Text Mining : Question Classification

Abstract

Question classification is a vital aspect of natural language processing (NLP) essential for developing automated question-answering systems. This study illustrated categorising questions into predefined types using two classifiers (Bag-of-words; Bilstm) with an NPL pipeline containing text parsing, pre-processing, word embedding and sentence representation; furthermore, we invested the impact of different approaches with word embedding through an experiment based on development data and macro F1 score. Experiment results were obtained to address the proposed hypnosis. We scored 68% for the fine and 80% for the coarse classes after 10 epochs.

Keywords: Word Embedding, Neural Network, BiLSTM, Natural Language Processing

1. Introduction

In natural language processing (NLP), question classification is a critical task (Moldovan et al. 2003) that involves categorising questions into predefined categories based on their content and structure, in which the machine can comprehend the meaning of questions and provide meaningful responses (Yilmaz and Toklu 2020). Despite the progress made in NLP techniques, developing accurate question classifiers remains a complex problem. One of the main reasons is the ambiguity of human language, which can lead to various interpretations of a single sentence or question. For instance, the question "What is the capital of China?" can be classified as a location question, but it may also be a question about politics or history. Additionally, the diversity of question types poses a further challenge, as they may range from straightforward factual questions to complex and unstructured ones (Mishra and Jain 2016). However, word embedding and sentence representation from the question classification system can address these by enhancing the machine to understand the meaning and context of words and sentences in machine language.

Word embedding is crucial because it represents words as dense vectors in a continuous vector space, capturing their meaning and semantic relationships. It allows machine learning models to better understand the context of words in a sentence, improving the accuracy of question classification (Ghannay et al. 2016). Sentence representation captures a sentence's overall meaning and context as a single vector. It allows machine learning models to better understand the semantic relationships between words in a sentence, which is crucial for accurately predicting classification results from a given question (Li and Roth 2002). However, these can be difficult due to the technical requirements, computational demands, and inherent complexity of natural language processing.

In this report, we proposed two question classifiers using different NLP techniques: (i) Bag-of-words, (ii) Bidirectional Long Short-Term Memory (BiLSTM) based on two different word embedding approaches: (i) Randomly initialised word embeddings, (ii) Pre-trained word embeddings with GloVe along with the different sentence representations. We aim to compare two models with various settings and develop a highly accurate question classification system.

2. Methodology

The methodology in this study is shown in Fig 1, comprising four parts: 1) Pre-processing, 2) Word Embedding, 3) Sentence representation, 4) Classifier

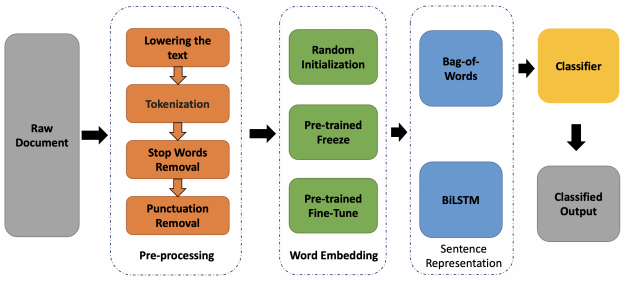

Figure 1. NLP Pipeline for Question Classification

2.1 Preprocessing

Preprocessing involves two sections: creating a vocabulary of the training set and indexing all the words in sentences within the training set, as well as converting the GloVe vectors into a format that can be loaded by PyTorch. To accomplish this, We firstly converted all words to lowercase, removed stopwords and pure punctuation tokens. However we deliberately kept some question words provided in the original stopword list such as "when" or "where." Because these question words could have an intuitive connection to the question class (Dong et al. 2015). Then, we constructed the vocabulary and stored it as a text file on disk to ensure uniform word indexing across all data set involved. We set a fixed sentence length of 20, truncated any longer sentences, padded shorter sentences with 0 and mapped the words to their indices in the vocabulary list.

As for the pre-trained weight, we chose GloVe for its clear formatting (Pennington, Socher, and Manning 2014). Each line in the GloVe file represents a word and its corresponding embedding weight (Pennington, Socher, and Manning 2014). The first token is the lowercased word, while the remaining tokens consist of float numbers represent the embedded weight vector (Mikolov et al. 2013). However, the GloVe contains over 400,000 vocabulary items, whereas our training set has

less than 9,000. The incompatible vocabulary sizes of our training set and GloVe significantly hampered the program performance. To address this, we pruned the GloVe word vector set by only keeping words that appeared in the training set (Dharma et al. 2022). We then wrote the pruned vocabulary list and weights into files to avoid the need to load the original GloVe word vector set file.

