Bio-Medical Question Classification

Jorge Alexander

Abstract—

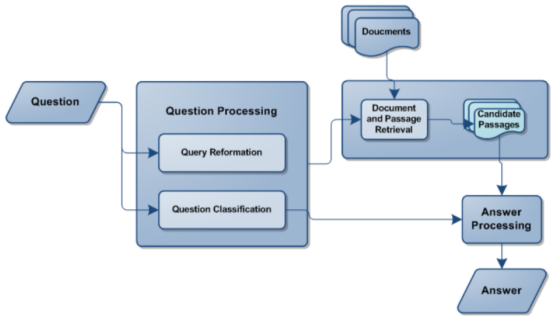
A multiclass classifier for classifying questions into four types is designed and applied over two parts: The first part consists of applying it to a set of 2,252 bio-medical questions annotated with their type from the BioAsQ challenge task 6b, and the second is to apply it to a dataset of Quora duplicate question pairs, in order to improve the overall classification accuracy of the first part. It is found that randomly initialized embeddings perform better than weighted embeddings on the BioASQ dataset. It is also found that increasing the train data set through model predictions of the Quora duplicate questions improves the performance of the original model, although only for a limited increment in the extended dataset size.

Keywords—BI-LSTM, Biomedical, Multiclass, BioAsq, Classification, Question, Word2Vec, Factoid, Summary.

1 Introduction

An important part of natural language is the understanding of questions. An important step in being able to understand a question is categorizing the question by type. In particular, when building a question answering system, a common architecture would contain a question classification module as a key component (See Figure 1). [1]

Figure 1: Example architecture of a question answering system (image source: [2])



Question classes, also referred to as question taxonomy, vary depending on the work. BioASQ is an organization which "organizes challenges on bio-medical semantic indexing and question answering (QA)." and defines 4 question classes for one of its tasks (BioASQ Task 6b): *Factoid*; *List*; *Summary*; *Yes/no*.

Yes/no questions: These are questions that, strictly speaking, require "yes" or "no" answers, though of course in practice longer answers will often be desirable. For example, "Do CpG islands colocalize with transcription start sites?" is a *Yes/no* question.

Factoid questions: These are questions that, strictly speaking, require a particular entity name (e.g., of a disease, drug, or gene), a number, or a similar short expression as an answer, though again a longer answer may be desirable in practice. For example, "Which virus is best known as the cause of infectious mononucleosis?" is a *Factoid* question.

List questions: These are questions that, strictly speaking, require a list of entity names (e.g., a list of gene names), numbers, or similar short expressions as an answer; again, in practice additional information may be desirable. For example, "Which are the Raf kinase inhibitors?" is a *List* question.

Summary questions: These are questions that do not belong in any of the previous categories and can only be answered by producing a short text summarizing the most prominent relevant information. For example, "What is the treatment of infectious mononucleosis?" is a *Summary* question.

In order to develop a classification model that has reasonable accuracy, this report presents an

analysis of the dataset provided by BioAsq task 6B, followed by the design of a multiclass classifier. The results of said classifier on the BioAsq task 6B dataset are explored, and later the classifier is retrained on a dataset of Quora duplicate questions, in order to extend the training dataset in the hope of improving the classifier's accuracy.

2 Data analysis of BioAsq task 6B dataset

To begin, we look at the dataset to understand the type and cardinality of data that we are dealing with. The data consisted of **2,251** questions in total: **619** *Factoid*; **616** *Yes/no*; **531** *Summary*; **485** *List* (See Table 1).

