Macquarie University at BioASQ 6b: Deep learning and deep reinforcement learning for query-based multi-document summarisation

Diego Mollá
Macquarie University
Sydney, Australia
diego.molla-aliod@mq.edu.au

Abstract

This paper describes Macquarie University's contribution to the BioASQ Challenge (BioASQ 6b, Phase B). We focused on the extraction of the ideal answers, and the task was approached as an instance of query-based multi-document summarisation. In particular, this paper focuses on the experiments related to the deep learning and reinforcement learning approaches used in the submitted runs. The best run used a deep learning model under a regression-based framework. The deep learning architecture used features derived from the output of LSTM chains on word embeddings, plus features based on similarity with the query, and sentence position. The reinforcement learning approach was a proof-of-concept prototype that trained a global policy using REINFORCE. The global policy was implemented as a neural network that used $tf.idf$ features encoding the candidate sentence, question, and context.

1 Introduction

The BioASQ Challenge[1] consists of various tasks related to biomedical semantic indexing and question answering (Tsatsaronis et al., 2015). Our participation in BioASQ for 2018 focused on Task B Phase B, where our system attempted to find the ideal answer given a question and a collection of relevant snippets of text. We approached this task as an instance of query-based multi-document summarisation, where the ideal answer is the summary to produce.

The BioASQ challenge focuses on a restricted domain, namely biomedical literature. Nevertheless, the techniques developed for our system were domain-agnostic and can be applied to any domain, provided that the domain has enough training data and a specialised corpus large enough to train word embeddings.

[1] http://www.bioasq.org/

	Summary	Factoid	Yesno	List
n	6	2	2	3

Table 1: Value of n (the number of sentences returned as the ideal answer) for each question type.

We were interested in exploring the use of deep learning and reinforcement learning for this task. Thus, Section 2 explains our experiments using deep learning techniques. Section 3 details our experiments using reinforcement learning. Section 4 specifies the settings used in the experiments. Section 5 shows and discusses the results, and Section 6 concludes the paper.

2 Deep Learning

The deep learning experiments followed the general framework introduced by Mollá (2017a), which can be summarised as a regression approach that follows these three main steps:

1. Split the input text into candidate sentences.
2. Score each candidate sentence independently.
3. Return the n sentences that have the highest score.

In all of the experiments reported in this paper, the input text is the set of relevant snippets that are associated with the question. These snippets are pre-processed by splitting them into sentences. The sentences are then scored using the deep learning approaches described below. Then, after all candidate sentences are scored, the top n sentences are returned as the ideal answer. The value of n is determined empirically and it depends on the question type as shown in Table 1. These are the same settings as in Mollá (2017a)'s framework.

Proceedings of the 2018 EMNLP Workshop BioASQ: Large-scale Biomedical Semantic Indexing and Question Answering, pages 22–29

The deep learning experiments predict the score of an input sentence by applying supervised regression techniques. Following Mollá (2017a)'s framework, the training data was annotated with the F1 ROUGE-SU4 score of each individual sentence using the ideal answer as the target. Also following Mollá (2017a)'s framework, the architecture of our system architecture was based on the following main stages:

1. Obtain the word embedding of every word in the input sentence and the question.

2. Given the word embeddings of the input sentence and the question, obtain the sentence and question embeddings.

3. Feed the sentence embeddings and the similarity between the sentence and question embeddings to a fully connected layer and final linear combination. In this stage, as an extension to Mollá (2017a)'s approach, we also incorporated information about the sentence position.

The word embeddings were obtained by pretraining word2vec (Mikolov et al., 2013) on a collection of PubMed documents made available by the organisers of BioASQ. Given a sentence (or question) i with n_i words and vectors of word embeddings m_1 to m_{n_i}, we ran experiments using the following two alternative approaches to obtain the sentence (or question) vector of embeddings s_i:

