A Study on Question Classification in Natural Language Processing

Aparna(11106479) and Balendra Singh(11166812) and Mariam Jamilah(11106105)
and Surya Ramessh(11041102) and Thushara Nair(11078927)
University of Manchester

Abstract

Question classification is a key aspect of modern question-answering systems, as it allows for the effective retrieval of relevant answers from knowledge bases, search engines, or data sources. In this study, the performance of two question classification techniques, namely bag-of-words (BoW) and bidirectional long short-term memory (BiLSTM) models, were compared using PyTorch. The study involved testing both models on a dataset of labelled questions with different word embeddings and parameters. Results showed that the BiLSTM model outperformed the BoW model. Additionally, the use of pre-trained GloVe word embeddings was found to be more effective than randomly initialized word embeddings. Furthermore, the study found that fine-tuning word embeddings during training can lead to improved model performance.

1 Introduction

Question classification is an important task in natural language processing and information retrieval, as it involves categorizing a given question into a specific class based on its semantic content. Accurately classifying questions is crucial for developing conversational agents, chatbots, and other intelligent systems that rely on natural language interactions to provide high-quality responses to users.

However, this task is challenging due to the variability and ambiguity in natural language, where the same question can have multiple interpretations depending on the context or domain. Recent advancements in deep learning-based approaches, such as word embeddings and recurrent neural networks, have shown promising results for question classification tasks.

This paper compares the performance of bag-of-words (BoW), bidirectional long short-term memory (BiLSTM) models, and ensemble model for question classification using PyTorch. The study examines various word embeddings and techniques, such as freezing and fine-tuning pre-trained embeddings.

2 Related Work

Question Classification (QC) aims at assigning the question to one of k classes that impose a semantic constraint on the desired answer(1). A hybrid framework using CNN for query title processing and BiLSTM for query description handled question classification task demonstrated their effectiveness in NLP tasks such as sentence classification and text classification (2). Another research experimented with five machine learning algorithms using two kinds of features and found that SVM performed the best. It proposed a special kernel function called the tree kernel to take advantage of the syntactic structures of questions(3). An article proposed a compact feature set for question classification in automated question answering, including a head word feature and augmented semantic features using WordNet and Lesk's word sense disambiguation algorithm (4).

Research Question

1. *How does the performance of question classification models change depending on the different types word embeddings that are used?*

2. *How can question classification models in natural language processing (NLP) be improved in terms of accuracy?*

3 Methodology

3.1 Experimental setup and Design

To conduct an effective investigation into the research questions related to question classification, a comprehensive experimental setup was devised in four distinct phases. Firstly, the data was loaded and split into a training set and a development set, and a separate test set was also loaded. Subsequently, a vocabulary was developed based on the available data (questions), and data pre-processing was performed. In the third phase, the classifier was trained using the training set with an appropriate learning rate of 0.001, batch size of 8, and a hidden layer of dimension 128 and 10 epochs. Finally, the classifiers were evaluated against the development and test sets using the performance metrics, accuracy, and F1 score(macro).

Accuracy (5) represents the percentage of correctly classified instances out of the total number of instances.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

In contrast, the F1 score (5) is a more informative metric that balances precision and recall.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \quad (2)$$

Precision measures the proportion of true positives out of the instances predicted positive, and recall measures the proportion of true positives out of the actual positives.

3.2 Dataset Description and Pre-Processing

In this research study, the task of question classification is performed on the Text REtrieval Conference (TREC) dataset. This dataset contains fact-based questions in the English language and is categorized into six broad semantic categories and fifty fine-grained categories. The dataset consists of 5,452 samples for training and 500 samples for testing, with a train-to-validation ratio of 9:1.

To pre-process the data, the text was first converted to lowercase, punctuation and stop words were removed, and then a vocabulary was created to generate word embeddings. This involved mapping each word in the vocabulary to a fixed-length vector representation. These embeddings were then used to train the models for the task of question classification.

3.3 Word Embeddings

The creation of word embeddings is a crucial step in the Question classification task. Word embeddings are generated using either random or pre-trained methods, with a fixed embedding dimension of 300.