Table 1: BioAsq task 6B Questions

Type	Count	Avg Length (Chars)
Factoid	619	62
Yes/no	616	62
Summary	531	56
List	485	66

Term frequency–inverse document frequency (TF-IDF) weights were constructed from the BioAsq question texts to identify the importance of words in the questions in relation to whole dataset. The TF-IDF weights were separated according to question class and it was found that stop-words were important in differentiating between the classes. For example, it is observed that "what" and "which" are two common terms in the *Factoid*, *List* and *Summary* questions, but not present in the common terms of *Yes/no* question types. It was also observed that words of the biomedical subject, such as "protein" were common in all the question types and would not help us differentiate so much. In order to make these deductions, word-cloud visualizations were created from the TF-IDF weights (See Figure 2, 3, 4 and 5)

Figure 2: Factoid TF-IDF word-cloud

Figure 3: Yes/No TF-IDF word-cloud

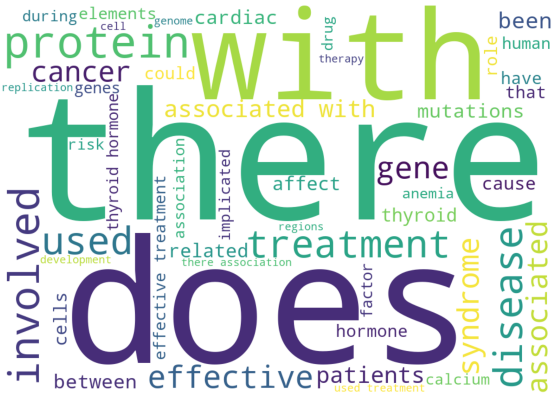


Figure 4: Summary TF-IDF word-cloud

Figure 5: List TF-IDF word-cloud

3 Principle Component Analysis (PCA) of Question Vectors

Having established that some of the words could help us differentiate between question types, it was proposed to train a Word2Vec model from the questions, and perform a PCA analysis to understand if there were any separable features.

3.1 Methodology

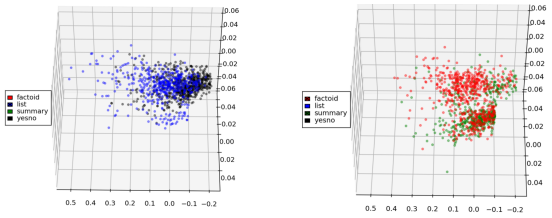
In order to build the Word2Vec model, we used the python gensim Word2Vec library. The model was trained on all of the questions present in the dataset, using the default Word2Vec parameters except a minimum word count of 0. It was found that Bag Of Words produced more separable features than the Skip Gram model.

Having built the vector space from the questions, for each question a vector was built by summing the individual word vectors of the words it contained. Then, a PCA transformation on the question vectors, keeping the top 3 components, and plotting them on axes colored by their corresponding question type.

3.2 Results

Two regions were observed to emerge, which could indicate a solution plane to classify the questions. Whereas Factoid and Summary questions were spread over both regions, List and Yes/No questions were more constrained to one region, Yes/No in particular (See Figure 6).

Figure 6: PCA Analysis for List,Yes/No (Left) and Factoid,Summary (Right)



This suggested that the word vectors technique may provide enough information for a preliminary classification of the question types, from the word vectors. To test this, a support vector machine was used to classify the PCA transformation (using 4 components) of the question vectors. A grid search over kernel, gamma and C was used in optimization. The results can be seen in Figure 7.

Figure 7: Support Vector Machine Classification of PCA of Question Word Vectors

```
SVC(C=100.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=51.79474679231223,
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
              precision    recall  f1-score   support

     factoid       0.67      0.42      0.52        66
        list       0.52      0.43      0.47        35
     summary       0.66      0.70      0.68        63
       yesno       0.67      0.95      0.79        62

   micro avg       0.65      0.65      0.65       226
   macro avg       0.63      0.63      0.61       226
weighted avg       0.64      0.65      0.63       226
```

It was found as expected that some degree of classification accuracy was possible, in particular classifying the Yes/No questions, although the classifier performed badly at the Factoid and List questions, which were spread over both regions in the PCA scatter plot.

4 Word Embedding BI-LSTM BioASQ Classifier

Having had partial success in classifying the word vectors using Word2Vec, PCA and an SVM classifier, a different approach was used. Related work shows success in classifying sequences and questions using LSTM networks with a word embedding.