NNR Mean. Compute the mean of the word embeddings:

$$s_i = \frac{1}{n_i} \sum_{j=1}^{n_i} m_j$$

NNR LSTM. Feed the sequence of word embeddings to bidirectional Recurrent Neural Networks with LSTM cells. We used TensorFlow's LSTM implementation, which is reportedly based on Hochreiter et al. (1997)'s seminal work. More explicitly, the embedding s_i of sentence i is the concatenation of the output of the last cell in the forward chain ($\overrightarrow{h}_{n_i}$) and the first cell in the backward chain ($\overleftarrow{h}_1$):

$$s_i = \left[\overrightarrow{h}_{n_i}; \overleftarrow{h}_1\right]$$

Cell at position t of the forward chain receives its input from the embedding of word at position t and cell at position $t-1$:

$$\begin{aligned}
\overrightarrow{c}_t &= \overrightarrow{f} \odot \overrightarrow{c}_{t+1} + \overrightarrow{i} \odot \overrightarrow{z} \\
\overrightarrow{h}_t &= \overrightarrow{o} \odot \tanh(\overrightarrow{c}_t) \\
\overrightarrow{i} &= \sigma(\overrightarrow{W}_i \cdot m_t + \overrightarrow{U}_i \cdot \overrightarrow{h}_{t+1} + \overrightarrow{b}_i) \\
\overrightarrow{f} &= \sigma(\overrightarrow{W}_f \cdot m_t + \overrightarrow{U}_f \cdot \overrightarrow{h}_{t+1} + \overrightarrow{b}_f) \\
\overrightarrow{o} &= \sigma(\overrightarrow{W}_o \cdot m_t + \overrightarrow{U}_o \cdot \overrightarrow{h}_{t+1} + \overrightarrow{b}_o) \\
\overrightarrow{z} &= \tanh(\overrightarrow{W}_z \cdot m_t + \overrightarrow{U}_z \cdot \overrightarrow{h}_{t+1} + \overrightarrow{b}_z)
\end{aligned}$$

where σ is the logistic sigmoid function, $\odot$ is the element-wise multiplication of two vectors, and $\cdot$ is the dot product between a matrix and a vector.

Cell at position t in the backward chain receives its input from m_t and cell at position $t+1$:

$$\begin{aligned}
\overleftarrow{c}_t &= \overleftarrow{f} \odot \overleftarrow{c}_{t+1} + \overleftarrow{i} \odot \overleftarrow{z} \\
\overleftarrow{h}_t &= \overleftarrow{o} \odot \tanh(\overleftarrow{c}_t) \\
\overleftarrow{i} &= \sigma(\overleftarrow{W}_i \cdot m_t + \overleftarrow{U}_i \cdot \overleftarrow{h}_{t+1} + \overleftarrow{b}_i) \\
\overleftarrow{f} &= \sigma(\overleftarrow{W}_f \cdot m_t + \overleftarrow{U}_f \cdot \overleftarrow{h}_{t+1} + \overleftarrow{b}_f) \\
\overleftarrow{o} &= \sigma(\overleftarrow{W}_o \cdot m_t + \overleftarrow{U}_o \cdot \overleftarrow{h}_{t+1} + \overleftarrow{b}_o) \\
\overleftarrow{z} &= \tanh(\overleftarrow{W}_z \cdot m_t + \overleftarrow{U}_z \cdot \overleftarrow{h}_{t+1} + \overleftarrow{b}_z)
\end{aligned}$$

As is often done with bidirectional LSTM chains, all weights of the parameter matrices in the forward chain $\overrightarrow{W}$, $\overrightarrow{U}$, $\overrightarrow{b}$ are shared among all cells of the forward chain, and all weights of the parameter matrices in the backward chain $\overleftarrow{W}$, $\overleftarrow{U}$, $\overleftarrow{b}$ are shared among all cells of the backward chain. As in Mollá (2017a)'s framework, there are separate sets of parameter matrices for the sentence and for the question.

Mollá (2017a) used all the sentences of the full abstracts as the candidate input. Given that subsequent experiments observed an improvement of results by using the snippets only, our entries to BioASQ 6b used the snippets only. Also, Mollá (2017a) observed very competitive results of a simple baseline that returned the first n sentences. This suggests that sentence position is a useful feature and we consequently incorporated the sentence position as a feature in stage 3.

