Transformer Models for Question Answering at BioASQ 2019

Michele Resta(✉), Daniele Arioli, Alessandro Fagnani, and Giuseppe Attardi

Dipartimento di Informatica, Università di Pisa, Pisa, Italy
{m.resta5,d.arioli,a.fagnani}@studenti.unipi.it, attardi@di.unipi.it

Abstract. We describe our experiments in building a system to tackle task B of the BioASQ 2019 challenge on semantic question answering. We built separate systems to handle the five different types of questions in the dataset. We explored using transformer-based models using both ELMo, BERT and BioBERT. For the *yesno* questions, the results of our submissions using BERT ranked first in batches 3 and 4, while second best in batch 5.

Keywords: Question-answering · Transformer · ELMo

1 Introduction

Along the years the BioASQ challenge has been growing in popularity as well in the difficulty of the tasks to perform. The three tasks of the BioASQ 2019 challenge [7] concern biomedical semantic indexing and question answering.

Task *B* on Biomedical Semantic Question Answering requires creating an automated system capable of responding to a set of biomedical questions with relevant concepts, articles, snippets, and RDF triples, from designated resources, as well as exact and 'ideal' answers. Questions are divided into various types according to the expected answer (yes/no, a single fact, a list of entities, or a summary). Systems often exploit different strategies to address each type of questions.

2 Dataset

The BioASQ training dataset for Task 7b consists of 2747 questions in the biology and medical domain. All questions were constructed by biomedical experts from around Europe. Each dataset item consists of several fields, the most relevant of which are:

- *type:* type of the question;
- *exact_answer:* exact answer (absent in summary questions);
- *ideal_answer:* an answer summarizing the most relevant information;

P. Cellier and K. Driessens (Eds.): ECML PKDD 2019 Workshops, CCIS 1168, pp. 711–726, 2020.
https://doi.org/10.1007/978-3-030-43887-6_63

– *documents:* PubMed articles relevant to the question. The supplied answer snippets are extracted from these documents.

Questions are classified into the following types:

– *yes-no*: systems must provide a "yes" or "no" answer;
– *list*: systems must provide a list of entity names (e.g. a list of gene names);
– *factoid*: similar to list answers, these require as an answer a single entity name (e.g. a disease, drug, or gene), a number, or a similar short expression. Differently from list questions though there are lesser entities in the answer: often, only a single entity or up to 3–4 in a few cases;
– *summary*: these questions must be answered by producing a short text summarizing the most relevant information.

Besides answers of the types described above, submissions may provide an *ideal answer*: a single text paragraph that best summarizes the most relevant information answering the question.

For each question, a list of *snippets* is provided. Snippets are short texts that are expected to contain the information needed to answer the corresponding question. However, some of them may not be useful for extracting the answer.

Table 1 shows in detail the composition of the training dataset.
Most of the questions have 1 to 15 associated snippets while only a few of them have a larger number of snippets (>40).

Table 1. Dataset composition

Question type	Number	Dataset %
List	556	20,2
Summary	667	24,2
Yesno	745	27,1
Factoid	779	28,3
Total questions	2747	100

Yesno Questions. There is a great imbalance in the answers to these questions: 82% of the answers are "*yes*" and only 18% have a "*no*" answer.
List Questions. List answers contain between 1 and 38 entities, but over 80% have less than 10 elements.
Factoid Questions. The answer to a factoid question might be phrased differently, therefore for some questions, there may be multiple answers with different wording. This is important since it may affect the scoring, which is based on MRR (Mean Reciprocal Rank).
Summary Questions. For these types of questions no *exact answer* is requested; but the system is expected to provide just a kind of *ideal answer*.

3 Models

Since the challenge expects different types of answers according to the type of questions, we developed specialized systems for each of them. We explored several solutions, some of which had to be discarded because they did not produce good results. Simpler approaches were tried first, such as Sentiment Analysis or information retrieval based on a bag-of-words model, and more complex ones were attempted as soon as the simpler ones showed unsatisfactory results.