2.2 Model Structure

The model consisted of three main components: word embedding, sentence representation, and classifier. Fig.1provides an overview of the structure for six model settings. To explore the performances with various approaches, we experimented with both a pre-trained approach (with options of freezing and fine-tuning) and random initialisation for word embedding, and two different representation models which were Bag-of-Words (BoW) and Bidirectional Long Short-Term Memory (BiLSTM).

Table 1. Detailed Model Structure with Output Resolution

Component	BoW	BiLSTM
Input[a]	(500, 20)	(500, 20)
Word Embedding	(500, 20, 300)	(500, 20, 300)
Sentence Representation[b]	pretrain:(500, 300) Non-pretrain:(500,1100)	(500, 20, 512)
Normalization	pretrian:(500, 300) Non-pretrain:(500, 1100)	— —
Dropout	—	(500, 512)
Fully Connected Layer[c]	(500, 6 or 50)	(500, 6 or 50)

a The input is the output of TextParser.
b The value 512 is twice the size of the hidden dimension. To generate the input for the classifier, we retrieved the first and last sequence of the 3D output and concatenated them into a 2D matrix with dimensions of (500, 512).
c The classification task with options of COARSE or fine classes determines the final output shape. This is the output layer. We did not apply activation layer with Softmax because we used Cross-entropy loss.

1. `Word Embedding`
   We used pre-trained GloVe vectors with the option for freezing or fine-tuning vectors as embedded weights. We also experimented with randomly initialised vectors.
2. `Sentence Representation`
   BoW: Bag of words is a representation of text that describes the occurrence of words within a document. We used it to keep track of word counts and disregard the grammatical details and the word order (Salton and Buckley 1988).
   BiLSTM: BiLSTM is a recurrent neural network to process sequential data. We used BiLSTM to process word vectors in both directions simultaneously to capture information (Siami-Namini, Tavakoli, and Namin 2019).
3. `Classifier`
   Instead of feeding directly to a fully connected layer, we passed the data to a random-dropout layer after sentence representation to prevent overfitting. Finally, the classifier classified the data by a fully connected layer and produced a probability distribution matrix.

2.3 Experimental Settings

Hyperparameters:
batch_size: the size of batches, set to 500
epoch: the number of epochs, set to 10
embedding_dim: embedding dimension, set to 300
hidden_dim_bilstm: hidden dimension of BiLSTM layer, set to 256
hidden_layer_size: hidden layer size, set to 256 (only used in Bow)
Loss Function: Cross-entropy loss
Optimizer: Adam optimizer with learning rate 7e-3, L2 regularizer with weight decay 1e-5, and an exponential scheduler with gamma set to 0.9
Dropout: randomly dropout with probability of 0.2 (Bow) and 0.3(BiLSTM)
Performance Metrics: This project used accuracy and macro f1 score metrics to evaluate the model's performance.
In the experiments, we also tried various loss functions like negative log-likelihood, batch normalization, and wider fully connected layers with various activation functions. We used the above settings due to their better performance.

3. Experiments

To ensure the credibility of our experimental results, we conducted each experiment three times, rounded the averages to 2 decimal places, and recorded the final results in Table 2. Our work involved using data from the training set 5 of Xin Li and Dan Roth's work (Li and Roth 2002), which consisted of 5500 labelled questions with a two-layered taxonomy that provided a natural semantic classification for typical answers in the TREC task. The taxonomy included six coarse classes (Abbreviation, Entity, Description, Human, Location, and Numeric) and 50 fine classes. Since there was no development set, we manually split the dataset into 10 portions and allocated 9 portions for training and the remaining portion for development. We utilized TREC 10 questions, which contained 500 labelled questions, for testing purposes. Furthermore, based on the experimental results, we conducted in-depth analyses as described below.