[3], [4], [5]

4.1 Methodology

The general steps the researched papers follow are:

1) Pre-process the text into a list of tokens, keeping the same label as the piece of text.
2) Create a vocabulary of all the possible tokens the model will be able to handle.
3) Train an appropriate word embedding model from the vocabulary (e.g. Word2Vec, Glove, FastText).
4) Encode the list of tokens from part 1) as a list of integers, each integer representing the index of the token in the vocabulary from part 2).
5) Use the embedding model as the input layer to a neural network.
6) Use subsequent LSTM / BI-LSTM network layers, sometimes preceded by a convolution layer, in order to find relations in the output of the embedding layer, with an appropriate output layer for classification.

4.1.1 Question Pre-processing

The following text pre-processing steps were found to help the classification accuracy:

1) Lower-case the text.
2) Replace question marks with a question token.
3) Replace digits with a digit token.
4) Replace **what's** with **what is**.
5) Replace **'s** with **is**.
6) Replace **'ve** with **have**.
7) Replace **n't** with **not**.
8) Replace **i'm** with **i am**.
9) Replace **'re** with **are**.
10) Replace **'d** with **would**.
11) Replace **'ll** with **will**.
12) Remove all non alpha-numeric characters.

It's interesting to note that stop-words were not removed. As this was a question classification task, key question words (identified previously by the TF-IDF analysis) would have been removed. This supports the research from rule-based question classification systems, that suggests the question word is crucial for question classification. Intuitively as well, pieces of information such as "What is" and "What are", which can differentiate between a "List" and *Factoid* label, would be lost if stop-words were removed. [1], [6]

4.1.2 Vocabulary

Initially, a vocabulary was created from the BioAsq question tokens. However, later on, when extending the dataset, it was found that any new tokens were incompatible with the classification model built from the initial vocabulary. Therefore, the vocabulary was recreated with all the tokens from all considered data-sets, the sizes are shown in Table 2.

Before the tokens were input into the neural network, they were first padded to equal length arrays. The vocabulary had the index of 0 reserved for a padding token, which preceded all token lists shorter than the maximum token list available.

Table 2: Vocab Size

Data Set	Total tokens	Longest Sequence (Tokens)
BioASQ	4135	31
BioASQ with Quora	30529	84

4.1.3 Embedding Model

The Word2Vec model [7] was used to build the word embedding with the following parameters.

- vector size 100
- minimum occurrence of a word of 5
- bag of words model
- negative sampling parameter of 5
- window size 10,

4.1.4 Network Design

The network consisted of the following layers in order: embedding layer, dropout layer, convolutional layer, bidirectional LSTM, batch normalization and dense output layer (See Figure 8).

The embedding layer, originally had an input shape of (31, 100) when applied to the BioAsq dataset only, however it became (84, 100) because largest sequence length identified had 84 tokens when using the Quora dataset. Then the Word2Vec vector size chosen was 100 which is why it had a shape parameter of 100 too.

The dropout layer was added after the embedding to reduce over fitting, a problem that LSTM networks often fall prone to, particularly with small data sets such as the BioASQ. [8]

After the dropout layer, a 32-filter convolutional layer was added, followed by a max-pooling layer of size 2 to reduce the dimensionality and over-fitting too. The convolution layer was added as research has shown that convolutional networks are good at sentence classification. [9]

Following the dropout layer, a bi-directional LSTM layer was chosen. LSTM because of its success in classifying sequences. Bi-directional because it was deduced that question classification was not order dependent (it's not a case of predicting the next word in the sentence).

Following the LSTM layer a batch normalization layer was used, although a dropout layer was sometimes used instead, not both together though as this causes disharmony. [10]

Finally a dense output layer with sigmoid activation was used (size 4 to predict each classification label).

The model was compiled and optimized using the Adam method, and a grid search over the dropout, learning rate, and batch size was conducted to optimize those hyper-parameters.

The epochs that the model was trained on was controlled by using an Early Stopping callback with patience of 5, meaning that if 5 epochs passed without improvement of the loss on the validation set, the training was ended.

