Universal Passage Weighting Mecanism (UPWM) in BioASQ 9b

Tiago Almeida[1], Sérgio Matos[1]

[1]*University of Aveiro, IEETA*

Abstract

This paper presents the participation of the University of Aveiro Biomedical Informatics and Techologies group (BIT) in the ninth edition of the BioASQ challenge for document and snippet retrieval. Our proposed systems follow a two-stage retrieval pipeline, similar to our BioASQ 8B submissions. However, we completely rebuilt our neural ranking model, maintaining the key ideas of its inception while improving its computational efficiency as well as adding interoperability with the transformer architecture. This resulted in a novel universal passage weighting mechanism (UPWM), which offers a more powerful way to derive a document relevance score from the combination of its sentences. More concretely, we have built two variants that use our passage mechanism, the lightning UPWM and the transformer UPWM. The first uses a shallow interaction model and the second uses a BERT model. Additionally, we also propose an effective pairwise joint training mechanism that combines document retrieval with snippet retrieval. Our systems achieved competitive results scoring at the top and close to the top for all the batches, with MAP values ranging from 0.3573 to 0.4236 in the document retrieval task. Although we only submitted for the snippet retrieval task in the last two batches, our system scored at top position in the last batch by using rank reciprocal fusion of pointwise and pairwise joint training approaches. Code to reproduce our submissions are available on https://github.com/bioinformatics-ua/BioASQ9b.

1. Introduction

The BioASQ [1] challenge is an annual competition on document classification, retrieval and question-answering applied to the biomedical domain. This competition is notorious for continuously fostering the research in automatic and intelligent retrieval systems over the biomedical literature. A fresh example is the difficulty that researchers and biomedical experts have to search the growing literature about the 2019 novel coronavirus, showing us that new and more powerful methods are still needed.

More concretely, the BioASQ challenge is divided into two tasks (**A** and **B**) that are isolated challenges. **Task A** addresses the biomedical annotation problem, and is concerned with automatic document labelling with terms from the MeSH hierarchy. On the other hand, **task B** is further subdivided into **phase A** and **phase B**, the first addressing the information retrieval problem and the second addressing the answer extraction and answer generation problems.

In more detail, in **phase A** the objective is to retrieve, from the PubMed baseline, the most relevant articles and/or document snippets that answer a given biomedical question written in English. In contrast, the objective in **phase B** is to extract or generate ideal answers from the information retrieved in **phase A**.

This paper describes our participation the participation of the Biomedical Informatics and Techologies (BIT) group of the Aveiro University in the BioASQ task B phase A challenge. Our approach is a direct evolution of the document retrieval approach used in our previous participation [2]. This year we focused on a ground up rebuilt of the previous model by maintaining the key ideas of its inception, while improving the computation flow for efficiency purposes and adding interoperability with the transformer architecture. In the end, we devised a universal passage weighting mechanism (UPWM) that offers a novel way to combine the sentence relevance in order to derive the final document score. This mechanism tries to solve the usual overlooked problem of sentence aggregation. More precisely, it is usually unfeasible to feed the entire document to a neural IR model and a common practice is to do sentence splitting followed by simple level sentence aggregation [3, 4]. However, little attention has been given to this important step.

We built two variants that use our passage mechanism, the **lightning UPWM** (L-UPWM) and the **transformer UPWM** (T-UPWM). The first uses a shallow interaction model, with only **597 trainable parameters**, and the later uses a BERT model. These were the cornerstone models of our submissions for this year challenge for the document and snippet retrieval challenge.

Our submissions achieved the top and close to the top positions for all document retrieval batches. While we only participated in the last two batches for snippet retrieval, we still achieved a top scoring position in the last batch. Finally, our transformer UPWM consistently outperformed the lightning UPWM bringing to evidence the efficacy vs efficiency trade off.

In the remaining of the paper we start by describing our universal passage weighting mechanism and detailing the two concrete implementations, lightning UPWM and transformer UPWM. We then describe the submissions and present and discuss the results obtained.

2. UPWM

The universal passage weighting mechanism (UPWM) is, as the name suggests, a high-level mechanism that provides other neural architectures the capability to perform sentence score aggregation. In other words, this mechanism can be viewed as a wrapper that encapsulates already existing models offering them the capability to better combine the individual sentence scores in order to derive the final document score.

The UPWM is inspired by the human judgement process of selecting relevant articles, as proposed in previous works [5, 6]. More precisely, when searching for some information, a person usually scans a document looking for relevant sentences, signalled by the presence of keywords that are similar to their information need. Then the relevance of the entire document will be judged based only on the selected sentences. Therefore, the purpose of the UPWM is to mimic this judgement process and combine it with the neural relevance signal extracted by another model. So, this human judgement process can be viewed as a heuristic to guide neural models during the sentence aggregation step.

Another aspect taken into consideration by the UPWM is that query terms are not equally relevant. Similarly, in the human judgement process, when scanning a document only a few keywords, the most representative terms, are considered. So, based on this observation we also compute the importance that each query term carries and weigh each sentence by this importance, thus boosting the sentences that contain the most important query terms.
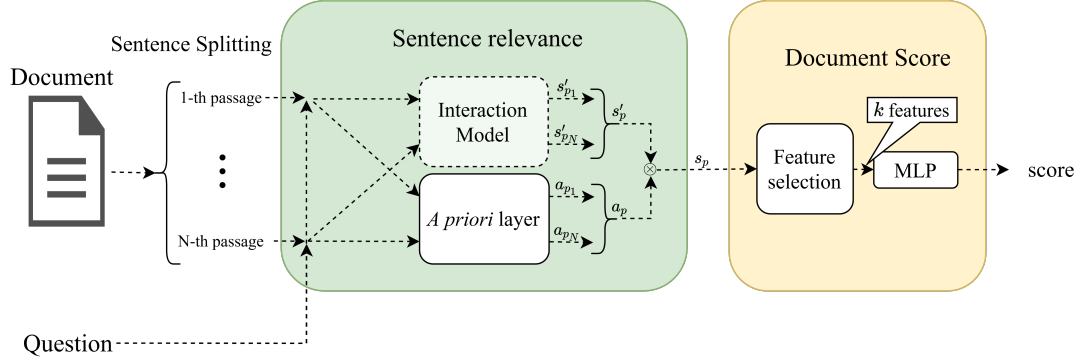
Figure 1: A general overview of the Universal passage weighting mechanism.