1.Effects of word embedding approaches
As is shown in the Table 2, different embedding approaches had slighter effects on coarse-class classification than that on fine-class classification. Freezing and fine-tuning are two approaches to pre-trained word embedding (Mathew and Bindu 2020). While freezing used pre-trained GloVe vectors with no weight tuning, fine-tuning updates the embedding vectors during training (Rezaeinia, Ghodsi, and Rahmani 2017). Fig.2 shows that fine-tuning outperformed freezing. The reason for this was that fine-tuning could adjust weights during training, allowing it to adapt to gradient descent process (Dodge

Table 2. Experiment Result after 10 epochs

Experiment No.	Word Embedding		Sentence Representation	Fine Class		Coarse Class	
				Accuracy	Macro F1	Accuracy	Macro F1
1	Random Initialisation	–	BoW	Dev: 0.70	Dev: **0.60**	Dev: 0.81	Dev: **0.79**
				Test: 0.75	Test: 0.57	Test: 0.84	Test: 0.83
2	Pre-trained	Freeze	BoW	Dev: 0.69	Dev: **0.55**	Dev: 0.75	Dev: **0.73**
				Test: 0.74	Test: 0.57	Test: 0.81	Test: 0.82
3	Pre-trained	Fine-Tuning	BoW	Dev: 0.76	Dev: **0.65**	Dev: 0.80	Dev: **0.78**
				Test: 0.77	Test: 0.62	Test: 0.86	Test: 0.86
4	Random Initialisation	–	BiLSTM	Dev: 0.70	Dev: **0.59**	Dev: 0.78	Dev: **0.78**
				Test: 0.76	Test: 0.60	Test: 0.85	Test: 0.85
5	Pre-trained	Freeze	BiLSTM	Dev: 0.78	Dev: **0.68**	Dev: 0.84	Dev: **0.82**
				Test: 0.80	Test: 0.67	Test: 0.88	Test: 0.88
6	Pre-trained	Fine-Tuning	BiLSTM	Dev: 0.76	Dev: **0.68**	Dev: 0.83	Dev: **0.80**
				Test: 0.80	Test: 0.67	Test: 0.87	Test: 0.86

et al. 2020). On the contrary, freezing embedding only used fixed pre-learned features.

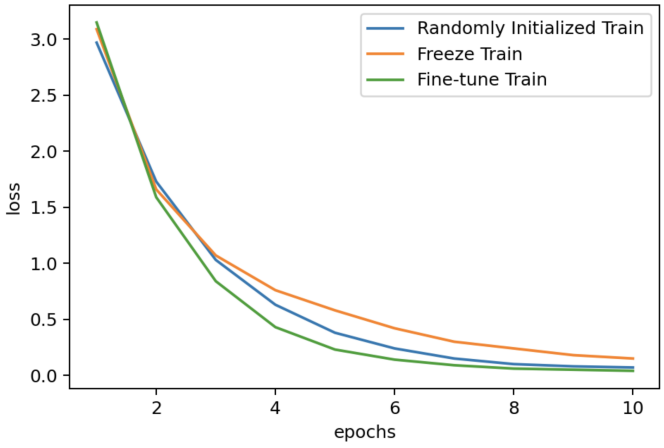

Figure 2. BiLSTM Training Loss of Fine Classes

Unlike pre-training, random initialization (RI) began with random vectors and no prior experiences (McCann et al. 2017). Fig.2 shows that RI performed better than freezing but worse than fine-tuning. Based on prior experiences, the losses of pre-trained model decreased fast while that of RI decreased slowly. However, fine-tuning and RI eventually outperformed freezing. If we increase the size of the dataset or the number of epochs, RI may even outperform.

Furthermore, while freezing performed worse during training, it outperformed RI and fine-tuning during development and testing, particularly with BiLSTM. One possible reason could be that RI and fine-tuning with BiLSTM severely overfitted the dataset, whereas freezing prevented overfitting to some extent.

2.Effects of using only part of the training set

We chose BiLSTM PreTrained FineTunning model to test this experiment on the Fine class. The initial training set consisted of 4906 data items, and we used 75% (3679), 50% (2453), and 25% (1226) of the original data set to do this experiment. The result is shown in Table 2.

Comparing with the results obtained using the entire training set, it is evident that as the amount of training data decreases, both the accuracy and F1 score on development set and test set decrease accordingly. We concluded that using only a portion of the training set will hamper model performance. One possible explanation is that a portion of the dataset is inadequate to provide data that can reflect its patterns and implicit relations and may lead to underfitting. Consequently, the model's performance is compromised.

Table 3. Part of Training Set Training Performance Results

Amount of Training set	Accuracy	F1 Score
75%	dev:0.72	dev:0.64
	test:0.71	test:0.56
50%	dev:0.67	dev:0.50
	test:0.69	test:0.50
25%	dev:0.64	dev:0.47
	test:0.63	test:0.41

3.Which Classes are more difficult to classify?