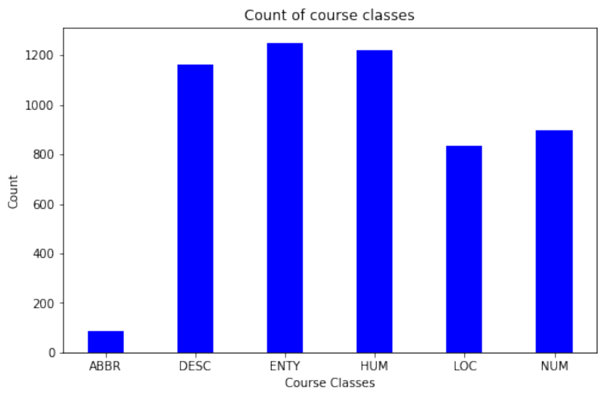

Figure 1: Coarse-Grained Categories

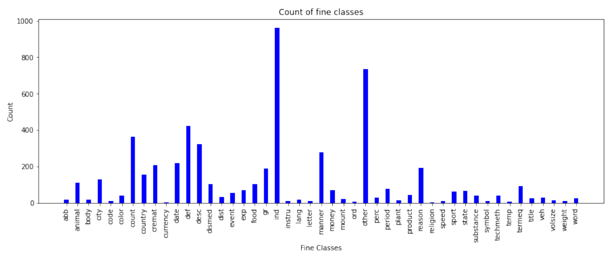

Figure 2: Fine-Grained Categories

To generate randomly initialised word embeddings, the sentence is tokenized, and a word dictionary is constructed to map each word in a sentence to its corresponding index. The resulting list is padded to generate a fixed-length array that represents the sentence as a sequence of indices. This is then transformed into a PyTorch tensor, to feed into a machine-learning model, along with the randomly intialised tensors to generate a vector representation of the sentence for classification tasks.

For the pre-trained embedding, the GloVe corpus is used to generate a dictionary of words and their corresponding embedding vectors. These word embeddings are then subjected to BoW or BiLSTM.

3.4 Bag of Words

Bag-of-Words (BoW) is a highly prevalent technique in natural language processing for text classification tasks. In this approach, a document or sentence is represented as a bag of individual words, without considering their order or context (6).

The BoW Classifier performs a forward pass on sentence sequence of indices using the embeddings. Then a mean operation is performed over the embeddings of all the words in the sentence to obtain

Model & Embedding	Freeze Embedding?	Class	Validation Accuracy	Validation F1-Macro	Validation Average Loss	Test Accuracy	Test F1-Macro	Test Average Loss
BiLSTM Random	No	Coarse	83%	0.76	0.86	89%	0.88	0.52
BiLSTM Random	No	Fine	76%	0.62	1.36	78%	0.58	1.17
BiLSTM Pre-trained	No	Coarse	82%	0.75	0.98	85%	0.82	0.71
BiLSTM Pre-trained	Yes	Coarse	80%	0.73	1.02	86%	0.86	0.61
BiLSTM Pre-trained	No	Fine	75%	0.62	1.21	77%	0.58	1.11
BiLSTM Pre-trained	Yes	Fine	70%	0.55	1.4	75%	0.46	1.21
BoW Random	No	Coarse	79%	0.76	0.58	82%	0.82	0.58
BoW Random	No	Fine	73%	0.55	1.25	77%	0.55	1.03
BoW Pre-trained	No	Coarse	83%	0.81	0.73	88%	0.87	0.46
BoW Pre-trained	Yes	Coarse	79%	0.77	0.59	81%	0.82	0.55
BoW Pre-trained	No	Fine	79%	0.65	1.28	78%	0.59	1.02
BoW Pre-trained	Yes	Fine	72%	0.5	1.12	72%	0.44	1.04
Ensemble Pre-trained	No	Coarse	83%	0.81	0.87	84%	0.83	0.79
Ensemble Pre-trained	No	Fine	78%	0.66	1.04	76%	0.58	1.14

Table 1: Accuracy, F1 Score and Average Loss on Validation and Test Dataset

a fixed-length sentence representation. This is then passed through two fully connected layers with ReLU activation to generate the final output which is subjected to cross-entropy loss (which uses a softmax layer).

