia Jin, Bo-Wen Zhang, Fan Fang, Le-Le Zhang, and Xu-Cheng Yin. 2017. A multi-strategy query processing approach for biomedical question answering: Ustb_pir at BioASQ 2017 Task 5B. In *BioNLP 2017*, pages 373–380.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Robert Krovetz. 1993. Viewing morphology as an inference process. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 191–202, Pittsburgh, PA.
- Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, pages 1367–1375, Lake Tahoe, NV.
- Ryan McDonald, Georgios-Ioannis Brokos, and Ion Androutsopoulos. 2018. Deep relevance ranking using enhanced document-query interactions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, pages 3111–3119, Lake Tahoe, Nevada.
- Sunil Mohan, Nicolas Fiorini, Sun Kim, and Zhiyong Lu. 2017. Deep learning for biomedical information retrieval: Learning textual relevance from click logs. In *BioNLP 2017*, pages 222–231.
- Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 24(4):694–707.
- Jinfeng Rao, Hua He, and Jimmy J. Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *CIKM*, pages 1913–1916, Indianapolis, IN.
- Stephen Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. 1995. Okapi at TREC3. In *Overview of the Third Text Retrieval Conference*, pages 109–126.
- Stephen E. Robertson and Karen Sparck Jones. 1976. Relevance weighting of search terms. *Journal of the Association for Information Science and Technology*, 27(3):129–146.
- Aliaksei Severyn and Alessandro Moschitti. 2015. 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, Santiago, Chile.
- Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.
- 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 Artières, Axel-Cyrille Ngonga Ngomo, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, and Georgios Paliouras. 2015. An overview of the BioASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16(1):138.

- Di Wang and Eric Nyberg. 2015. 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 (Volume 2: Short Papers)*, pages 707–712, Beijing, China.
- Xing Wei and W Bruce Croft. 2006. LDA-based document models for ad-hoc retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178–185, Seattle, WA.
- Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.
- Chengxiang Zhai and John Lafferty. 2001. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 334–342, New Orleans, LA.