In the following sections, a detailed description of the models employed for the competition is presented. The discussion is organized into sections corresponding to the type of questions to handle.

3.1 Models for YesNo Questions

In order to answer this type of questions, we exploited different versions of word embeddings. We explored embeddings created from the BioASQ dataset itself using *word2vec* [13]. Given the small size of the dataset however this type of embeddings showed poor results.

Embeddings of ELMo, BERT, and ELMo-Pubmed (described below) have been extracted using Flair [3] since it offers the same pre-trained weights and simple programming interfaces.

ELMo. Elmo [15] is a deep contextualized word representation that models both complex characteristics of word use (e.g., syntax and semantics), and how these uses vary across linguistic contexts (i.e., to model polysemy). These word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus.

This model consists of two layers of 4096 LSTM units. Of these units, half handle the forward language modeling, and the remaining the backward language modeling.

We exploited a pre-trained model on the "One billion word Benchmark" [5].

Figure 1 shows the network architecture.

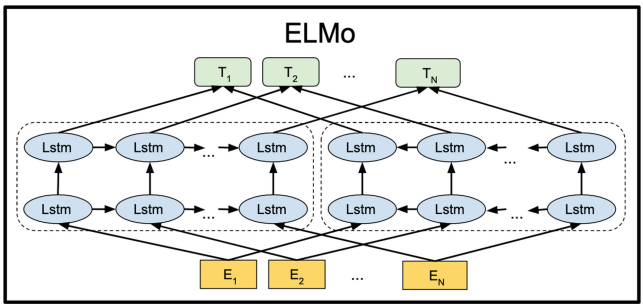

Fig. 1. Elmo architecture

ELMo-Pubmed. This model is identical to the one described in Sect. 3.1, except that it is trained on PubMed abstracts.

BERT. BERT [6] stands for Bidirectional Encoder Representations from Transformers. The *BERT Large* model consists of a stack of 24 Transformer encoders. Each encoder incorporates an attention mechanism to help the model focus on the most relevant parts of the input.

In this model, the attention mechanism is a multi-headed one with 16 attention heads.

The model was trained on the entire Wikipedia and BookCorpus [18] for a total of 1 million update steps.

BERT architecture is showed in Fig. 2.

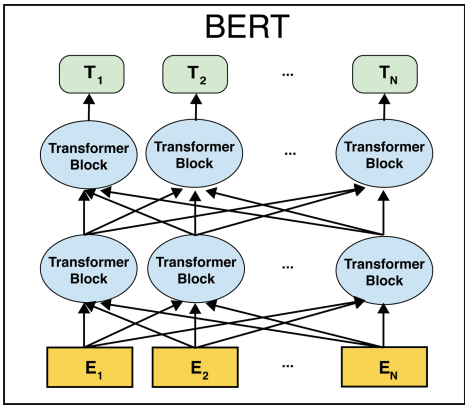

Fig. 2. Bert architecture

BioBERT. The BioBERT model [11] has the same architecture as Sect. 3.1 but it was trained on PubMed abstracts.

After evaluating the performance of this model on the validation set we decided not to use it for the yes-no questions.

Data Pre-processing. Before feeding the models with data, the following pre-processing steps were performed on all questions and snippets:

- removal of spurious newlines,
- tokenization.

Each of the pre-trained models handles tokenization differently. All models are capable of handling case sensitive text, so there was no need to lowercase inputs. The inputs were normalized by each model in its own way, and the resulting embeddings are of fixed size.

To extract the embeddings from the BioBERT model we prepared the input concatenating the question, a separator token, and its snippets. However, since the max token length for BioBERT is 512, we built multiple inputs by pairing a question with each sentence from the snippets. Scispacy [14] was used to perform sentence splitting on the snippets.