Given the sentence embedding s_i and question embedding q_i, each obtained either by the mean of embeddings or by applying LSTM chains as described above, and given the sentence position p_i, stage 3 is implemented as a simple neural network with one hidden layer with a relu activation, followed by a linear combination:

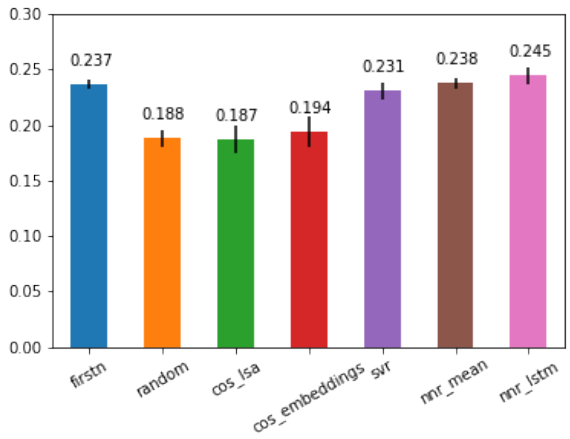

Figure 1: Comparison of deep learning experiments with several baselines. The error bars indicate the standard deviation of 10-fold cross-validation.

$$
\begin{aligned}
r_i &= \max(0, W_r\,[s_i; q_i \odot s_i; p_i] + b_r) \\
score_i &= W_{score} r_i + b_{score}
\end{aligned}
$$

Following Mollá (2017a)'s framework, we used the element-wise multiplication between q_i and s_i as a way to encode the similarity between the question and the input sentence.

Figure 1 compares the results of the deep learning approaches against the following baselines:

Firstn. Return the first n sentences. As mentioned above, this baseline is often rather hard to beat.

Random. Return n random sentences. This is the lower bound of any extractive summarisation system.

Cos LSA. Return the n sentences that have the highest cosine similarity with the question. The feature vectors for the computation of the cosine similarity were obtained by computing $tf.idf$, followed by a dimension reduction step that selected the top 200 components after Latent Semantic Analysis.

Cos Embeddings. Return the n sentences that have the highest cosine similarity with the question. The cosine similarity was based on the sum of the word embeddings (using embeddings with 200 dimensions) in the sentence/question.

SVR. Train a Support Vector Regression system that uses as features a combination of $tf.idf$, word embeddings and distance metrics as described by Mollá (2017a), plus the position of the snippet.

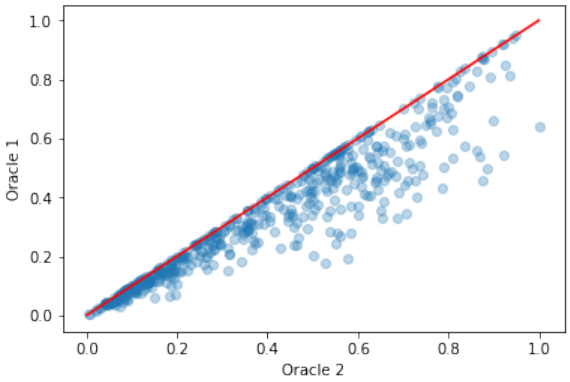

Figure 2: Scatter-plot comparing the F1 ROUGE-SU4 score of two oracles using a random sample of 500 questions.

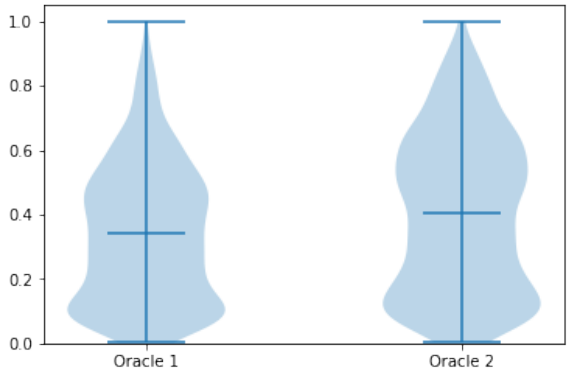

Figure 3: Violin plots of the F1 ROUGE-SU4 scores of two oracles.

Figure 1 shows that the "firstn" baseline is indeed hard to beat, and was matched only by the deep learning frameworks. Of these, the system using LSTM obtained the best results and was chosen for submission to BioASQ.