Figure 1 presents an overview of the main architectural concepts of the UPWM, which is divided into two major blocks, the **sentence relevance** block and the **document score** block. The idea is that the first block will produce the individual sentence scores that carry the sentence relevance. Then the document score will select the most representative features based on these sentences to derive the final document relevance with respect to the query.

2.1. Sentence Relevance Block

The sentence relevance block has two parallel layers, the Interaction Model layer and the *A priori* layer. Both layers produces scores for each sentence that will be linearly combined. The intuition here is that the interaction model layer focuses on thoroughly analysing the sentence information, while the *a priori* layer will act as a heuristic by mimicking the quick human judgement process of finding relevance. In other words, the main idea of the *a priori* layer is to produce sentence scores without thoroughly processing the sentences, therefore its name, since we will give an "*a priori*" score without analysing it.

Then, the final sentence score comes from the linear combination of the *a priori* scores with the interaction model scores. So, the *a priori* score will act as a gating mechanism deciding which scores from the interaction model should be considered for the document ranking. Another interpretation can be gained by considering the types of signals that both extract. Specifically, the interaction model layer will focus on analysing the meaning, context and sentence semantics, hence carrying a more semantic interaction signal, while the *a priori* will only focus on a more exact matching type of signal, weighted by the importance of each individually query term.

2.1.1. Interaction Model Layer

The interaction model layer acts as a placeholder for neural models that are more specialised in analysing the relevance between the query and sentence. For a better fit, the type of models that make sense to adopt in this layer are models that derive the final relevance score by taking into consideration the meaning, context and semantic relations between the query and the sentence. For example, ARC-II [7] is an example of an earlier and simpler candidate. However, a more powerful transformer based model, like BERT [8] can also be adopted as the interaction model in this architecture.

2.1.2. *A Priori* Layer

The *a priori* layer implementation is summarised in Figure 2 and has two major steps, the exact matching signal extraction and the query terms importance weighting.
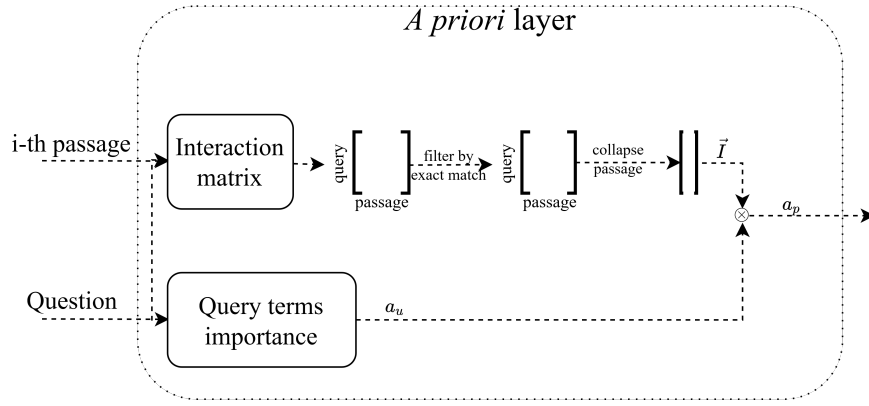
Figure 2: The main concepts behind the inner workings of the *a priori* layer

To better understand the layer inner workings, lets us first define a **query** as a sequence of tokens $q = \{u_0, u_1, ..., u_Q\}$, where u_i is the i-th token of the query and Q the size of the query; a **document passage** as $p = \{v_0, v_1, ..., v_S\}$, where v_j is the j-th token of the passage and S the size of the passage; and a **document** as a sequence of passages $D = \{p_0, p_1, ..., p_N\}$, where N is the total number of passages in the document. To further simplify the explanation lets consider that this layer is only applied to one sentence query pair, since the extension to N pairs is trivially achieved by replicating this procedure for each pair.

Regarding the exact matching signal extraction, an interaction matrix is first created by applying an interaction function, $f_{interaction}(q, p)$, to the query and the passage. This function will perform a pairwise combination of all the query terms with all the passage terms. The output is a matrix, $I^{[0,1]}$, where the rows are the query terms and the columns are the passage terms. We adopt two types of implementations for this function, an *exact interaction* and a *semantic interaction*. The first one directly uses the token index to create the matrix I, therefore can be defined as follows

$$I_{ij} = \begin{cases} 1 & u_i = v_j \\ 0 & otherwise \end{cases}. \tag{1}$$

The second approach computes the cosine similarity between the embedding of the terms, described as follows,

$$I_{ij} = \frac{\vec{u_i}^T \cdot \vec{v_j}}{\|\vec{u_i}\|_2 \times \|\vec{v_j}\|_2}. \tag{2}$$

In either case, the exact matching signal will always be captured, since it will correspond to the matrix entry that has the corresponding value of 1 or close to. The next step is to filter these exact matching signals. For that, we defined a *matching threshold*, that when applied over the interaction matrix returns a matrix with all matching terms between the query and the passage, i.e. it returns a matrix where each entry defines if a query term is present in the passage or not, as described in Equation 3

$$I_{ij} = \begin{cases} 1 & if\ I_{ij} >=\ threshold \\ 0 & if\ I_{ij} <\ threshold \end{cases}. \tag{3}$$

Note, that it only makes sense to apply this equation over the semantic interaction, since the exact interaction directly indicates all the exact matches.