Figure 8: Network Model Schema

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 84, 100)	3069900
spatial_dropout1d_1 (Spatial	(None, 84, 100)	0
conv1d_1 (Conv1D)	(None, 84, 32)	9632
max_pooling1d_1 (MaxPooling1	(None, 42, 32)	0
bidirectional_1 (Bidirection	(None, 200)	106400
batch_normalization_1 (Batch	(None, 200)	800
dense_1 (Dense)	(None, 4)	804

Total params: 3,187,536
Trainable params: 117,236
Non-trainable params: 3,070,300

4.2 Results

To evaluate the model, a 10-fold cross-validation was used. The average of the validation are presented as classification reports and can be found in appendix A along with a summary of the trained model and a JSON formatted summary of the configuration used for the run.

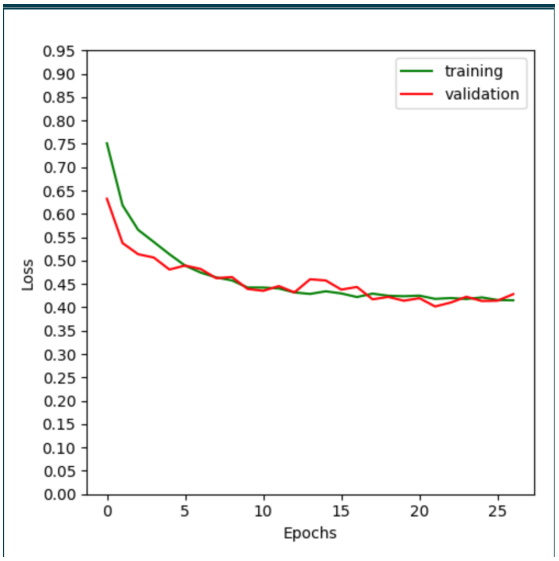
Figure 9 shows the classification report for the network applied to the BioASQ dataset, using a Word2Vec trained embedding (the full run configuration is under the label "2019-01-08 21:20:18.330121" in appendix A).

Figure 9: Weighted Word2Vec BioASQ dataset only classificationreport

Classification Report:

	precision	recall	f1-score	support
factoid	0.38	0.65	0.48	60
list	0.79	0.40	0.53	48
summary	0.69	0.18	0.29	50
yesno	0.66	0.84	0.74	68
micro avg	0.55	0.55	0.55	226
macro avg	0.63	0.52	0.51	226
weighted avg	0.62	0.55	0.53	226

Figure 10: Weighted Word2Vec BioASQ dataset only train/validation loss



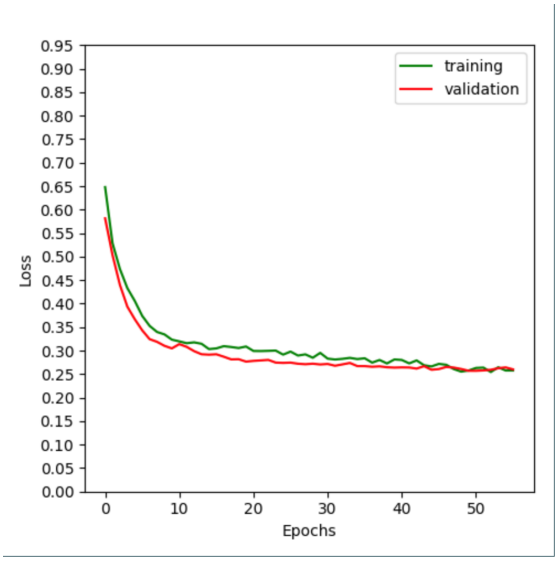
It was not found to show improvement on the PCA approach. However, when the weights in the embedding layer were initialized randomly, better results were achieved (See Figure 11, 12).

Figure 11: Random embedding weights BioASQ dataset only classification report

Classification Report:

	precision	recall	f1-score	support
factoid	0.74	0.56	0.64	77
list	0.96	0.60	0.74	40
summary	0.58	0.92	0.71	49
yesno	0.92	1.00	0.96	59
micro avg	0.76	0.76	0.76	225
macro avg	0.80	0.77	0.76	225
weighted avg	0.79	0.76	0.76	225

Figure 12: Random embedding weights BioASQ dataset only train/validation loss



It was found that a randomly initialized embedding layer outperformed a Word2Vec weighted embedding when training the model on the BioASQ dataset only.