3 Reinforcement Learning

While the results using deep learning are encouraging, the models are trained on individually annotated sentences, and a summary is obtained by selecting the top k sentences. An upper bound of the results obtained using such an approach would be an oracle that selects the k sentences with highest individual ROUGE scores. Figures 2 and 3 show a comparison between the following two oracles:

Oracle 1. Return the k snippets with highest *in-*

dividual ROUGE score. This is a reasonable upper bound of supervised machine learning approaches such as presented in Section 2.

Oracle 2. Return the combination of k snippets with highest *collective* ROUGE score. In particular, the oracle calculates the ROUGE score of every combination of k snippets, and selects the combination with highest ROUGE score. This is an upper bound of any conceivable extractive summarisation system that returns k snippets.

Figure 2 shows the scatter-plot between oracles 1 and 2. It shows that oracle 1 under-performs oracle 2 in a number of questions. Figure 3 plots the distributions of the ROUGE scores of each oracle side by side, and it clearly shows that the mean of the ROUGE scores of oracle 1 is lower than that of oracle 2.

Reinforcement learning allows to train the system on the ROUGE score of the final summary. This is done by iteratively allowing the system to extract a summary based on its current policy, and then updating the policy based on the feedback given by the ROUGE score of the extracted summary.

The reinforcement learning approach in our system is based on Mollá (2017b)'s method. In particular, the reinforcement learning agent receives as input a candidate sentence and additional context information, and uses a global policy to determine the best possible action (either to select the sentence or not to select it). The global policy is implemented as a neural network with a hidden layer and is trained on a training partition of the development data by applying a variant of REINFORCE (Williams, 1992).

More specifically, the global policy learns a set of parameters θ so that the policy predicts the probability of not selecting the sentence ($Pr(a = 0; \theta)$) by applying the following neural network:

$$\begin{aligned} Pr(a = 0; \theta) &= \sigma(W_h h + b_h) \\ h &= \max(0, W_s s + b_s) \end{aligned}$$

where $\theta = [W_h; W_s; b_h; b_s]$ and the input h is the concatenation of the following features:

1. *tf.idf* of candidate sentence i;
2. *tf.idf* of the entire input text to summarise;
3. *tf.idf* of the summary generated so far;
4. *tf.idf* of the candidate sentences that are yet to be processed;
5. *tf.idf* of the question; and
6. Length (in number of sentences) of the summary generated so far.

The features chosen are such that the global policy has information about the candidate sentence (1), the entire list of candidate sentences (2), the summary that has been generated so far (3), the input sentences that are yet to be processed (4), and the question (5). Experiments by Mollá (2017b) show that this information suffices to learn a global policy. In addition, we added the length of the summary generated so far (6). Our preliminary experiments showed that this additional feature facilitated a faster learning of the policy, and produced better results overall.

The specific algorithm that learns the global policy is presented in Figure 1. The system

Data: `train_data`
Result: θ
1 `sample` $\sim Uniform($`train_data`$)$;
2 $s \leftarrow$ `env.reset(sample)`;
3 `all_gradients` $\leftarrow \emptyset$;
4 `initialise`(θ);
5 `episode` $\leftarrow 0$;
6 **while** *True* **do**
7 $\quad \xi \sim Bernoulli\left(\frac{Pr(a=0;\theta)+p}{1+2\times p}\right)$;
8 $\quad y \leftarrow 1 - \xi$;
9 $\quad$ `gradient` $\leftarrow \frac{\nabla(\text{cross\_entropy}(y, Pr(a=0;\theta))}{\nabla\theta}$;
10 $\quad$ `all_gradients.append(gradient)`;
11 $\quad s, r, done \leftarrow$ `env.step`(ξ);
12 $\quad$ `episode` $\leftarrow$ `episode` $+ 1$;
13 $\quad$ **if** *done* **then**
14 $\quad\quad \theta \leftarrow \theta - \alpha \times r \times \text{mean}($`all_gradients`$)$;
15 $\quad\quad$ `sample` $\sim Uniform($`train_data`$)$;
16 $\quad\quad s \leftarrow$ `env.reset(sample)`;
17 $\quad\quad$ `all_gradients` $\leftarrow \emptyset$;
18 $\quad$ **end**
19 **end**