Classifier Inputs. Given a dataset consisting of questions $\{Q_n, n < N\}$, their corresponding answer snippets $\{S_{n,k}, k < K\}$ and outputs $\{y_n, n < N\}$, a training set is created consisting of inputs $\{x_i = \langle Q_n, S_{n,k} \rangle, n < N, k < K\}$ and outputs $\{y_i\}$, where i is the index n of the Q_n corresponding to x_i. In other words we create pairs of each question with all its related snippets and assume that they would all have the same answer. This is realistic, since all snippets are assumed to have been chosen in the previous stage of question answering as candidates for containing the answer.

Questions and snippets are transformed into a vector representation by using a language model as follows.

For all models except the one based on BioBERT, for each input pair $x_i = < Q_i, S_i >$ we pass separately through the language model the sequence of *tokens* from the question Q_i and those from the snippet S_i.

By means of functions from the Flair library, we extract the embeddings from all layers of the language model for each token of the input sequence. The vectors for the question tokens are added together and similarly those from the snippet, obtaining fixed length vectors irrespective of the length of the sentences. A *mean*-pooling is applied to the question vectors and snippet vectors so produced for each layer, to obtain the final vectors representing the i-th question, $Emb(Q_i)$, and the i-th snippet, $Emb(S_i)$. These two vectors are concatenated to obtain vector

$$[Emb(Q_i); Emb(S_j)]$$

to be used as input to the classifier described below. We tested also using different types of pooling (*min* and *max*), but they led to poorer results with respect to *mean*-pooling. Embeddings vectors computed through BioBERT did not provide improvements with respect to the previous models, therefore we did not employ them in our submissions for the yes-no questions. Even though BioBERT is trained on the biomedical domain, the fact that has fewer parameters than BERT_{LARGE} is probably the cause of performance improvements lack.

Classifiers. Since the rules of the BioASQ challenge allow participants to send up to 5 submissions, we trained 5 different classifiers with different embeddings and different architectures.

Before training, the dataset yes-no ratio was re-balanced as shown in the column *Yes-No* in Table 3 to avoid bias towards "yes" answers.

The classifier consists of fully connected feed-forward networks with the architecture described in Fig. 3.

The classifier was implemented using Keras [4] on a TensorFlow [2] backend.

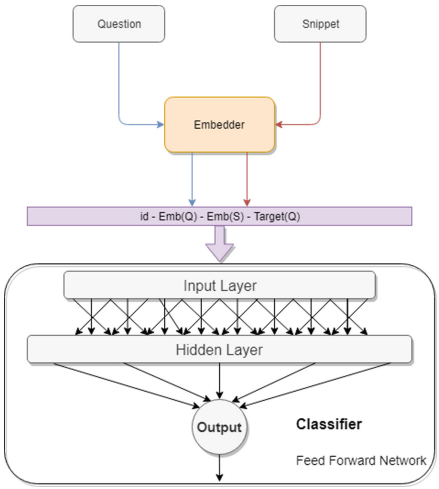

Fig. 3. Yes-No model with feed-forward network. The activation functions are *tanh* for the input and hidden layers and the *sigmoid* for the output layer. The vector dimensions are shown in Table 2.

The hyperparameters for each model were tuned by performing grid searches, in order to find the best combination of:

- number of hidden layers $\rightarrow [1 \dots 5]$
- size of vectors in the first layer $\rightarrow [50 \dots 120]$
- size of vectors in the hidden layers $\rightarrow [50 \dots 150]$
- type of optimizer $\rightarrow [Sgd, RmsProp, Adam]$
- activation functions for the hidden layers $\rightarrow [tanh, ReLU]$
- activation function for the input layer $\rightarrow [tanh, ReLU]$

We used 80% of the training set as development set and the remaining 20% as the validation set. We also performed a *4-fold cross-validation* on the development set in order to select the best models for each task.

We then trained each classifier on the development dataset and tested it on the validation set, to estimate the ability of the model to generalize to unseen data.

The results of the grid searches are summarized in Table 2, while Table 3 shows the scores on yes-no questions on the validation and test datasets.