4.3 Results

The poor performance of Word2Vec weighted embeddings compared to randomly initialized weights suggested that the network was improving performance at extracting relations between the question class and question sequences without the need for a Word2Vec embedding.

5 Word Embedding BI-LSTM Extended dataset

In an attempt to improve the classifier, the dataset was extended using a public database of Quora duplicate questions. The model trained on the BioASQ dataset was used to classify the list of Quora duplicate questions. If the classifier classified two questions as the same class, the question text was added to the new dataset, upon which a new model was trained and tested.

5.1 Quora Data Set

The Quora[1] duplicate question dataset is a publicly available set of question pairs, which have been tagged as being paraphrases of each other (referred to as duplicates). In total **149263** duplicate question pairs are present.

A task was proposed to try and take advantage of the possible information contained by the question duplicity. The aim of the task was to investigate whether a multiclass classifier trained on the BioASQ dataset, could then be used to predict the question labels of the Quora duplicates, adding the duplicates which shared the same predicted label to the original dataset, and subsequently train an improved model for question classification on the extended dataset.

5.2 Methodology

In order to carry out the task, the first step was to parse the Quora tokens, using the same text pre-processing as when parsing the BioASQ dataset. From the Quora tokens and the BioASQ tokens, an extended vocabulary was created from which the original model could be trained on. The same network design was used as shown in Figure 8 and the results it was able to achieve on the BioASQ dataset are shown in Figure 13. For the rest of the experiment the Word2Vec weights were used for the embedding layer of the network.

Figure 13: Original Model Results

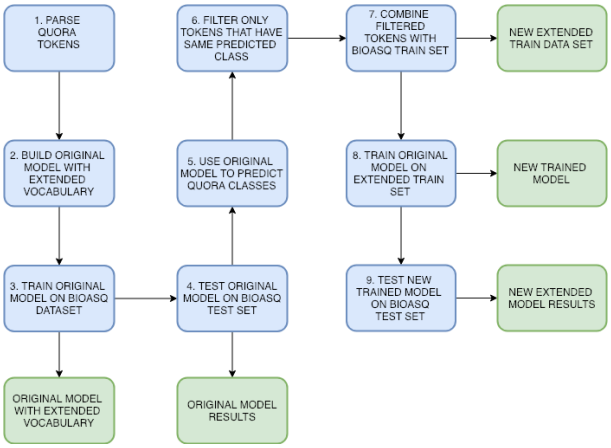
	precision	recall	f1-score	support
factoid	0.76	0.60	0.67	58
list	0.83	0.86	0.85	51
summary	0.76	0.79	0.78	53
yesno	0.89	1.00	0.94	64

The next step was to use the trained original model to predict the Quora classes of the duplicate

1. https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

questions. Once they had been predicted, any pair of questions that did not share the same prediction were discarded. The Quora tokens left were combined with the BioASQ train dataset to create an extended train data set. Then, the original model was retrained on the extended dataset. Finally, the re-trained original model was tested on the same BioASQ test set as originally, and the results compared with the original results. An overview of the methodology is shown in Figure 14.

Figure 14: Steps to construct new extended model



5.3 Results

Table 3 shows the number of predicted labels of each type. This defined the hard limit allowed for increasing the number of training question pairs for each class. The worst performing classes in the original model were *Factoid* and *Summary* classes and during this evaluation only the question count for these were incremented, separately. The hypothesis was that increasing the number of questions for these classes should improve the model's prediction accuracy for them. The training questions for *Factoid* and *Summary* were incremented by **1,000**, **5,000**, **10,000** and **50,000**. [2]

2. Note that as there were only 16,159 predicted duplicate *Factoid* pairs only 32,318 questions were added during the 50,000 increment run. This was an oversight only noticed at the end of the experiment that did not affect the conclusions.