Algorithm 1: Training by Policy Gradient, where $\theta = [W_h; W_s; b_h; b_s]$.

first chooses one question from the training data (line 1). Then, after randomly initialising the parameters of the global policy (line 4), it iteratively

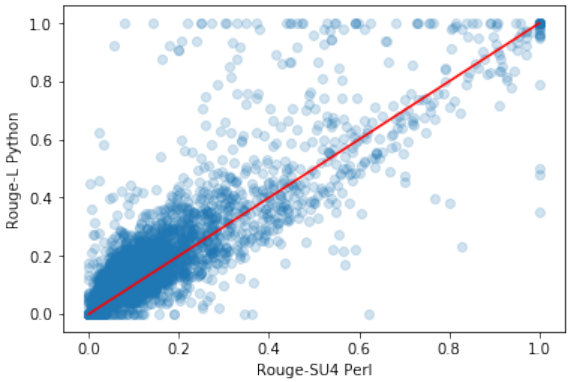

Figure 4: Comparison between the Python and Perl implementations of ROUGE for a random sample of 5000 snippets and their respective ideal answers. The Python implementation uses ROUGE-L. The Perl implementation uses ROUGE-SU4.

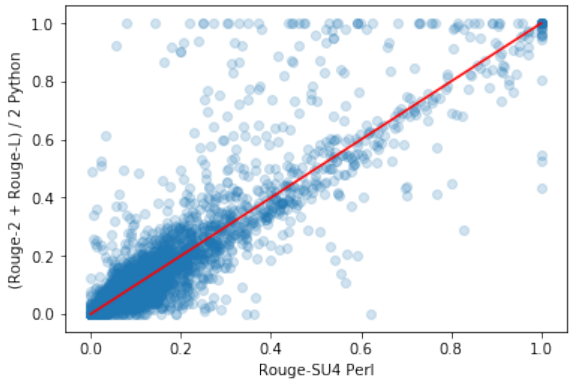

Figure 5: Comparison between the Python and Perl implementations of ROUGE for a random sample of 5000 snippets and their respective ideal answers. The Python implementation uses (ROUGE-2 + ROUGE-L) / 2. The Perl implementation uses ROUGE-SU4.

samples an action from the current global policy plus some perturbation p (line 7) and applies the action (line 11). When all the candidate sentences related to the question have been processed and actioned on (line 13), the resulting summary is evaluated and a reward r produced (done previously, in line 11). Then, the policy parameters are updated by multiplying the mean of all gradients obtained at every step by the reward (line 14), and a new question is selected from the training data (line 15). Each iteration step is called an episode (lines 5 and 12).

The perturbation p forces a wide exploration of the action space during the first episodes and is gradually reduced at every episode according to this formula:

$$p = 0.2 \times 3000/(3000 + \text{episode})$$

3.1 ROUGE Variants

The evaluation scripts of the BioASQ task used the original Perl implementation of ROUGE-2 and ROUGE-SU4 (Lin, 2004). Our experiments aimed to use ROUGE-SU4. Whereas the Perl implementation of ROUGE was used for the deep learning experiments described in section 2, as an implementation decision we used Python's `pyrouge` library for the reinforcement learning experiments. Python's `pyrouge` provides ROUGE-1, ROUGE-2 and ROUGE-L, but not ROUGE-SU4.

Figures 4 and 5 compare the ROUGE F1 scores of the Python libraries against the ROUGE-SU4 F1 score of the Perl implementation. Figure 4 uses the Python library for ROUGE-L, and Figure 5 uses the Python library for the mean between ROUGE-2 and ROUGE-L. We can observe some noise in the correlation between the Perl and Python implementations, but in general the mean between ROUGE-2 and ROUGE-L was a better approximation of the Perl implementation of ROUGE-SU4.

In general, we observed lower results of the Python versions of ROUGE-L and ROUGE-2 compared with the Perl versions. On the light of this, we strongly advise always to specify the implementation of ROUGE being used, since the results produced by different versions may not be comparable.

