COMP61332 Text Mining – Question Classification

Abhinav Thomas (T42413AT), Anusha Haleema (M35218AH), Aryaan Shome (P67841AS), Regson Do Rego (A26412RD), Shefali Singhal (J59842SS) and Shruti Govind (K23216SG)

Abstract - The goal of this project is to design and implement a question classifier based on two popular Sentence Representation models in the Statistical Method of text mining viz. Bag of Words (BoW) and BiLSTM. The aim is to further compare their performances and conclude which method gives maximum accuracy. Based on the fact that the dataset given to us has labelled questions, this paper analyses and states that the best percentage for splitting the dataset for training-validation purposes is 10%. The ease of classification of the various classes has also been assessed and ENTY classes have been found to be the most difficult to classify . This paper also scrutinizes the Confusion Matrices generated on implementing the Sentence Representation models by making use of accuracy and weighted F1 scores as parameters. The code used can be found at https://github.com/abhinavthomas/question-classification.

1 Introduction

Natural Language Processing (NLP) steers Question Classification (QC) towards accurately categorizing questions into classes with labels. QC is a type of sequence classification which can be performed using multiple approaches. We aim to discuss and demonstrate that in this paper. Given a set of labels, we have to determine which class each sentence belongs to. QC is an important research problem because its application helps to limit the search space to find the most appropriate response from a set of search results. QC along with named entity recognition and entity linking is required for Question Answering System (QAS). QAS aims to improve user experience by returning relevant results as responses to the user queries. The improvement of these systems over the years has aided the evolution of web chatbots and mobile virtual assistant softwares such as Siri, Alexa and Cortana.

Most errors caused by QAS are due to question misclassification as this heavily influences the expected answer. An effective way of obtaining the correct answers is to evaluate only the answer candidates that follow the results of question classification while restricting the answer candidates to a pool of smaller answer candidates. For example, the answer type to "Who was the first president of America?" will be a person's name, so the search space can be limited to "names" and "America" only. Therefore, it is beneficial to classify the questions correctly as better question classification would automatically improve the overall performance of the QAS.

In this project, we experiment with the various approaches to question classification and compare their performances in an attempt to determine which implementation method manages to achieve the best accuracy. Before working with the questions, an initial data cleanup is required since there are several factors and problems which we must take into account.

2 Methodology

There are two majors methods used for question classification viz. Symbolic (Rule-based) and Statistical (Machine learning-based). The former involves focusing on pattern-matching and parsing, but since the dataset already had the labels for the questions, we used the Statistical approach. It makes use of an Artificial Neural Network (ANN) to perform the text classification. The project mainly articulates two models used to convert word embeddings to sentence vectors i.e. Bag of Words (BoW) and BilSTM. The relative semantic meanings of the sentences are generated as vectors and

forwarded into a classifier. The analysis of this can be found in section 4.

2.1 Bag of Words Model

$$vec_{bow}(s) = \frac{1}{|bow(s)|} \sum_{w \epsilon bow(s)} vec(w)$$

Bag of Words is a method of extracting features from the questions for use later in the algorithm. The classifier makes predictions based on sentence vectors with dense dimensionality. It uses 2 types of word embeddings learned by prediction-based approaches:

- **Randomly Initialised Vectors**
- **Pre-trained Word Vectors** - Taken from GloVe (5) embeddings. One of the main issues while making use of pre-trained embeddings is the difficulty in handling words that don't have a vector. This is taken care of by making use of the embedding of the #UNK# token. But the recent researches (6) show that making use of subword tokeniser and aggregating the embeddings for n grams of the words have resulted in better accuracy in most of the NLP problems. We have created an n_gram and simple averaging function to achieve this.

2.2 BiLSTM Model

BiLSTM generates sentence vectors from word embeddings present in a sequence. The ability to access the previous hidden states makes the LSTM network more semantically aware than BoW. BiLSTM tends to be more specific than BoW in the sense that it can store the context of the words of the training set, as opposed to BoW which does not handle word embeddings in an ordered manner. It is a time-series analysis as it contains the meaning of the word with context to the previous as well as future words and so it is bi-directional. For this too, we can randomly initialise word embeddings or make use of pre-trained vectors from word embedding models such as GloVe in our experiment.