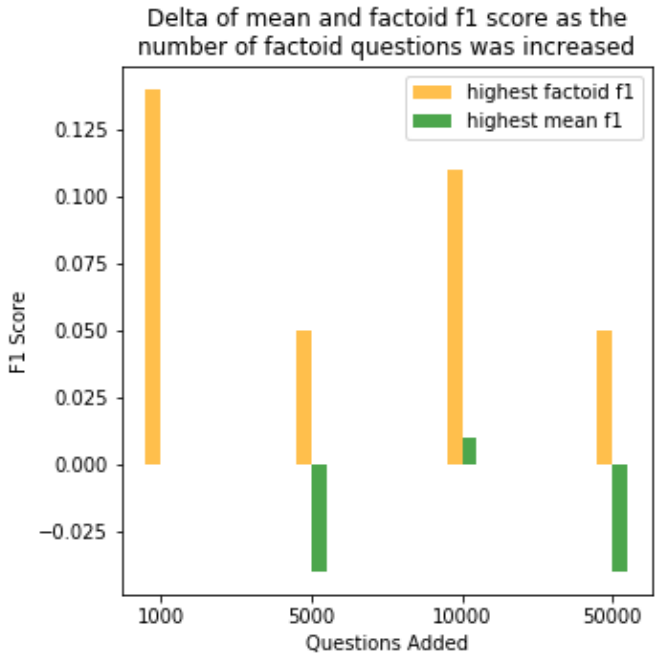
Table 3: Predicted Duplicates

Type	Pair Count
factoid	16159
yes/no	20490
summary	56657
list	1034

To evaluate the new model, at each increment in questions, a grid search over the dropout, learning rate and batch size was performed and the model was used to predict the test set. These results look at the runs with the highest mean F1-scores for each question increment and the runs with the highest *Factoid* or *Summary* F1-score at each *Factoid* or *Summary* question increment respectively.

Figure 15 shows what happened when the *Factoid* training set was incremented. It shows the difference between the mean F1-score and *Factoid* F1-score of the original model on the original dataset and the mean F1-score and highest *Factoid* F1-score for each increment of the extended model.

Figure 15: Factoid increment graph

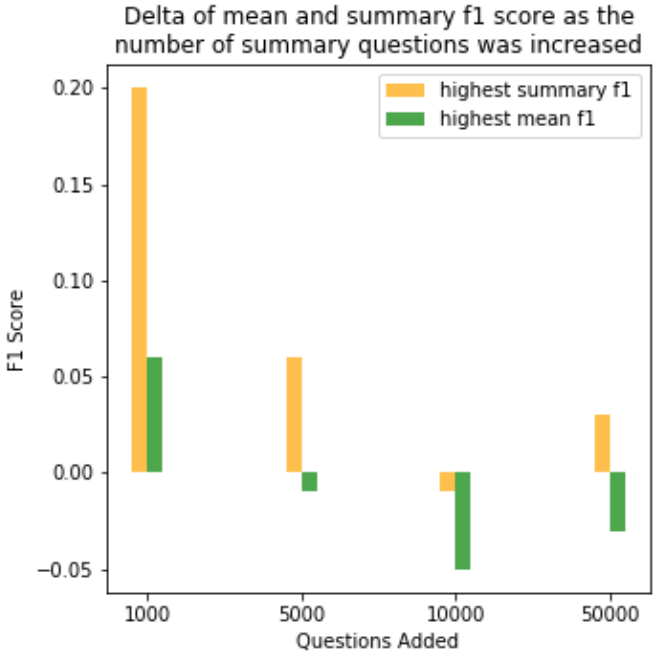


It was observed that although increasing the *Factoid* training data seemed to improve the F1-score for the *Factoid* class, the mean F1-score

did not improve much, in fact it degraded during the 5000 and 50,000 increment. Additionally, the improvement in *Factoid* F1-score seemed to decrease as questions were added, although not enough increments were done for this trend to be conclusive.