Note. By inspecting the answers of the models and the training curves as well, we noticed an increase in the validation score respect to training score. Since in the 4-fold CV all the models had less data, more epochs were needed to fit. During training we noticed a classifier tendency to overfit. To limit this problem, we implemented an early stopping technique with patience equal to 4.

Table 2. Grid search results.

System	**QA-1**	**QA-2**	**QA-3**	**QA-4**	**QA-5**
Embedding	Elmo-pubmed		BERT_*LARGE*		
Pooling	Mean				
Num h. layers	1				
Neurons h. layer	120	120	120	120	80
Neurons 1-st layer	90	90	90	90	50
Act. hidden	tanh				
Act. input	tanh				
Optimizer	RMSProp				
Loss	Binary-crossentropy				

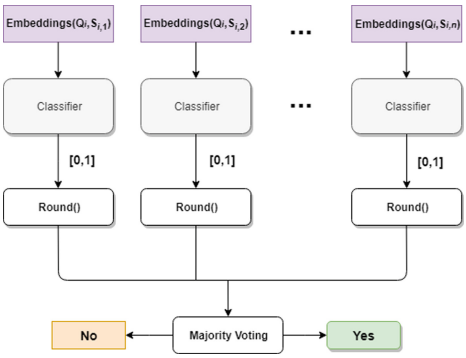

Fig. 4. Ensemble of classifiers for the Yes/No Model. The result is obtained by majority voting

Ensemble Classifier. A ensemble of K classifiers is used to classify each pair of question/snippet $< Q_i, S_i >$ for the same Q_i and the answer is obtained through a majority vote among these classifiers. More precisely, the ensemble outputs:

- *model QA-1*: "yes" if $(votes \geq floor(K/2))$
- *other models*: "yes" if $(votes > floor(K/2))$

The last system, QA-5, was trained on an augmented dataset constructed by manually annotating the yes-no datasets of the previous editions of the BioASQ challenge. Since previous models appeared biased towards giving positive answers, we chose to add to the training set just the questions with negative answers.

3.2 Models for List Questions

To answer this type of questions we explored two approaches: one based on frequency and *tf-idf* and one based on the analysis of the dependency trees of questions and snippets.

Table 3. Scores before retraining.

System	Yes-No%	Epochs	4-fold VL score	Held-out TS score
QA-1	50-50	40	0.9861	0.9612
QA-2	50-50	40	0.9861	0.9564
QA-3	50-50	50	0.9840	0.9668
QA-4	50-50	50	0.8801	0.9429
QA-5	60-40	50	0.8201	0.8455

The processing pipeline to produce the list of entities for the answer consists in the following steps:

- *data pre-processing*: stop words such as "list", punctuation and parentheses were removed from the input texts;
- *entity extraction*: entities from both questions and snippets were extracted using scispacy [14] and then converted to lowercase;
- *ranking*: the extracted entities were scored with the metrics discussed below;
- *filtering*: unrelated entities were dropped;
- *rank-boosting:* entities that were more likely to be an answer received an increase in their score;
- *list-trimming*: entities with a score below a given threshold are discarded.

Entity Extraction. We explored two approaches for entity extraction. The first one is the simplest: the scispaCy mention detector was used to extract entities from questions and snippets. We used the en_core_sci_md model available from the scispaCy website.[1]

This model was trained on a collection of biomedical data, recognizes a wide variety of entity types, has a vocabulary of 101,678 tokens and includes 98,131 word vectors.

Entities that are already present in the question were dropped, assuming that a question will typically look for answers not already known and hence terms appearing in the question are unlikely to be part of an answer.

The second approach is based on the analysis of parse trees. First we determine the *main entity* in the question, that is the entity required as answer to the question. For example, consider the following question: "*Which miRNAs could be used as potential biomarkers for epithelial ovarian cancer?*". The main entity is *miRNAs*, since a list of entities of this type is the required answer.