Then, since we only care about the presence or absence of a term, we collapse the column dimension in I, transforming it into a vector, $\vec{I}^{[0,1]}$, where each entry indicates if a query-term is present or not in the passage. Note that each entry in I is a boolean value; we also tried different alternatives, such as using the number of times a query term appeared in a sentence, but this did not improve the results.

In parallel with this step, this layer also computes the importance of each query-term, since different terms in a query can carry different importance regarding the information need, as addressed by [6, 9]. To accomplishing this, we compute a probabilistic distribution over the query terms, as shown in Equation 4,

$$c_{u_i} = \vec{w}^T \cdot \vec{u_i}\,,$$
$$a_{u_i} = \frac{e^{c_{u_i}}}{\sum_{u_w \in Q} e^{c_{u_w}}}. \tag{4}$$

Here, we used the standard softmax operation to compute the probabilistic distribution over a linear combination of each query term embedding, $\vec{u_i}$, and a trainable vector, w. Finally, a_{u_i} corresponds to the probabilistic importance of the query-term u_i regarding the entire query.

Finally, the *a priori* sentence score arises from the linear combination of the query-importance distribution and the presence vector, $\vec{I}$, as described in Equation 5,

$$a_p = \sum_{i \in Q} a_{u_i} \times \vec{I_i}. \tag{5}$$

Given this formulation, a_p, represents the importance of each sentence following the human judgement heuristic. Let us consider a passage containing all the query terms as an intuitive

example. Under this condition the *a priori* sentence score will be 1, and contrarily, if a passage does not contain any term, it will have a score of zero. Similarly, if a sentence contains important query-terms, it will have a score close to 1, and close to zero on the opposite case.

An important note is to consider the role of the *matching threshold.* If too high, e.g., $threshold = 0.99$ it will only consider exact match signals, i.e., terms that exactly appear in the passage. However, for lower values, it will also include semantically similar terms, e.g., a $threshold = 0.7$ can be beneficial since it will dynamically include semantically similar terms, in some cases synonyms.

2.2. Document Score Block

The document score block aims to produce the final document score based on the passages score previously obtained. To avoid creating a bias for longer documents, since they have more scores, we employed a feature selection step. More precisely, we construct a feature vector that contains the max sentence score, the normalised summation over the scores, the average over the scores and several top k-max-average scores. Finally, the document score is computed by combining this feature vector using a multi-layer perceptron.

3. UPWM Models

As previously mentioned, the UPWM is intended to be combined with an neural model that is capable of computing the relevance between the query and a sentence. So, in this section we present the two concrete implementation that we adopted for the BioASQ challenge, the lightning UPWM and the transformer UPWM.

3.1. Lightning UPWM

The lightning UPWM, uses a neural interaction model similar to our last year submission [2], which was already a fast and lightweight model [10]. However, with this much cleaner architecture we achieved a 4x speed up making it a lightning fast model, when compared to the current transformer based models.
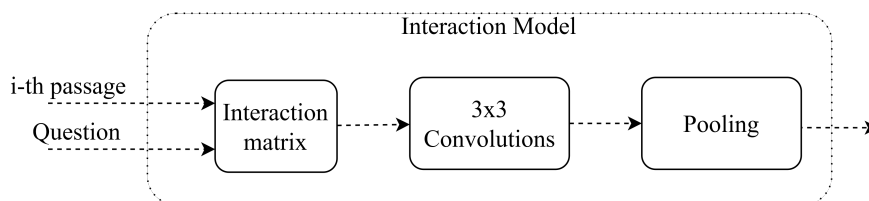

Figure 3: A simplistic overview of the neural architecture for the interaction model.

We show in Figure 3 the overview of the neural architecture. Very briefly, an interaction matrix is built using Equation 2. Then 3 by 3 convolution kernels are applied over this matrix to learn context patterns that are then extracted by a pooling layer. This layer is applied over the filter dimension and we combined the max, average and k-max average pooling.

3.2. Transformer UPWM

The transformer UPWM, as the name suggests, uses a transformer based model in the interaction model layer. More precisely, we chose PubMedBERT [11], which is a BERT model that was trained from scratch using abstracts from PubMed and full-text articles from PubMed Central. This model keeps the biomedical specific terms, that would be decomposed into subwords if using other BERT models. This is an important aspect since we can directly use, in the *a priori* layer, the tokens produced by the PubMedBERT model.

To produce the relevance score between the query and the sentence, we adopted the usual strategy [12], described in Figure 4, of concatenating the query tokens with the sentence tokens, separated by the *[SEP]* token. Then we feed this to the BERT model, that outputs a sequence of contextualised embedding per tokens. Finally, we feed the *[CLS]* embedding to a linear layer in order to produce the sentence score.
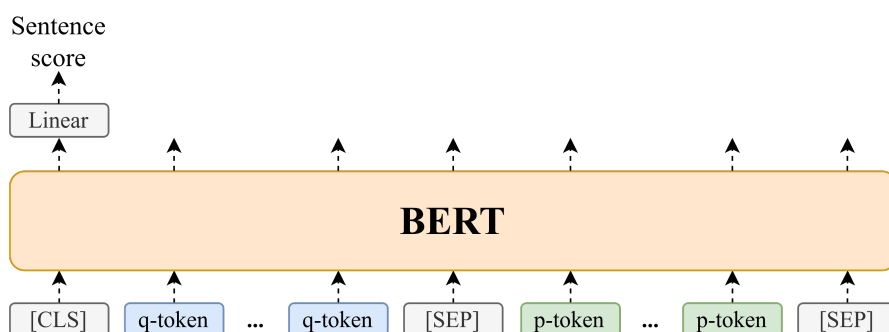

Figure 4: Diagram of the interaction model as a BERT model.