3 The Experiment

3.1 Data Split

In our experiment, we have used the question classification data set from Learning Question Classifiers in (2). For the splitting ratio, we tried out a variety of splits ranging from 50% - 50% to 90%-10% for

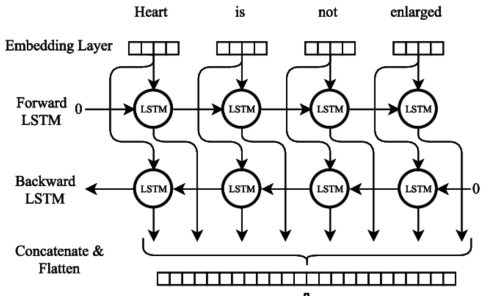

Figure 1: Basic conversion of a sentence to vectors (1)

training and validation respectively. After rigorous experimentation we decided to go with a 90%-10% split for the 5000 entities. 90% of the data was used for training while the rest 10% was used for validation, rounding off to the nearest decimal i.e. 4500 rows for training and 500 for validation as that leads to maximum validation accuracy.

3.2 Preprocessing

Before starting work on this data, it was important to clean it to remove any noise or inconsistency. Having a clean data set allows the models to learn the meaningful features only and reject the irrelevant ones.

- **Capitalization Inconsistency** - Variations in Text Capitalization for the same word occurring multiple times but with different casing. To solve this, we converted all the text to lower case.
- **Irrelevant Characters** - Any non-alphanumeric character is deemed unnecessary and so all of the following characters were removed: . ' , ' ' 's ! ( ) - [ ] ; : ? @ * _ .
- **Stop Words** - Certain words which belong to Part of Speech (PoS) viz. Prepositions, Pronouns, Articles, Conjunctions etc. were removed because they do not convey much information about the sentence. Normally interrogative words like 'what', 'why','how' etc. would be removed as well, but we haven't done that in this case because these words will convey useful information for the classification of questions.
- **Label and Question Separation** - The data set included both labels and questions as one entity, so separating them was necessary using ":" as the delimiter.

- **Vocabulary Selection** - Instead of using all the words in the text corpus, we have used only words which occurred at least 3 times.
  - In order to achieve the same dimension for each sample, we have used the token <pad> and it has been represented as 0 in the encoding and a 0 - vector for its corresponding embedding.
  - The tokens of the sentences not present in the vocabulary are represented using "#UNK#" and encoded as 1. GloVe makes use of the same word token for unknown words. In random embeddings, "#UNK#" is also randomly initialised.

3.3 Other Parameters

- **Learning Rate** - The learning rate for each model was determined after experimentation. The optimal learning rate was found to be 1.5.
- **Batch Size** - We tried to identify the batch size that provided the best accuracy in the least number of epochs (set as 10 in the coursework). It was found to be 64.
- **Embedding Dimensions** - The embedding dimension was taken as 300 so that it matched with the GloVe embedding dimension. But the code was able to handle the varied embedding dimensions for random initialisation.
- **Number of Nodes** - The number of nodes in the hidden layer was set as 50 for BiLSTM. There was 1 hidden layer.

3.4 Embedding Dimensions

Ensemble models were created using model stacking and selection for the class which occurred the most among all the models for the same input. We have used global micro and weighted F1-scores to check the accuracy of the model because this is a multi-class classification problem with an imbalanced set of labels.

4 Results

For testing, the Test Set: TREC 10 questions has been used. As mentioned above the training set was split into 10 portions to prevent look-ahead bias, over-fitting and under-fitting. 9 portions were used for training and one for development and this 10% split yielded maximum accuracy. The labelled vectors generated during sentence representation were given as inputs to the NLP pipeline. A total of 8 models were used for the experiment. Two sentence representation techniques viz. BoW and BiLSTM made use of 2 types of vectors each i.e. Random Initialisation Vectors and Pre-trained GloVe Vectors. They were further combined with enabling and disabling of fine-tuning of the word embeddings. In a class, the more the frequency of a word, the higher is its TF-IDF value. Words with lower frequencies of occurrence are not considered significant enough to affect the results. A frequency of 3 was set to extract words that had the maximum effect on the model. Batch sizes in the range 4, 8, 16, 32, 64, 128, 256, 512 have been used for the experiment. With an optimum batch-size of 64, we were able to train the models with about 50% accuracy in less than 10 epochs with fewer fluctuations. Table 1 depicts the results for each model during training and validation while Table 2 shows the results for each model during testing. The results of the experiments indicate that the lower batch size makes the model unstable but more accuracy is achieved after reaching the stopping condition. On the other hand, when the batch size is larger, less accuracy is yielded at early epochs but the model stabilizes towards the end. It was also found that a learning rate of 1.5 is a sweet spot for BiLSTM but it doesn't work that well for BoW.