The next step splits the snippets into sentences and then it parses them with scispaCy. From the parse trees, we extract the entities that:

- are child nodes of an entity similar to the main entity;
- are child nodes of a verb that is the same of the main one;

[1] https://allenai.github.io/scispacy/.

– are child nodes of an entity similar to the father of the main entity in the question's parse tree;
– are child nodes of an entity similar to the verb of the main entity's father in the question's parse tree.

The similarity between two entities is based on string comparisons. An approach with distances between vector representation has been taken into account but it was not possible to finish it before the deadline of the last batch.

All the resulting entities are merged with those obtained with the first approach described at the beginning of this section. The whole list is sent to the ranking phase.

Ranking. In this phase all elements of the entity list are ranked based on the frequency of occurrence in the snippet list. So e.g. the entity *genes* appear three times in question's snippets, then $Score(genes) = 3$.

We investigated also a different type of scoring scheme based on *tf-idf*. In this case the collection of documents was composed of all snippets in the training set, and the terms to be scored are the elements of the list created in the previous steps.

The entities list is then sorted according to the scores.

Filtering. Filtering of entities is based mainly on Part of Speech tags from scispacy. If the considered entity is composed of a single word, and this word is a verb or an adjective we delete this element from the list. The same is done if the entity is composed of 2 words and POS are verbs and adposition. In this way we discard entity like *"associated with"*.

Score Boosting. In this phase the score of each entity in the resulting list is increased according to the context of the question.

First, there is an ontology checking, where the entities are boosted if they appear in a local database of the biomedical domain (the choice of the database depends on the main entity of the question). Available databases are: *bacteria* [1], *viruses* [1], *genes* [9,10], *drugs* [8], *proteins*, *human symptoms*, *human organs*.

We check if the main entity concern one of the domains above. If it happens, all the entities of the list are searched in the databases and, if an entity is found, its score is increased. This process increases the probability of finding the entities required for the answer with greater rank in respect to those that are unrelated.

In the presented code the score of an entity is increased by the mean of all the ranks $>= 1$.[2]

After the ontology checking, **tf-idf** boosting is performed. Thanks to the `tfidf()` function we can extract the value of the entities based on an analysis of all the snippets in the dataset. With some analysis and test, we found that

[2] This is only an empirical choice due to the short time of development, but it could easily changed with more in-depth work.

entities which have a value between 3 and 4 have more chances of being a good answer to the question.

Based on these considerations, we boosted another time the scores of these entities. In the presented code this function doubles the entity's rank. This choice is empirical and could be refined through a more accurate and in-depth analysis of the problem.

Normalization. Before the trimming phase, scores of all the entities are normalized in the [0,1] range with the formula:

$$score' = \frac{score - min\_score}{max\_score - min\_score}$$

Where *score* is the actual score of the entity and *max* and *min* score are the maximum and minimum score of entities in the list.

List Trimming. The last phase of the process is based on a threshold. All entities that have a score below this threshold are removed. Anyway, if the list exceeds the 100 elements then only the first 100 are returned as the answer.

The threshold has been determined experimentally by running the entire pipeline with different thresholds and evaluating the final Mean F1 Score, since it is the official metrics for these types of questions.

In the current implementation, the threshold is fixed at 0.15.

3.3 Model for Factoid Questions

The approach used for this kind of answer was to exploit the very well known BERT [6] and how it is used for the challenge SQuAD [17]. As a matter of fact, SQuAD is, to some extent, a factoid task.

We employed the pre-trained BioBERT [11] model and the technique of *fine-tuning*, in order to better fit the results to our dataset.