4. Overall Architecture

This section addresses the complete system architecture that we adopted for the BioASQ 9b challenge. Similarly to our last year submission, we used a two-stage reranking mechanism, as presented in Figure 5.

For the first stage we adopted a traditional retrieval model, BM25 [13], to efficiently select the top-100 scientific articles for each biomedical question. Then this set of documents is fully reranked by our neural model, in this case, the lightning UPWM and transformer UPWM.

Given this pipeline and the operation of the UPWM as described above, we observe that the UPWM operates a detailed analysis of the exact match signals captured by the BM25. This makes sense, because according to the UPWM, the interaction model will only evaluate sentences that carry some exact match signal, which is the only type of signal used by BM25. Therefore, we can say that, in this pipeline, the UPWM will "look" more in depth to the signals that contributed for the BM25 score.
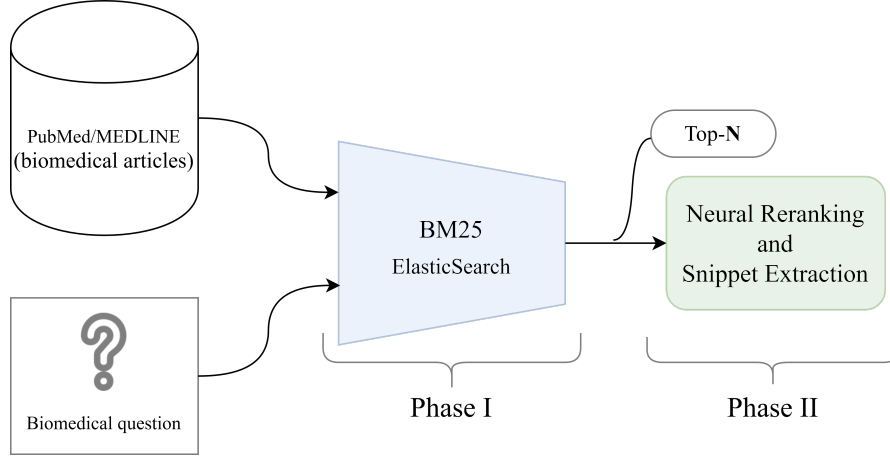
Figure 5: Overview of our two-stage retrieval system.

4.1. Training

Regarding the training of the neural models, both were trained with the same exact data collection from BioASQ. In terms of document ranking, similarly to last years submission, we trained both models using a pairwise cross entropy loss, described in Equation 6,

$$L_{doc}(q, d^+, d^-) = -log(\frac{e^{(score(q,d^+))}}{e^{(score(q,d^+))} + e^{(score(q,d^-))}}).$$ (6)

Here, the loss is computed as a function over a triplet that contains a query, a positive document and a negative document. Since the BioASQ data only provides positive examples, we sampled the negatives examples from the documents in the BM25 ranking order that are not labelled as positives. Additionally, given that the gold-standard was built as a concatenation of the judged documents from different years, we decided to restrict the BM25 search by year so that only the available documents at that time are available for the training of the model.

Additionally to the document level training, we also tried to perform joint training by using the snippets feedback data available in the training data. Equation 7 describes how the overall loss is computed in the joint training approach, namely as a weighted average between the document loss and the snippet loss.

$$L = \gamma L_{doc} \times (1 - \gamma)L_{snippet}.$$ (7)

Regarding the snippet loss, we experimented with pointwise and pairwise variants for the loss. Furthermore, we only applied the snippet loss to the sentences from the positive documents that had feedback.

Following the ideas of joint training presented in [14], we further augmented our UPWM to produce new snippet scores, as presented in Figure 6. Note that the actual solution already produces snippet scores in the sentence relevance block. However, these do not depend on the final document score, therefore the snippet loss would not contribute, through back propagation,
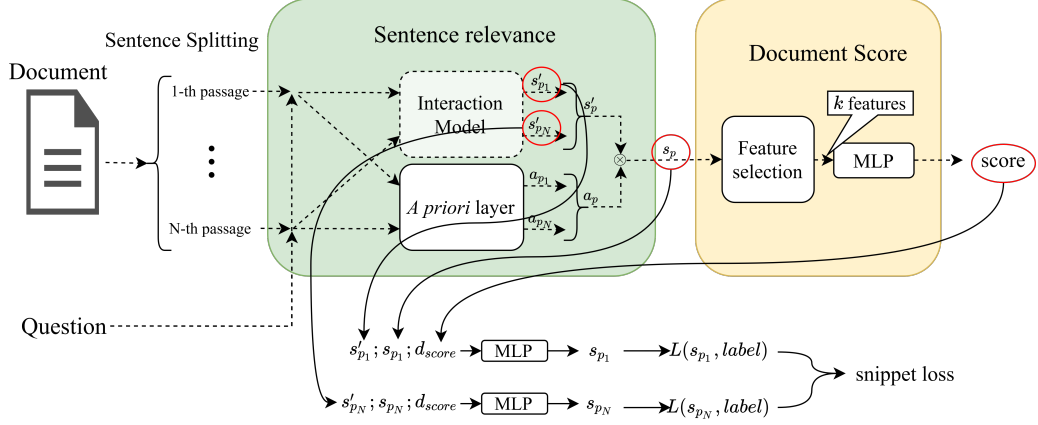
Figure 6: Pointwise snippet loss.

to the training of the document score block. So, we added a MLP that computes a new snippet score from the concatenation of the interaction model score, sentence score and document score. This way, the new snippet score is dependent on the final document score, making it a more joint train approach.