3.5 BiLSTM

The BiLSTM model is a recurrent neural network architecture that has demonstrated promising results in several natural language processing tasks, such as sentiment analysis, named entity recognition, and machine translation. It is a variant of the standard LSTM model, designed to handle input sequences in both forward and backward directions (7).

The BiLSTM model takes as input word embeddings into an embedding layer. These embeddings can either be kept fixed or updated during training. The embeddings are then fed into a bidirectional LSTM layer, which processes the input sequence in both forward and backward directions. This enables the model to capture both past and future context information from the text. The outputs of the forward and backward LSTMs are combined by concatenation to produce a fixed-length vector representation of the input text. The concatenated output is then passed through a fully connected linear layer with a ReLU activation function.

3.6 Ensemble

In addition to these models, we have implemented an ensemble model which comprises of BiLSTM layer followed by BoW layer. The BiLSTM layer helps retain information on the context and sequence of the sentence which is then passed to the BoW layer.

4 Results and Analysis

The study found that the decision to freeze pre-trained layers during training depends on the specific problem and the model being used. If the pre-trained layers are suitable and require minimal modification, freezing them and training only the new layers can result in faster training and improved performance. However, if the pre-trained layers are not a good match or require significant modification, fine-tuning the pre-trained layers along with the new layers may be the better option. The experiment demonstrated that fine-tuning the pre-trained layers along with the new layers resulted in better performance for both fine and coarse classifications, achieving higher F1 scores compared to freezing the pre-trained layers as seen in Table 1.

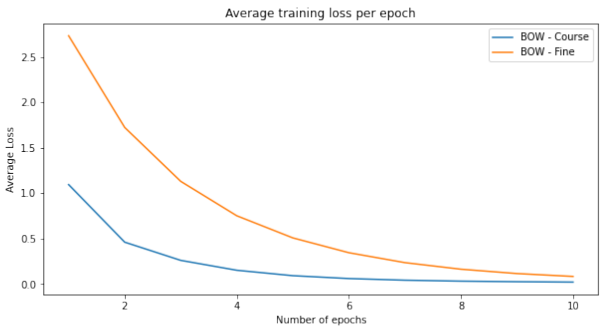

Figure 3: Average Training loss per epoch for BoW

BiLSTM is considered superior to BoW for natural language processing tasks because it can comprehend the context and sequence of words in a

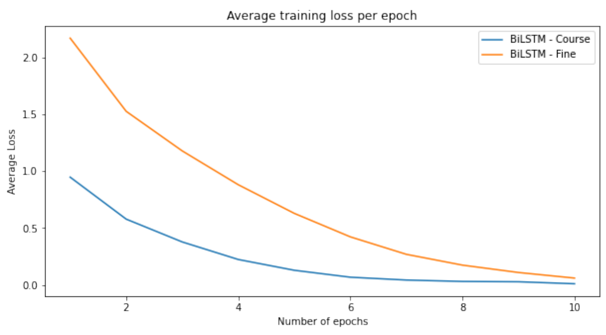

Figure 4: Average Training loss per epoch for BiLSTM

sentence, whereas BoW treats each word in a sentence as a separate entity. Based on Figures 3 and 4, it can be inferred that BiLSTM has a lower loss compared to BoW, indicating that BiLSTM outperforms BoW.

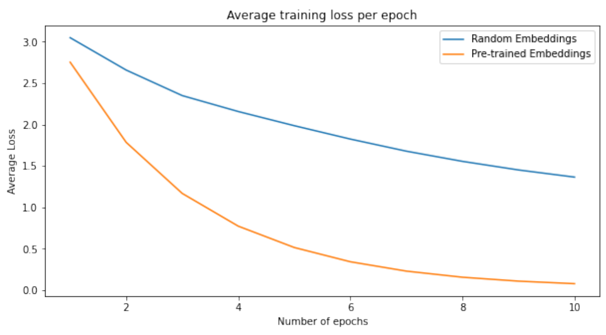

Figure 5: Average Training loss per epoch for Random and Pre-trained for BoW

The experiment results showed that pre-trained GloVe embeddings performed better compared to randomly initialized word embeddings. One possible reason for this is that the GloVe embeddings, which are pre-trained, have more context knowledge than the word embeddings that are randomly initialized. This can be observed from Figure 5 which illustrates the average loss of the BoW model for random and pre-trained embedding.