By analyzing the output of test set, we can find that in the fine class, the BoW model had 0% correct classifications for 'ENTY:event', 'ENTY:product' and 'HUM:title '; the BiLSTM model has 0% correct classification rate for 'ENTY:plant', 'ENTY:currency', 'ENTY: event', 'ENTY:product', 'HUM:title' and 'ENTY:instru '. So we can conclude that 'ENTY:event' and 'ENTY:product' are the two most difficult categories in fine class to classify.

In the coarse class, the Bow model has a higher than 70% probability of correctly classifying each class. For the Bilstm model, the probability of correctly classifying the 'ABBR' class and the 'LOC' class is 0% and 9% respectively. Therefore, for the BoW model, the classification accuracy is high for all categories. For the BiLSTM model, it can be seen that both the 'ABBR' and 'LOC' categories have very low accuracy and are much more difficult to classify.

4.Between coarse and fine classes, which one is more difficult?

Fine class is more difficult to classify than coarse class. This is due to the fact that the fine class has more complex and well-defined categories. This is due to the fact that the fine class has more complex and well-defined categories. Fine class has fifty categories, whereas the coarse class has only six. In Table 2, we can see that regardless of the word embedding and sentence embedding methods, the accuracy and F1 scores of the coarse class are always greater than fine class.

5.Effects of other preprocessing steps

We also conducted experiments to determine whether stopwords and punctuation would affect the results. The stopword list used in these experiments was the original one including all question words. The parameter setting is the same as our previous experiment using BiLSTM, pre-trained weight and fine-tune. We obtained the results shown in Table 2.

The table illustrated that in all experiments, the removal of punctuation improved results. Nevertheless, removing stopwords significantly reduced both the accuracy and F1 score. As hypothesized, certain stopwords such as question words may possess an intuitive connection to question classes. Removing these question words may lead to decreased performance.

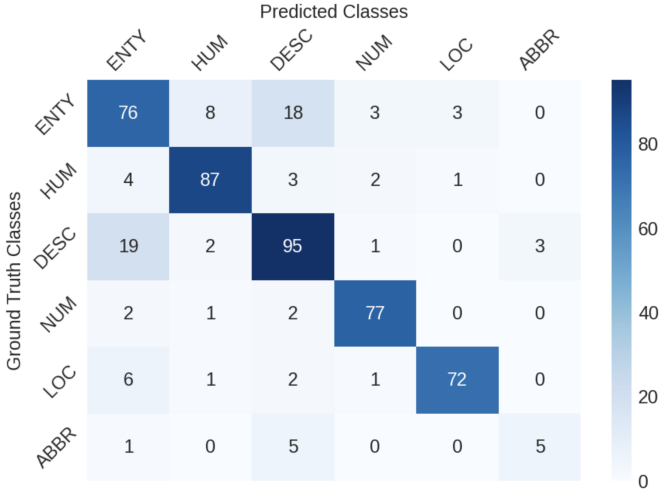

Figure 3. Confusion Matrix of COARSE

6. Confusion Matrix Analysis

A confusion matrix is a summary of prediction results on a classification task. Fig 3 depicts a visual confusion matrix of COARSE classification using fine-tuned pre-train embedding with six classes. Here are some observations:

Correctness: The diagonal of the matrix represents the instances that were correctly classified. Most of the instances were classified correctly considering the total number of instances of each class.

Class Imbalance: The total number of instances of ABBR classes is significantly lower than of the other classes. In this case, ABBR is a minority class, while the other classes, particularly ENTY and DESC, are majority classes. The misclassified example in Fig.3 tended to be classified as majority classes. For example, there are 11 ABBR instances in total, with 5 correctly classified, 1 classified to ENTY, and 5 classified to DESC. The majority classes, on the other hand, also influenced each other. There were 18 ENTY instances classified as DESC and 19 DESC instances classified as ENTY. The class imbalance resulted in poor performance on minority classes and a bias towards majority classes, which caused worse general performance.

7. Evaluation Metrics Analysis

In Table 2, the accuracies and f1 scores for fine-class classification were significantly different but very similar for COARSE classification. The reason for this was that the F1 score was independent of the true-negatives, whereas accuracy was not. Assuming that the output of one instance was a binary vector with length equal to the number of classes, COARSE classification had a few zeros but fine-class classification had a large number of zeros. In this case, the accuracy could be much higher than f1 scores for fine classes, according to the accuracy formula. As a result, the macro f1 score was a better metric for evaluating our task.