For the pointwise loss, we adopted the binary cross entropy loss described in Equation 8,

$$L_{snippet}(y, \hat{y}) = -(ylog(\hat{y}) + (1 - y)log(\hat{y})). \tag{8}$$

Here, y correspond to the true relevance of a snippet, 1 for positive and 0 for negative, and $\hat{y}$ corresponds to the probability of a snippet being positive assigned by the model. From this definition, we also explored using label smoothing by considering a wide range of positive and negative values for y.

For the pairwise loss, we adopted the pairwise cross entropy loss, already described in Equation 6. This pairwise loss is computed between all the positive snippets and all the negative snippets that belong to the same document. As before, a snippet loss is only computed over the snippets of a positive document.

5. Submission

In this section, we start by describing the data collection and some pre-processing steps that are common to our official submission for all the batches. Then we describe each run that was submitted for the BioASQ 9b phase A challenge.

5.1. Collection and Pre-processing

In this year's edition, the document collection was the 2020 PubMed/MEDLINE annual baseline consisting of almost 31 million articles. However, all the articles with missing abstract were discarded, meaning that around 21 million articles were indexed with ElasticSearch using the **english** text analyser, which performs tokenization, stemming and stopword filtering.

Regarding the neural model, we built a simple Regex based tokenizer and trained 200-dimension word embeddings using the GenSim [15] implementation of the word2vec algorithm [16] over the 21 million abstracts from the baseline. Furthermore, when using the UPWM each document is split into a set of individual sentences through the Punkt algorithm [17].

For the training data we used the BioASQ dataset with the exception of the last year test set that we used for validation purposes. With respect to the joint training, given that the snippet gold standard does not respect sentence boundaries, e.g. it can be composed of several sentence, we consider a sentence to be relevant (hard label of 1) if its text matches the text in a positive snippet. However, when using soft label, we instead use the overlap between the sentence text and the gold snippet text as the value for the soft label. The intuition is that a sentence that only partially belongs to a gold snippet should not be considered fully relevant, and therefore should not have a label of 1.

Regarding the first stage of the pipeline, we finetuned the BM25 parameters, $k1$ and b, for each batch by performing an extensive grid search. The validation data used for this process corresponds to the last year's test set for each corresponding batch.

5.2. Runs

This year, the document/snippet retrieval challenge received, on average across all batches, 27 submission from seven teams. Our group submitted five runs for each batch, which are identified by the prefix "bioinfo" on the official results[1]. Table 1 presents a summarised description of the systems used in each run, where L-UPWM stans for lightning UPWM, T-UPWM stands for transformer UPWM, JT corresponds to joint training and "-> RRF" means that we made an ensemble of several runs using the rank reciprocal fusion (RRF) method [18].

Table 1
Summary of the submitted runs for each round of the 2021 BioASQ 9B phase A.

Run name	Description		
	Batch 1 2 and 3	Batch 4	Batch 5
bioinfo-0	L-UPWM	L-UPWM -> RRF	L-UPWM -> RRF
bioinfo-1	L-UPWM -> RRF	L-UPWM + JT(Pointwise) -> RRF	L-UPWM + JT(Point/Pairwise) -> RRF
bioinfo-2	T-UPWM	T-UPWM -> RRF	T-UPWM -> RRF
bioinfo-3	T-UPWM -> RRF	T-UPWM + JT(Pointwise) -> RRF	T-UPWM + JT(Point/Pairwise) -> RRF
bioinfo-4	RRF of all runs	T-UPWM + JT(Pointwise)	T-UPWM + JT(Pointwise)

In more detail, for the first, second and third batches, as observable in Table 1, we submitted the same base system configuration for each run. In some cases, we made an ensemble run that used several models trained with a slight difference in its hyperparameters. In these three batches we did not employ the joint training technique and therefore we did not submit any run for snippet retrieval.

For the fourth batch, the main difference was the addition of the joint training methodology with pointwise snippet loss. Then in the fifth batch we only added the pairwise snippet loss for some models that used joint training. More precisely, runs "bioinfo-1" and "bioinfo-3"

correspond to RRF ensembles of two models, one jointly trained using pointwise and the other using pairwise snippet loss.

Regarding snippet retrieval, the only runs in which we submitted snippets were the ones using joint training. Furthermore, to produce the ranked snippet list we followed a heuristic where the sentences presented on the top documents should have a higher probability of being relevant. Therefore, during snippet ranking we preserve this document order and only extract the snippets that score above a specific threshold. More precisely, for the forth batch we retrieved snippets from the top-10 retrieved documents, while for the fifth batch we only considered the top-1 retrieved document.

6. Results and Discussion

In this section, we separately address the document results and the snippet results, since we only submitted snippets for the last two batches. Note that at the time of writing only the preliminary results, regarding the systems' performance, were available.

6.1. Document Retrieval

The overall results regarding the document retrieval task are shown in Table 2, together with the median of all submissions, the top performing system in each batch, apart from our own submissions, and the baseline score obtained with BM25, corresponding to the ranking order without applying neural reranking. The results are organised in terms of Mean Average Precision at ten (MAP@10), which was the official measure adopted by the organisers to rank all the submissions. There were a total of 16, 30, 29, 27 and 28 submissions respectively for each batch.

Table 2
Summary of the results obtained for the document retrieval task.