Kernel	C	gamma
rbf	1.0	0.1

Table 2: Settings used in the SVR experiments.

4 Settings

The snippets were split into sentences using NLTK's sentence tokeniser.

The $tf.idf$ information used for the baselines was computed using NLTK's TfidfVectorizer. As in the system by Mollá (2017a), this vectoriser was trained using a collection of text consisting of the questions and the ideal answers.

The SVR experiments were implemented using Python's `sklearn` library and used word embeddings with 200 dimensions. The specific settings of the SVR model are shown in Table 2.

The deep learning experiments were imple-

System	Batch	Dropout	Epochs
Mean	4096	0.4	5
LSTM	4096	0.8	10

Table 3: Settings used in the deep learning experiments.

mented using TensorFlow's libraries. The specific details of the model are:

- Dimension of embeddings: 100
- Length of the LSTM chain: 300. Sentences with more than 300 words were truncated.
- Dimension of $\overrightarrow{h}$ and $\overleftarrow{h}$: 100
- Number of cells in the final hidden layer: 50

Table 3 shows the training settings used by the deep learning architectures. These settings were obtained empirically in a fine-tuning stage.

The reinforcement learning experiments were implemented in TensorFlow. Due to hardware constraints, the reinforcement learning approach only processed the first 20 sentences. The specific details of the architecture of the global policy are:

- Number of cells in the hidden layer: 200

The global policy was trained using a training partition and evaluated on a separate partition. The global policy parameters that generated the best results in the evaluation partition were used for the final runs to BioASQ.

5 Results

We submitted 5 runs for each batch as listed below.

MQ-1: Return the first n sentences. This is the Firstn baseline described in Section 2.

MQ-2: Return the n sentences with highest cosine similarity with the question. This is the Cos Embeddings baseline described in Section 2.

MQ-3: Return the n sentences according to the SVR baseline described in Section 2.

MQ-4: Score the sentences using the LSTM-based deep learning approach as described in Section 2.

MQ-5: Apply reinforcement learning as described in Section 3, with the variations described below.

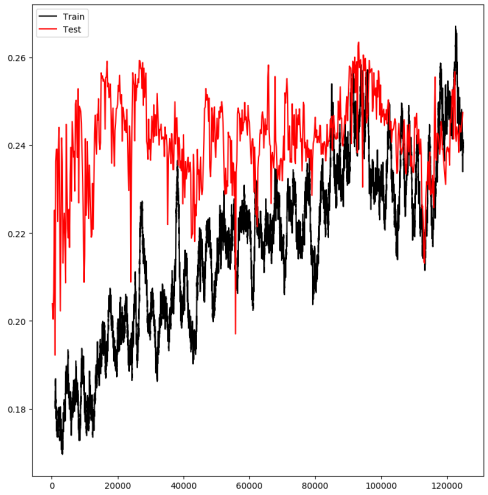

Figure 6: Reinforcement Learning model trained for batch 1. The reward (y axis) is ROUGE-L.

The policy trained for the reinforcement learning approach of run MQ-5 varied in several of the batches. In particular, the first batch was trained on ROUGE-L while batches 2 to 5 were trained on (ROUGE-2 + ROUGE-L) / 2. Also, to test the impact of different initialisation settings, we trained the system twice and generated two separate models. Batch 2 used one model, whereas batches 3 to 5 used the other model.

Figures 6 and 7 show the evolution of the results of the policies during the training stage for batches 1 and 2. We observe some differences during training, but in general the best model on the test set achieved a ROUGE score between 0.25 and 0.26.[2] This is higher than the results reported by Mollá (2017b), who reported a ROUGE score of about 0.2. The likely cause of the improvement in the results is the inclusion of the length of the summary generated so far in the context available by the policy.

Figure 8 shows the results of the submissions to BioASQ. In general, the deep learning approach (MQ-4) achieved the best results. While the "first n" baseline (MQ-1) was fairly competitive and outperformed some of the runs of other participants to BioASQ, the baseline was not as strong as reported by Mollá (2017a) on BioASQ5b.

We can also observe that the evaluation re-

[2]The training stage for batches 3 to 5 achieved a best result slightly over 0.26.