Name	Training Accuracy	Validation Accuracy
BoW with Random initialization without fine tuning	69.2%	63.1%
BoW with Random initialization with fine tuning	65.6%	60.9%
BoW with GloVe initialization without fine tuning	57.2%	55.9%
BoW with GloVe initialization with fine tuning	67.2%	65.7%
BiLSTM with Random initialization without fine tuning	98.6%	78.3%
BiLSTM with Random initialization with fine tuning	98.6%	78.0%
BiLSTM with GloVe initialization without fine tuning	98.3%	81.0%
BiLSTM with GloVe initialization with fine tuning	98.5%	82.2%

Table 1: Results of training

Name	Testing Accuracy
BoW with Random initialization without fine tuning	32.8%
BoW with Random initialization with fine tuning	23.2%
BoW with GloVe initialization without fine tuning	54.8%
BoW with GloVe initialization with fine tuning	48.8%
BiLSTM with Random initialization without fine tuning	41.8%
BiLSTM with Random initialization with fine tuning	38.4%
BiLSTM with GloVe initialization without fine tuning	59.4%
BiLSTM with GloVe initialization with fine tuning	68.0%

Table 2: Results of testing

5 Analysis

The findings after thorough examination of the above results are collated as follows:

- **Training-Validation Split** - Use of different percentages of the dataset for training and validation affects the results. It was observed that the maximum accuracy was achieved at 10% training-validation split.

- **Difficulties in Classifying Classes** - From the confusion matrix 5 we can clearly see that, certain classes were slightly complex to classify because they had a high number of False Negatives i.e. they were falsely predicted. For BoW with randomly initialised vectors and no fine tuning the **ENTY:termeq** class was tough to classify. On the other hand, for BoW with randomly initialised vectors with fine tuning, the **LOC:country** class (mainly containing questions such as 'What country?', 'Which country?') were incorrectly predicted. The **ENTY:animal** class was hard to classify in both BiLSTM using randomly initialised vectors with fine tuning as well as without.

- **Confusion Matrix Scrutiny** - We chose to use accuracy and weighted F1 scores (weight based on the number of test data present in each class) to represent the accuracy. These results can be seen in Table 3.

- **Effect of using more pre-processing steps** - Usage and removal of certain words has affected the accuracy in the following ways:

Name	Test Accuracy	F1 Score
BOW random init no fine tuning	32.8%	0.36
BOW random init fine tuning	23.2%	0.28
BOW Glove init no fine tuning	54.8%	0.54
BOW Glove init fine tuning	48.8%	0.50
BiLSTM random init no fine tuning	41.8%	0.40
BiLSTM random init fine tuning	38.4%	0.39
BiLSTM Glove init no fine tuning	59.4%	0.62
BiLSTM Glove init fine tuning	68.0%	0.67
Ensemble Model	66.4%	0.67

Table 3: Results of Confusion Matrix and F1 Scores

  - **Stop Words** : Removing some words from the list of Stop Words in the training set helped increase the accuracy. Interrogative words such as 'How', 'What', 'When' etc. were extremely important for the question classification algorithm. Failure to use these words resulted in loss of information and reduced accuracy by a great extent.
  - **Punctuation** : For each punctuation in the data set a token was created during the classification which was redundant. Hence, removal of punctuation helped increase the accuracy of the model.
  - **Lower Casing** : All words were put in lower case because when words are case sensitive they are treated as different words. When they are all in the same case the word embedding is the same. This helped the model classify words properly. Therefore, putting all the words in the lower casing increased accuracy.
  - **Frequency of Words** : In a class, if a word occurs multiple times then it's importance increases. At the same time, if a word occurs very few times then it does not have much effect on the classification. A frequency of 3 was set to extract the words that have maximum effect on the model.

5.1 Batch Size vs Accuracy Plots

Figure 2 shows visual graphs for the 8 models optimized using 8 different batch-sizes. Optimumm results were found at a batch size of 64.