Run name	Batch 1		Batch 2		Batch 3		Batch 4		Batch 5	
	Rank	MAP	Rank	MAP	Rank	MAP	Rank	MAP	Rank	MAP
bioinfo-0	15	31.73	18	33.80	16	35.50	12	38.45	21	30.09
bioinfo-1	8	32.96	15	35.37	13	36.22	11	38.49	15	31.86
bioinfo-2	3	35.15	6	37.84	10	38.24	**1**	**42.36**	**1**	**35.86**
bioinfo-3	**1**	**35.73**	4	38.13	8	39.35	7	40.42	8	34.40
bioinfo-4	2	35.25	5	37.87	9	38.65	8	40.42	7	34.61
Baseline (BM25)		31.03		33.20		35.94		36.86		32.89
Median		32.93		34.85		35.64		38.13		32.03
Top competitor	4	34.60	1	39.90	1	40.40	2	41.92	2	35.37

Looking at the results presented in Table 2, from a general perspective our transformer UPWM model had top performing results, outscoring every other system in terms of MAP@10 in three batches while being competitive in the remaining two batches. On the opposite side, the lightning UPWM model achieved results closer to the median. This observation brings to evidence the trade off between efficacy and efficiency, since our lightning UPWM model is 63 times faster than the transformer approach with a GPU (K80) [10]. So, although it did not achieve

the same kind of retrieval performance as the transformer counterpart, the lightning UPWM model may still be a viable or better solution depending on the perspective or requirements.

To better understand the effects of the neural ranking solution, we added the row "baseline" in Table 2, which represents the BM25 ranking order that all of our neural solution used for reranking. We present in Figure 7 the comparison between all the submission scores and the baseline score, to visualise the gains achieved by the neural reranking strategy.
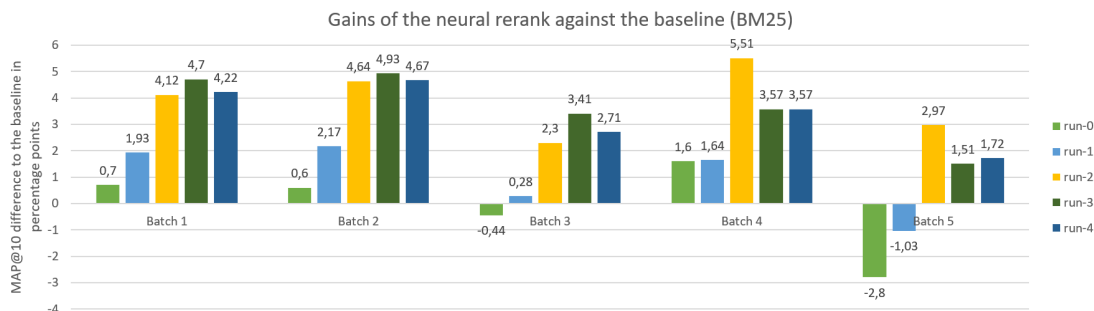

Figure 7: Neural reranking gains when compared to the BM25 baseline.

From an overall point of view, the neural reranking seems to be beneficial in achieving small increases in MAP@10 percentage points. The lightning UPWM model on the last batch was the only case were the neural reranking negatively influenced the baseline ranking order. This may be a consequence of using only the top 100 documents in the reranking phase. Therefore, we leave as future work the analysis of the impact of k in the top-k document selection for reranking. Figure 7 also clearly shows the difference in performance between the L-UPWM and the T-UPWM, since the gains of the solutions that used T-UPWM are clearly higher than the submissions that used the L-UPWM, as previously mentioned.

For a better context regarding the overall ranking positions, we show in Figure 8 the difference between our best system submission at each batch against the median score at that batch. As observable, our best submission achieved results that are clearly superior when compared to the median for all the batches. This is true even when our best submission was not in the top-5 (batch 3), which shows that the top scores were highly competitive and close.

6.2. Snippet Retrieval

The main results regarding the snippet retrieval task are shown in Table 2, together with the median of all submissions, the top performing system in each batch. Moreover, since we only submitted an experimental run for the fourth and fifth batches, we applied the bioinfo-3 system submitted for the fifth batch to the remaining batches, to gain an idea of the performance that our best snippet solution could have achieved. The results are organised according to the harmonic mean of precision and recall at ten (F1@10), which was the official measure adopted by the organisers. There were a total of 11, 19, 17, 18 and 19 submissions respectively for each batch.
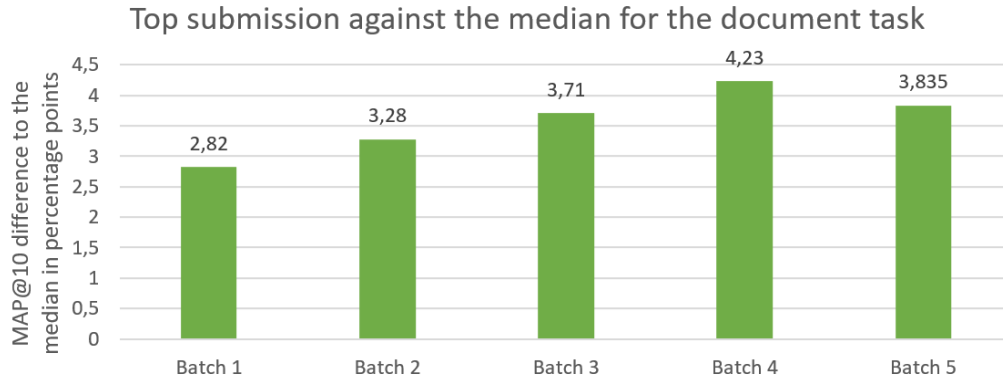
Figure 8: MAP@10 difference between our best run at each batch against the median score at that batch.