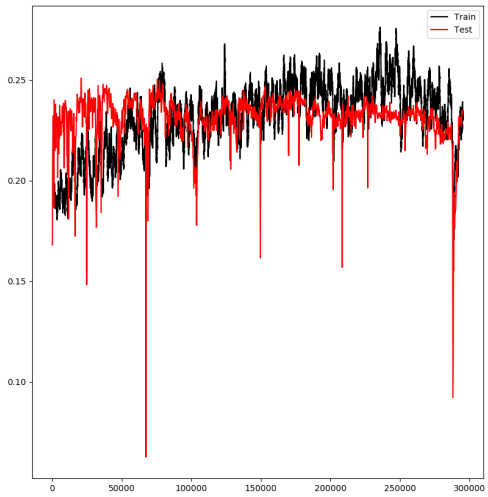

Figure 7: Reinforcement Learning model trained for batch 2. The reward (y axis) is (ROUGE-2+ROUGE-L)/2.

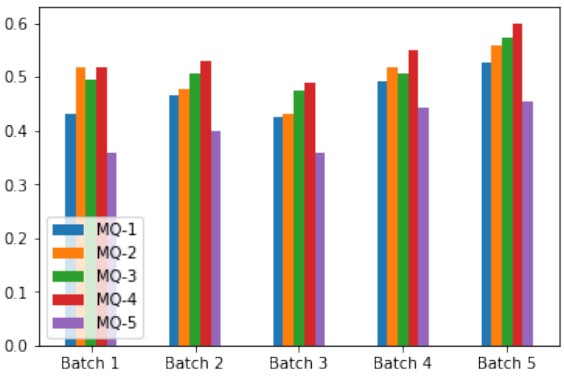

Figure 8: ROUGE-SU4 results of the BioASQ runs.

sults of the submissions to BioASQ give higher ROUGE scores than those obtained in our experiments, both using the original Perl implementation of ROUGE, and the Python version. In fact, all of the runs except for MQ-5 reported better BioASQ results than our experiments with the oracles. This is worth investigating.

The runs using reinforcement learning (MQ-5) gave worse results than the other runs. The cause for this is also worth investigating, especially considering that, in our experiments, the results of the reinforcement learning approach were very competitive compared with the results of the other approaches.

6 Conclusions

In this paper we have described the deep learning and reinforcement learning approaches used for the runs submitted to BioASQ 6b, phase B, for the generation of ideal answers.

The deep learning approach used a supervised regression set-up to score the individual candidate sentences. The training data was generated by computing the ROUGE score of each individual candidate sentence, and the summary was obtained by selecting the top-scoring sentences. The results of the deep learning runs outperformed all other runs.

The reinforcement learning approach trained a global policy using as a reward the ROUGE score of the summary generated by the policy. The results of our experiments were very competitive but the submission results were lower than those of the other runs.

Further work will focus on the refinement of the reinforcement learning approach. In particular, further work will include the addition of a baseline in the training of the policy, as it has been shown to reduce the variance and to speed up the training process. Also, the architecture of the neural network implementation of the policy will be revised and enhanced by incorporating a more sophisticated model.

Further work on the deep learning runs will focus on the incorporation of more complex models. For example, preliminary experiments seem to indicate that a classification-based approach could outperform the current regression-based approach. Also, it is expected that a sequence-labelling approach would produce better results since the candidate sentences would not be processed indepen-

dently.

References

Sepp Hochreiter, Jrgen Schmidhuber, Sepp Hochreiter, Jrgen Schmidhuber, and Jrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–80.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *ACL Workshop on Tech Summarisation Branches Out*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*, pages 1–12.

Diego Mollá. 2017a. Macquarie University at BioASQ 5b — query-based summarisation techniques for selecting the ideal answers. In *Proc. BioNLP2017*.

Diego Mollá. 2017b. Towards the use of deep reinforcement learning with global policy for query-based extractive summarisation. In *Proceedings of the Australasian Language Technology Association Workshop 2017*, pages 103–107.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artiéres, Axel-Cyrille Ngonga Ngomo, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, and Georgios Paliouras. 2015. An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16(1):138.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.