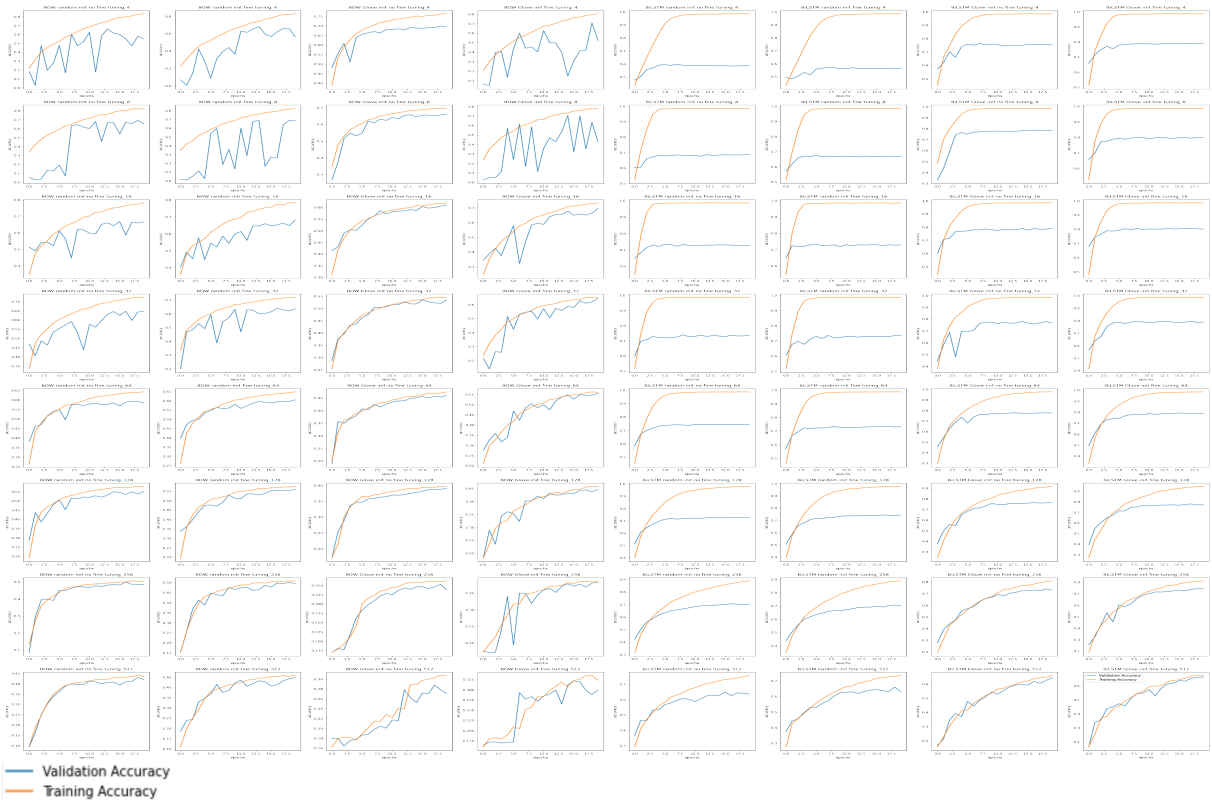

Figure 2: The effect of experimenting with various batch sizes in the range 4, 8, 16, 32, 64,128, 256, 512 for 8 models. Thus, 64 plots. Each row represents nth batch-size.

5.2 Training and Validation Accuracy Vs Epochs plots using optimised parameters (LR = 1.5, Batch-size = 64), Figure 3

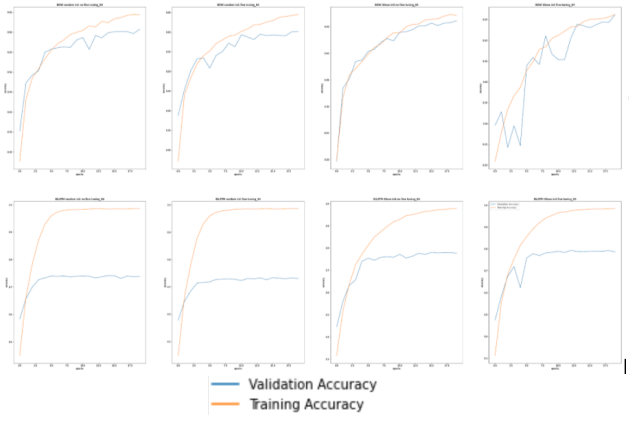

Figure 3: Left to right - BOW random initialization without freeze, BOW random initialization with freeze, BOW GloVe initialization without freeze, BOW GloVe initialization with freeze, BiLSTM random initialization without freeze, BiLSTM random initialization with freeze, BiLSTM GloVe initialization without freeze, BiLSTM GloVe initialization with freeze.

Name	Learning Rate
BOW random initialization with freeze	2.2
BOW random initialization without freeze	2
BOW GloVe initialization with freeze LR	2.8
BOW GloVe initialization without freeze	2.8
BiLSTM random initialization with freeze	1.5
BiLSTM random initialization without freeze	3
BiLSTM GloVe initialization with freeze	2.6
BiLSTM GloVe initialization without freeze	2.2

Table 4: Results for the Learning Rate vs Accuracy graphs in Figure 4

5.3 Learning Rate vs Accuracy

For batch size 64 we tried to analyse how it performs for varied learning rates. As we wanted a fast learner model, we tried to identify the best learning rate for each model that yielded the maximum validation accuracy within 10 epochs, Figure 4 shows the graphs and Table 4 shows the selected results.