Table 3
Summary of the results obtained for the snippet retrieval task.

Run name	Batch 1		Batch 2		Batch 3		Batch 4		Batch 5	
	Rank	F1	Rank	F1	Rank	F1	Rank	F1	Rank	F1
bioinfo-1							18	10.67	5	15.57
bioinfo-3							14	14.25	**1**	**17.68**
bioinfo-4							13	15.03	2	16.83
Post-challenge (bioinfo-3)	(3)	16.78	**(1)**	**19.71**	(3)	20.02	(2)	19.24		
Median		11.32		14.86		16.84		17.18		14.28
Top result	1	18.45	1	18.16	1	20.42	1	20.61	3	16.73

As shown in in Table 3, our last approach, in batch five achieved a top scoring result. Additionally, when applying this model to the remaining batches the performance is consistent, with results in the top-3.

In Figure 9, we show a comparison between the performance of the bioinfo-3 system (batch 5) in all batches against the median snippet results. Again, these results show the consistent performance of our system.

6.3. Joint Training Benefits

In order to assess the effect of the joint training methodology, we present in Figure 10 the difference between the same model with and without using the joint training approach. When looking at the lightning UPWM (L-UPWM) joint training lead to a marginal improvement on batch fourth and a clear more positive impact on the fifth batch. Otherwise, the transformer UPWM (T-UPWM) implementations clearly underperform when using the joint training methodology. From these results joint training seems only to be beneficial to the smaller model. The main reason to explain this behaviour may be related to the amount of training data available. Since we build the training set to always output documents that had positive snippets, this had a consequence of reducing the amount of training data available when compared with the version that is only trained with document feedback. More precisely, the joint training approach uses
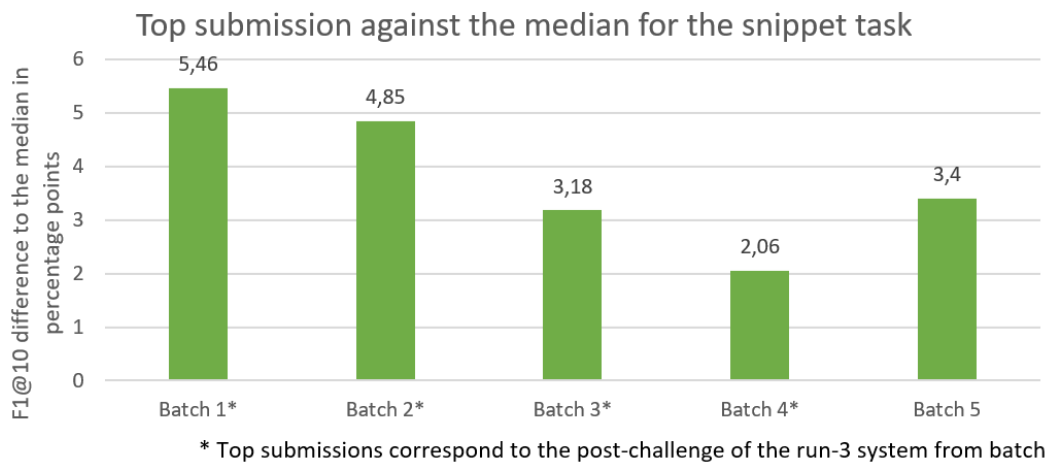
Figure 9: F1@10 difference between our best run at each batch against the median score of that batch.

18% less training pairs when compared to the document only training.
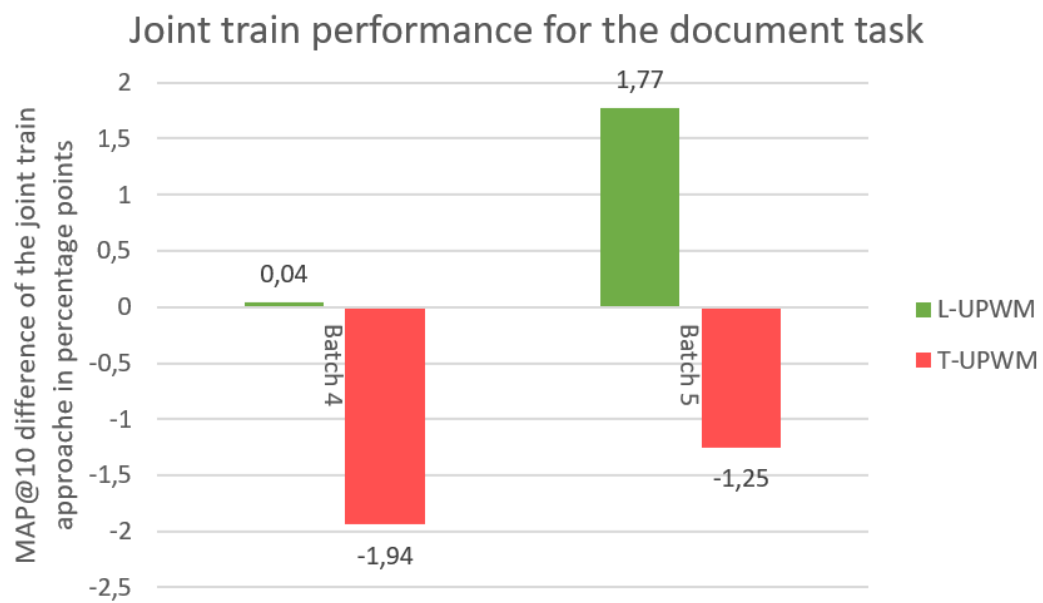
Figure 10: Comparison of both UPWM implementations with and without the joint train methodology.