Pre-processing and Augmentation. We needed to build the training data in the form of a SQuAD task. To do so, it was necessary to create a unique context, formed by the concatenation of the snippets. The exact answer had to be inside of this context. Indeed, SQuAD models need also the index of the answer inside the context, found by *lower-casing* both the answer and the context. At the end of the process, a JSON file is created, where each question has this format, e.g.:

```
{"qas": [{
    "question": "Which is the target protein
                of the drug nivolumab?",
    "id": "56af9f130a360a5e45000015",
    "answers": [{"text": "plasma membrane",
                "answer_start": 827}],
    "is_impossible": "false"}],
"context": "..."
}
```

The field *is_impossible* is used to define a test/validation question (when is "false", means that there is no exact answer inside the context).

As already mentioned in the description of the dataset, some factoid questions have more than one answer, to avoid misinterpretation. Each of them has been cloned a number of times equal to the provided exact answers, but with a different id.

For the final training, the dataset was augmented with the second and the fifth version of the BioASQ challenge data, available thanks to [11] and the BioASQ team itself. Unfortunately, the two datasets were the only others available, so there was no possibility to integrate more training sets of past editions.

Post-processing and Evaluation. Up to 5 possible answers are allowed in a submission and the evaluation metrics for the factoid task is the MRR (Mean Reciprocal Rank). This means that the system should aim to produce exact answers with a confident estimate since an answer at a lower ranking position is highly penalized.

By analyzing the answers returned by the model on the validation set, we decided to introduce two post-processing steps:

- *Parentheses' fix*: correcting or removing a sentence that has unbalanced open or closed parenthesis.
- *Dashes' fix*: if there is an answer with a dashed word in the first position (e.g."S-adenosyl-L-methionine"), a variant without dashes is added in the last ranking position (in order to minimize the misinterpretation of the answer given by the system).

Due to the complexity of the model, the evaluation of the fine-tuning was done only by varying the number of epochs (and only with a very small batch size). A validation set of 127 answers was created from the given training set, in order to validate the model.

3.4 Model for Summary and Ideal Questions

Since there are no exact answers in the summary questions, they have been treated with the ideal answers of the entire dataset. Similarly to the factoid system, BERT, BioBert and SQuAD approach have been used also for the generation of the ideal answers.

Nevertheless, the answer does not have to appear (always) inside the context, but it has to be a completely "original" summary of the paragraph, related to the question. OpenAI's GPT architecture [16] is promising, but the released model has fewer parameters than BERT and its later more sophisticated version, GPT-2, has not been released by its authors.

Our idea was to keep exploiting BERT model, in order to find the most attentive part of the context.

Pre-processing and Augmentation. In order to select as answer the most relevant part of the text and at the same time follow the structure of the SQuAD approach, we exploit the same evaluation metric used in the BioASQ ideal answer's task, ROUGE [12]. In fact, the `answer_start` index is found by calculating the ROUGE score between the returned answer and all the sentences of the snippets. The same idea was also used in the testing/validating phase. In addition, the dataset of the second and the fifth version of the BioASQ challenge were also added to the training, as we did for the factoid question model.

Post-processing and Evaluation. The post-processing phase aims mainly at finding the proper sentence within the text of the full answer returned by the model, and ROUGE metric has been used between all the sentences. Our model achieved a ROUGE-2 score of 0.4137 on the validation set. We tried increasing the number of epochs, but the score did not improve.

We tried also to build the answer starting from the factoid model but the results were worse (ROUGE-2 score of 0.0345).

4 Results

The competition benchmarks were split into 5 batches. We participated in batch 3, 4 and 5 shown in Tables 4, 5 and 6 respectively. Below results are reported for each of the submissions and the ranks obtained against other participating systems.

Table 4. Batch 3 results. Best result indicates the best scoring system across all participant systems. The score is the average of each question type. Highest scores are in bold.