Figure 16 shows what happened when the *Summary* training set was incremented in a similar way to how the *Factoid* was. It shows that when 1000 *Summary* predicted questions were added, there was a decent increase in both the mean F1-score and the *Summary* F1-score. However, it was also observed, as in the *Factoid* case, that increasing the number of questions further gave diminishing gains to the *Summary* F1-score and a worsening of the mean F1-score compared to the original model.

Figure 16: Summary increment graph



Overall, the model that performed the best on the test set was during the increment of **1,000** *Summary* questions, with a **0.01** learning rate, a dropout of **0.3** and batch size of **512**. The classification report is shown in Figure 17, the reports for the other runs can be found in appendix B.

Figure 17: Best model classification report

	precision	recall	f1-score	support
factoid	0.95	0.71	0.81	55
list	0.90	0.83	0.86	53
summary	0.77	0.94	0.85	68
yesno	0.94	1.00	0.97	50

It was observed that the best model was able to achieve an F1-score of over 0.8 for all classes, higher than what the original model had been able to achieve both with and without an initialized word embedding layer.

6 Conclusion and future work

Two experiments have been run on an initial, and later extended dataset. The first experiment, a question classification using an LSTM with an embedding layer on the BioASQ dataset, suggested that Word2Vec vector weights did not improve classification. It would be interesting to attempt a rule-based approach which might be able to achieve state-of-the-art results without the need for the intricacies of training and testing a neural network. [1]

The second experiment, exploring the effects of increasing the training set size through the prediction of question taxonomy of duplicate questions, seemed to suggest that increasing the training set size in the explored way can improve a model's performance, but only up to a certain point, beyond which the accuracy of any other labels decreases along with the label thats being predicted. This would make sense in the way that this experiment was performed, as adding only one type of question to the train dataset would bias the dataset and in turn make it difficult for the classifier to perform balanced predictions. It would be interesting to explore how the accuracy is affected if the dataset is increased with equal numbers of each class, or even further to explore what ratio of predicted question labels should be added to the train dataset, to maximise the performance of the classifier.

Furthermore, it would be interesting to identify if the Word2Vec weighting of the embedding layer is having an effect on the performance of the classifier as the training size is increased.

References

[1] H. T. Madabushi and M. Lee, "High accuracy rule-based question classification using question syntax and semantics," in *COLING*, 2016.

[2] B. Loni, "A survey of state-of-the-art methods on question classification," 12 2018.

[3] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, "Text classification improved by integrating bidirectional lstm with two-dimensional max pooling," *CoRR*, vol. abs/1611.06639, 2016. [Online]. Available: http://arxiv.org/abs/1611.06639

[4] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ser. HLT '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 142–150. [Online]. Available: http://dl.acm.org/citation.cfm?id=2002472.2002491

[5] H. Tayyar Madabushi, M. Lee, and J. Barnden, "Integrating question classification and deep learning for improved answer selection," in *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, 2018, pp. 3283–3294. [Online]. Available: http://aclweb.org/anthology/C18-1278

[6] C. Saedi, J. Rodrigues, J. Silva, A. Branco, and V. Maraev, "Learning profiles in duplicate question detection," in *2017 IEEE International Conference on Information Reuse and Integration (IRI)*, Aug 2017, pp. 544–550.

[7] Y. Goldberg and O. Levy, "word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method," *CoRR*, vol. abs/1402.3722, 2014. [Online]. Available: http://arxiv.org/abs/1402.3722

[8] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *CoRR*, vol. abs/1409.2329, 2014. [Online]. Available: http://arxiv.org/abs/1409.2329

[9] Y. Kim, "Convolutional neural networks for sentence classification," *CoRR*, vol. abs/1408.5882, 2014. [Online]. Available: http://arxiv.org/abs/1408.5882

[10] X. Li, S. Chen, X. Hu, and J. Yang, "Understanding the disharmony between dropout and batch normalization by variance shift," *CoRR*, vol. abs/1801.05134, 2018. [Online]. Available: http://arxiv.org/abs/1801.05134