7. Conclusion

In this paper we propose a novel sentence aggregation technique, named universal passage weighting mechanism (UPWM), that can be combined with neural interaction based models. We demonstrate two simple implementations, one a fast and lightweight variant called lightning UPWM (L-UPWM) and a larger one, relying in the success of the transformer architecture, the transformer UPWM (T-UPWM). The first uses a simple CNN and pooling architecture, while the later uses the BERT model as the interaction based model in the UPWM architecture.

We submitted runs with both implementations to the BioASQ 9b phase A challenge, addressing the document and snippet retrieval tasks. For document task our best solution, T-UPWM, was able to outperform all the other system in three of the five batches, while remaining competitive in the others. The L-UPWM showed an inferior performance, which was expected given that is a much lighter model that is 63 times faster than the T-UPWM. In both cases, neural reranking was generally beneficial when looking at the performance gains against the BM25 baseline. Finally, we also propose a joint training approach that showed encouraging results, leaving a clear open path for future work.

References

[1] G. Tsatsaronis, G. Balikas, P. Malakasiotis, I. Partalas, M. Zschunke, M. Alvers, D. Weißenborn, A. Krithara, S. Petridis, D. Polychronopoulos, Y. Almirantis, J. Pavlopoulos, N. Baskiotis, P. Gallinari, T. Artieres, A.-C. Ngonga Ngomo, N. Heino, E. Gaussier, L. Barrio-Alvers, G. Paliouras, An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition, BMC Bioinformatics 16 (2015) 138. doi:`10.1186/s12859-015-0564-6`.

[2] T. Almeida, S. Matos, BIT.UA at BioASQ 8: Lightweight neural document ranking with zero-shot snippet retrieval, in: L. Cappellato, C. Eickhoff, N. Ferro, A. Névéol (Eds.), Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020, volume 2696 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2696/paper_161.pdf.

[3] X. Liu, W. B. Croft, Passage retrieval based on language models, in: Proceedings of the Eleventh International Conference on Information and Knowledge Management, CIKM '02, Association for Computing Machinery, New York, NY, USA, 2002, p. 375–382. URL: https://doi.org/10.1145/584792.584854. doi:`10.1145/584792.584854`.

[4] Z. Dai, J. Callan, Deeper text understanding for ir with contextual neural language modeling, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'19, Association for Computing Machin-

ery, New York, NY, USA, 2019, p. 985–988. URL: https://doi.org/10.1145/3331184.3331303. doi:10.1145/3331184.3331303.

[5] L. Pang, Y. Lan, J. Guo, J. Xu, J. Xu, X. Cheng, Deeprank: A new deep architecture for relevance ranking in information retrieval, CoRR abs/1710.05649 (2017). URL: http://arxiv.org/abs/1710.05649. arXiv:1710.05649.

[6] T. Almeida, S. Matos, Calling attention to passages for biomedical question answering, in: J. M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M. J. Silva, F. Martins (Eds.), Advances in Information Retrieval, Springer International Publishing, Cham, 2020, pp. 69–77.

[7] B. Hu, Z. Lu, H. Li, Q. Chen, Convolutional neural network architectures for matching natural language sentences, in: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS'14, MIT Press, Cambridge, MA, USA, 2014, p. 2042–2050.

[8] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR abs/1810.04805 (2018). URL: http://arxiv.org/abs/1810.04805. arXiv:1810.04805.

[9] J. Guo, Y. Fan, Q. Ai, W. Croft, A deep relevance matching model for ad-hoc retrieval, 2016, pp. 55–64. doi:10.1145/2983323.2983769.

[10] T. Almeida, S. Matos, Benchmarking a transformer-FREE model for ad-hoc retrieval, in: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, Association for Computational Linguistics, Online, 2021, pp. 3343–3353. URL: https://www.aclweb.org/anthology/2021.eacl-main.293.

[11] Y. Gu, R. Tinn, H. Cheng, M. Lucas, N. Usuyama, X. Liu, T. Naumann, J. Gao, H. Poon, Domain-specific language model pretraining for biomedical natural language processing, 2020. arXiv:arXiv:2007.15779.

[12] R. Nogueira, K. Cho, Passage re-ranking with BERT, CoRR abs/1901.04085 (2019). URL: http://arxiv.org/abs/1901.04085. arXiv:1901.04085.

[13] S. Robertson, H. Zaragoza, The probabilistic relevance framework: Bm25 and beyond, Found. Trends Inf. Retr. 3 (2009) 333–389. doi:10.1561/1500000019.

[14] D. Pappas, R. McDonald, G.-I. Brokos, I. Androutsopoulos, AUEB at BioASQ 7: Document and snippet retrieval, in: P. Cellier, K. Driessens (Eds.), Machine Learning and Knowledge Discovery in Databases, Springer International Publishing, Cham, 2020, pp. 607–623.

[15] R. Řehůřek, P. Sojka, Software Framework for Topic Modelling with Large Corpora, in: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, ELRA, Valletta, Malta, 2010, pp. 45–50. http://is.muni.cz/publication/884893/en.

[16] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, Curran Associates Inc., Red Hook, NY, USA, 2013, p. 3111–3119.

[17] T. Kiss, J. Strunk, Unsupervised multilingual sentence boundary detection, Computational Linguistics 32 (????) 485–525. URL: https://www.aclweb.org/anthology/J06-4003. doi:10.1162/coli.2006.32.4.485.

[18] G. V. Cormack, C. L. A. Clarke, S. Buettcher, Reciprocal rank fusion outperforms condorcet and individual rank learning methods, in: Proceedings of the 32nd International ACM