Furthermore, the study found that using only a portion of the training data in question classification resulted in lower accuracy since the model did not have access to enough examples to learn the important patterns and features for accurate classification. Therefore, it is recommended to use as much of the training data as possible to improve the model's ability to learn an accurate classification function.

One can infer by looking at Figure 1 and Figure 2 that the coarse categories have a larger number of data available for training compared to the fine-grained categories. Consequently, we can observe in Table 1 and Figures 3 and 4, that the accuracy for coarse categories is superior to that of fine-grained categories and it is more complex for the model to differentiate between classes with greater precision for fine-grained classification when compared to coarse-grained classification. The 'abbr' class in coarse-grained classification presents a challenge since there is inadequate training data for this class. On the other hand, in fine-grained classification, the class 'religion' and 'currency' is found to be difficult. Furthermore, the study revealed a minor decline in accuracy when removing stop words from the input data. However, the punctuation is removed as there is an improvement in the accuracy.

5 Conclusion

In summary, this study investigated the impact of various factors on the performance of natural language processing models, specifically focusing on pre-trained models, word embeddings, training data size, and classification granularity. The findings showed that the decision to freeze or fine-tune pre-trained layers relies on the specific problem and pre-trained model being used. Moreover, pre-trained embeddings could outperform randomly initialized embeddings. The study emphasized the significance of using an adequate amount of training data to obtain accurate classification results, and highlighted the complexity of fine-grained classification.

5.1 Limitations and Threats to Validity

The research has a limitation due to the variable sentence length in the training dataset, with an average of 10 words per question, where padding bits are used to fill the gap between the maximum words and actual sentence length, which can adversely affect the accuracy. Additionally, the study relied on basic Python libraries for tokenisation, which may not be as effective as advanced libraries such as sklearn. Furthermore, the study used only a subset of the GloVe word embeddings, which may have restricted its overall performance. The study also found that the accuracy of the fine classes decreased due to the limited amount of data available, and increasing the dataset size could improve accuracy. One potential threat to the validity of the study is its evaluation only on a specific dataset, which may not be generalisable to other corpora.

5.2 Future Work

Moving forward, there are several avenues for improving the accuracy of the model in question classification. One potential approach is to incorporate techniques such as lemmatization, POS tagging, and named entity recognition to enhance the processing of natural language data.

Moreover, our study used PyTorch and standard Python libraries for model training and evaluation, which may have limited the model's accuracy potential. Future research could explore the use of alternative machine learning frameworks or libraries that may offer better performance for natural language processing tasks. Additionally, there may be opportunities to optimize the model's hyperparameters and architecture to further improve its accuracy.

Finally, it may be worthwhile to investigate the use of alternative pre-trained models or embeddings to see if they can provide a performance boost over the ones used in our study.

References

[1] X. Li and D. Roth, "Learning question classifiers," in *COLING 2002: The 19th International Conference on Computational Linguistics*.

[2] Z. Zhou, X. Zhu, Z. He, and Y. Qu, "Question classification based on hybrid neural networks," in *2016 4th International Conference on Electrical & Electronics Engineering and Computer Science (ICEEECS 2016)*. Atlantis Press, 2016, pp. 44–52.

[3] D. Zhang and W. S. Lee, "Question classification using support vector machines," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 26–32.

[4] Z. Huang, M. Thint, and Z. Qin, "Question classification using head words and their hypernyms," in *Proceedings of the 2008 Conference on empirical methods in natural language processing*, pp. 927–936.

[5] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: an overview," *arXiv preprint arXiv:2008.05756*.

[6] Y. HaCohen-Kerner, D. Miller, and Y. Yigal, "The influence of preprocessing on text classification using a bag-of-words representation," *PloS one*, vol. 15, no. 5, p. e0232525, 2020.

[7] G. Liu and J. Guo, "Bidirectional lstm with attention mechanism and convolutional layer for text classification," *Neurocomputing*, vol. 337, pp. 325–338, 2019.