Type	Metric	System					Best result
		QA-1	QA-2	QA-3	QA-4	QA-5	BioBERT-DMIS
Yes/No	Accuracy	**0.8261**	**0.8696**	–	–	–	0.6087
	F1 Yes	**0.9000**	**0.9231**	–	–	–	0.7429
	F1 No	**0.3333**	**0.5714**	–	–	–	0.1818
	Macro-F1	**0.6167**	**0.7473**	–	–	–	0.4623
List	Mean Prec.	–	–	–	–	–	0.4267
	Recall	–	–	–	–	–	0.3058
	F-Measure	–	–	–	–	–	0.3298
Factoid	Strict acc.	–	–	–	–	–	0.6207
	Lenient acc.	–	–	–	–	–	0.4724
	MRR	–	–	–	–	–	0.4267
Average		0.2055	0.2491	–	–	–	**0.4215**

Table 5. Batch 4 results. Best result indicates the best scoring system across all participant systems. The score is the average of each question type. Highest scores are in bold.

Type	Metric	System					Best Result
		QA-1	QA-2	QA-3	QA-4	QA-5	BioBERT-DMIS
Yes/No	Accuracy	0.7391	0.7391	**0.8261**	**0.8696**	-	0.7391
	F1 Yes	**0.8421**	**0.8421**	**0.8889**	**0.9143**	–	0.8125
	F1 No	0.2500	0.2500	**0.6000**	**0.7273**	–	0.5714
	Macro-F1	0.5461	0.5461	**0.7444**	**0.8208**	–	0.6920
List	Mean Prec.	0.1442	0.1442	0.1442	0.1442	–	**0.4841**
	Recall	**0.6268**	**0.6268**	**0.6268**	**0.6268**	–	0.5051
	F-Measure	0.2163	0.2163	0.2163	0.2163	–	**0.4604**
Factoid	Strict acc.	0.2059	0.2059	0.2059	0.2059	–	**0.882**
	Lenient acc.	0.3824	0.3824	0.3824	0.3824	–	**0.8235**
	MRR	0.2730	0.2730	0.2730	0.2730	–	**0.6912**
Average		0.3451	0.3451	0.4112	0.4367	–	**0.6145**

4.1 Ideal Answers

Table 7 summarizes the results obtained by our systems in the ideal answer type of questions. The Manual Scores were not yet published at the time of this writing.

Table 6. Batch 5 results. Best result indicates the best scoring system across all participant systems. The score is the average of each question type. Highest scores are in bold.

Type	Metric	System					Best result
		QA-1	QA-2	QA-3	QA-4	QA-5	BioBERT-DMIS-3
Yes/No	Accuracy	0.5429	0.5429	0.6857	0.7143	0.8000	**0.8286**
	F1 Yes	0.6800	0.6800	0.7556	0.7727	0.8293	**0.8500**
	F1 No	0.2000	0.2000	0.5600	0.6154	0.7586	**0.8000**
	Macro-F1	0.4400	0.4400	0.6578	0.6941	0.7939	**0.8250**
List	Mean Prec.	0.1713	0.1713	0.1713	0.1713	0.1713	**0.5653**
	Recall	**0.5873**	**0.5873**	**0.5873**	**0.5873**	**0.5873**	0.4131
	F-Measure	0.2537	0.2537	0.2537	0.2537	0.2537	**0.4619**
Factoid	Strict acc.	0.0857	0.0857	0.0857	0.0857	0.0857	**0.2857**
	Lenient acc.	0.1714	0.1714	0.1714	0.1714	0.1714	**0.4286**
	MRR	0.1152	0.1152	0.1152	0.1152	0.1152	**0.3452**
Average		0.2696	0.2696	0.3422	0.3543	0.3876	**0.5440**

Table 7. Rouge scores on Ideal Answers. Best result indicates the best scoring system.

Batches	Automatic Scores		Manual Scores				Best Result (MQ-4)	
	Rouge-2	Rouge-SU4	Readability	Recall	Precision	Repetition	Rouge-2	Rouge-SU4
Batch 4	0.3511	0.3638	–	–	–	–	**0.5177**	**0.5246**
Batch 5	0.4265	0.4275	–	–	–	–	**0.5035**	**0.5070**

5 Conclusions

Our participation in task B of the BioASQ 2019 competition focused mainly on answering the *yesno* questions.