Conclusion

In this project, we employed various techniques to parse raw text and chose the optimal combination. Subsequently, we implemented different options of using pre-trained weights, fine-tuning for word embedding, and used bag-of-words and BiLSTM to generate sentence representation. We trained neural networks as our classifiers and obtained classification results. Overall, BiLSTM with pre-trained weight and fine-tuning presented significant outperformance over bag-of-words when classifying fine classes. However, BiLSTM and bag-of-words showed merely subtle differences in performance when classifying coarse classes with various word embedding options. Despite this success, our work is not without limitations. As previously described, our model presented poor results when classifying specific classes. This may be due to an unbalanced training dataset or inadequate number of epochs. To improve our work, potential actions may include data augmentation or finding a balanced dataset for training, increasing the number of epochs, and adjusting the structure of the neural network.

References

Dharma, EDDY MUNTINA, F Lumban Gaol, HLHS Warnars, and BENFANO Soewito. 2022. The accuracy comparison among word2vec, glove, and fasttext towards convolution neural network (cnn) text classification. *Journal of Theoretical and Applied Information Technology* 100 (2): 31.

Dodge, Jesse, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305.*

Dong, Li, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: long papers),* 260–269.

Ghannay, Sahar, Benoit Favre, Yannick Esteve, and Nathalie Camelin. 2016. Word embedding evaluation and combination. In *Proceedings of the tenth international conference on language resources and evaluation (lrec'16),* 300–305.

Li, Xin, and Dan Roth. 2002. Learning question classifiers. In *Coling 2002: the 19th international conference on computational linguistics.*

Mathew, Leeja, and VR Bindu. 2020. A review of natural language processing techniques for sentiment analysis using pre-trained models. In *2020 fourth international conference on computing methodologies and communication (iccmc),* 340–345. IEEE.

McCann, Bryan, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: contextualized word vectors. *Advances in neural information processing systems* 30.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

Mishra, Amit, and Sanjay Kumar Jain. 2016. A survey on question answering systems with classification. *Journal of King Saud University-Computer and Information Sciences* 28 (3): 345–361.

Moldovan, Dan, Marius Paşca, Sanda Harabagiu, and Mihai Surdeanu. 2003. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems (TOIS)* 21 (2): 133–154.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning. 2014. Glove: global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp),* 1532–1543.

Rezaeinia, Seyed Mahdi, Ali Ghodsi, and Rouhollah Rahmani. 2017. Improving the accuracy of pre-trained word embeddings for sentiment analysis. *arXiv preprint arXiv:1711.08609.*

Salton, Gerard, and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24 (5): 513–523.

Siami-Namini, Sima, Neda Tavakoli, and Akbar Siami Namin. 2019. The performance of lstm and bilstm in forecasting time series. In *2019 ieee international conference on big data (big data),* 3285–3292. https://doi.org/10.1109/BigData47090.2019.9005997.

Yilmaz, Seyhmus, and Sinan Toklu. 2020. A deep learning analysis on question classification task using word2vec representations. *Neural Computing and Applications* 32:2909–2928.

Appendix

	Groud Truth Label	Predict Label	Question
2	NUM:dist	DESC:manner	How far is it from Denver to Aspen ?
3	LOC:city	ENTY:cremat	What county is Modesto , California in ?
4	HUM:desc	HUM:ind	Who was Galileo ?
5	DESC:def	DESC:def	What is an atom ?
6	NUM:date	NUM:date	When did Hawaii become a state ?
7	NUM:dist	NUM:dist	How tall is the Sears Building ?
8	HUM:gr	HUM:gr	George Bush purchased a small interest in which baseball team ?
9	ENTY:plant	ENTY:cremat	What is Australia 's national flower ?
10	DESC:reason	DESC:reason	Why does the moon turn orange ?
11	DESC:def	DESC:def	What is autism ?
12	LOC:city	LOC:city	What city had a world fair in 1900 ?
13	HUM:ind	ENTY:other	What person 's head is on a dime ?
14	NUM:weight	NUM:period	What is the average weight of a Yellow Labrador ?
15	HUM:ind	HUM:ind	Who was the first man to fly across the Pacific Ocean ?
16	NUM:date	NUM:date	When did Idaho become a state ?
17	NUM:other	NUM:period	What is the life expectancy for crickets ?
18	ENTY:substance	ENTY:cremat	What metal has the highest melting point ?
19	HUM:ind	HUM:ind	Who developed the vaccination against polio ?
20	DESC:def	DESC:def	What is epilepsy ?

Figure 4. Visualization of the Outputs