We started developing our system just 5 days before the scheduled release of the first test set. Despite the time constraints, our yes-no answering systems achieved top scores ones in Batch 3 and Batch 4 and ranked second in Batch 5. We exploited an ensemble of classifiers, whose input was obtained by processing questions and snippets though transformer-based language models. Our approach was incremental, we tried to exploit simpler word embeddings first, and as soon as they showed limitations due to task complexity, we moved to more powerful embeddings. Contextual embeddings obtained from deep models like ELMo and transformer-based language model (BERT) provided better results thanks to the built in attention mechanisms and to their ability to capture long term dependencies. These deep models are the core of our submitted systems, BERT specifically is employed for three out of four question types. Questions and snippets were processed separately. Concatenating them before processing might allow exploiting better the attention mechanism.

Submissions for factoid and list achieved lower scores. The latter type of question were more challenging given the presence of duplicate entities, lexical variation among entities, and the number of entities to be returned as answer. We noticed by inspecting a number of question that sci-spacy was able to extract almost all the required answer items, among *noisy* entities. Based on this empirical observation, we decided to try the described approach instead of a more computationally demanding fine-tuning of BERT. We had several ideas under development for improving these answers that we could not complete in time for the submission deadline: using tf-idf as main ranking metrics, removing duplicated entities using a clustering algorithm on their vector representation, improved entity extraction from parse trees.

The overall ranking of our submissions on all question types was fourth in Batch 4 and third in Batch 5.

The approach to semantic question answering using classifiers on top of transformer-based language models has proved quite effective and there are still margins for improvements.

Acknowledgments. The experiments were run on a server equipped with 4 Nvidia P100 GPUs, partly funded by the University of Pisa under grant "Grandi Attrezzature 2016".

References

1. GBIF—The Global Biodiversity Information Facility (2019). https://www.gbif.org/
2. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). Software available from http://tensorflow.org/
3. Akbik, A., Blythe, D., Vollgraf, R.: Contextual string embeddings for sequence labeling. In: 27th International Conference on Computational Linguistics, COLING 2018, pp. 1638–1649 (2018)
4. Chollet, F., et al.: Keras (2015). https://github.com/fchollet/keras
5. Chelba, C., et al.: One billion word benchmark for measuring progress in statistical language modeling (2014)
6. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR abs/1810.04805 (2018). http://arxiv.org/abs/1810.04805
7. Balikas, G., Krithara, A., Partalas, I., Paliouras, G.: BioASQ: a challenge on large-scale biomedical semantic indexing and question answering (2015)
8. Wishart Research Group. https://www.drugbank.ca/
9. HGNC: HUGO Gene Nomenclature Committee at the European Bioinformatics Institute. http://www.genenames.org/
10. MacArthur Lab: Lists of gene lists. https://github.com/macarthur-lab/gene_lists
11. Lee, J., et al.: BioBERT: a pre-trained biomedical language representation model for biomedical text mining. arXiv preprint arXiv:1901.08746 (2019)
12. Lin, C.Y.: ROUGE: a package for automatic evaluation of summaries. In: Text Summarization Branches Out (2004)

13. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. CoRR abs/1310.4546 (2013)
14. Neumann, M., King, D., Beltagy, I., Ammar, W.: ScispaCy: fast and robust models for biomedical natural language processing. CoRR abs/1902.07669 (2019). http://arxiv.org/abs/1902.07669
15. Peters, M.E., et al.: Deep contextualized word representations. In: Proceedings of NAACL (2018)
16. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI Blog **1**, 8 (2019)
17. Rajpurkar, P., Jia, R., Liang, P.: Know what you don't know: unanswerable questions for squad. CoRR abs/1806.03822 (2018). http://arxiv.org/abs/1806.03822
18. Zhu, Y., et al.: Aligning books and movies: towards story-like visual explanations by watching movies and reading books. arXiv preprint arXiv:1506.06724 (2015)