5.4 Train-validation set split Vs Accuracy

This section demonstrates the effect of using different percentages of the dataset for training and validations. It was observed that the maximum accuracy was achieved at 10% train-validation split. Figure 6

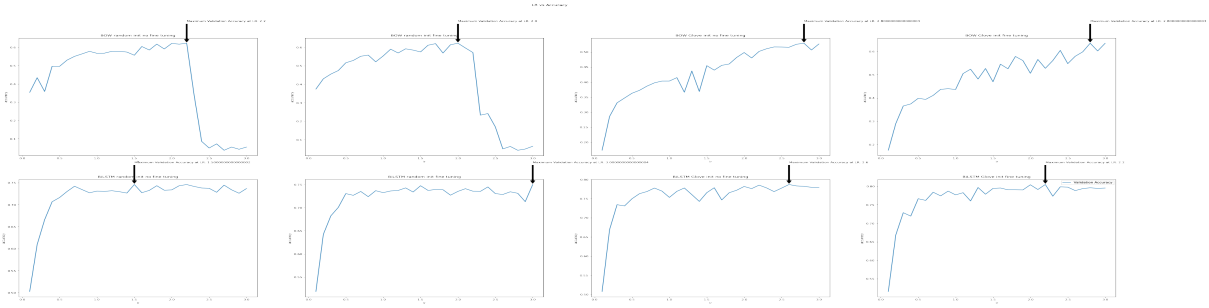

Figure 4: Plots for Learning Rate vs Accuracy graphs. Left to right - BOW random initialization without freeze, BOW random initialization with freeze, BOW GloVe initialization without freeze, BOW GloVe initialization with freeze, BiLSTM random initialization without freeze, BiLSTM random initialization with freeze, BiLSTM GloVe initialization without freeze, BiLSTM GloVe initialization with freeze

6 Ablations Study

6.1 The effect of freezing/fine-tuning the pre-trained embeddings

If we freeze the pre-trained word embeddings, then the weights of the word embeddings are preserved i.e. they are not updated during the learning process. The idea behind doing this is that if for some reason we do not have a good amount of word embeddings and the data that we are using has some tokens that are absent in the word embeddings, then these new tokens could have some synonymous pairs. But, by using gradient updates this type of connection would be lost. A drawback of this is that all the unknown words will be treated the same and will be kept frozen, thus failing to model them effectively (3).

Fine-tuning the pre-trained word embeddings helps in improving the model performance. This is done by intuitively searching for the best value of parameters such as the learning rate, batch size and hidden layers which will give us the highest accuracy.

Based on our findings, we can see that a question classification model with fine-tuned pre-trained word embeddings gives a higher accuracy on the test dataset than the one with frozen pre-trained word embeddings.

6.2 The effect of using randomly initialized word embeddings instead of pre-trained word embeddings

If we use randomly initialized word embeddings instead of pre-trained word embeddings, then we would be picking the word vector at random. The problem with this method is that the resulting embedding space would have no structure. For example, if we take the words ‘surprise’ and ‘astonish’, then they might end up with completely different word embeddings even though they are interchangeable in most of the sentences. This makes it very hard for a deep neural network to make sense of such a noisy, unstructured embedding space (4).

On the other hand, if we use pre-trained word embeddings, then that means we would be using embedding vectors from a pre-computed pre-embedded space that are known to be highly structured and show useful properties that capture generic aspects of the language structure. The reason behind using pre-trained word embeddings is that we do not have enough data available to holistically learn the powerful features on our own while expecting the common semantic features to be present. So, it makes sense to reuse features learned on a different problem (4).

An important conclusion that we can draw from our experiment is that a question classification model using randomly initialized word embeddings yields a lower accuracy on the test dataset while the same model using pre-trained GloVe embeddings yields a higher accuracy.

6.3 Batch Size Optimisation Analysis

We tested the models with batch-sizes in the range (4, 8, 16, 32, 64,128, 256, 512). It was found that with a lower batch-size, the model is unstable (accuracy varies in consequent epochs) but becomes more accurate on reaching the stopping condition. When the batch size is larger, the accuracy is less at early epochs but the model becomes more stable as it reaches the end. Keeping this in view, the optimum batch size was taken to be 64 and it yielded a good trade-off between accuracy and stability.

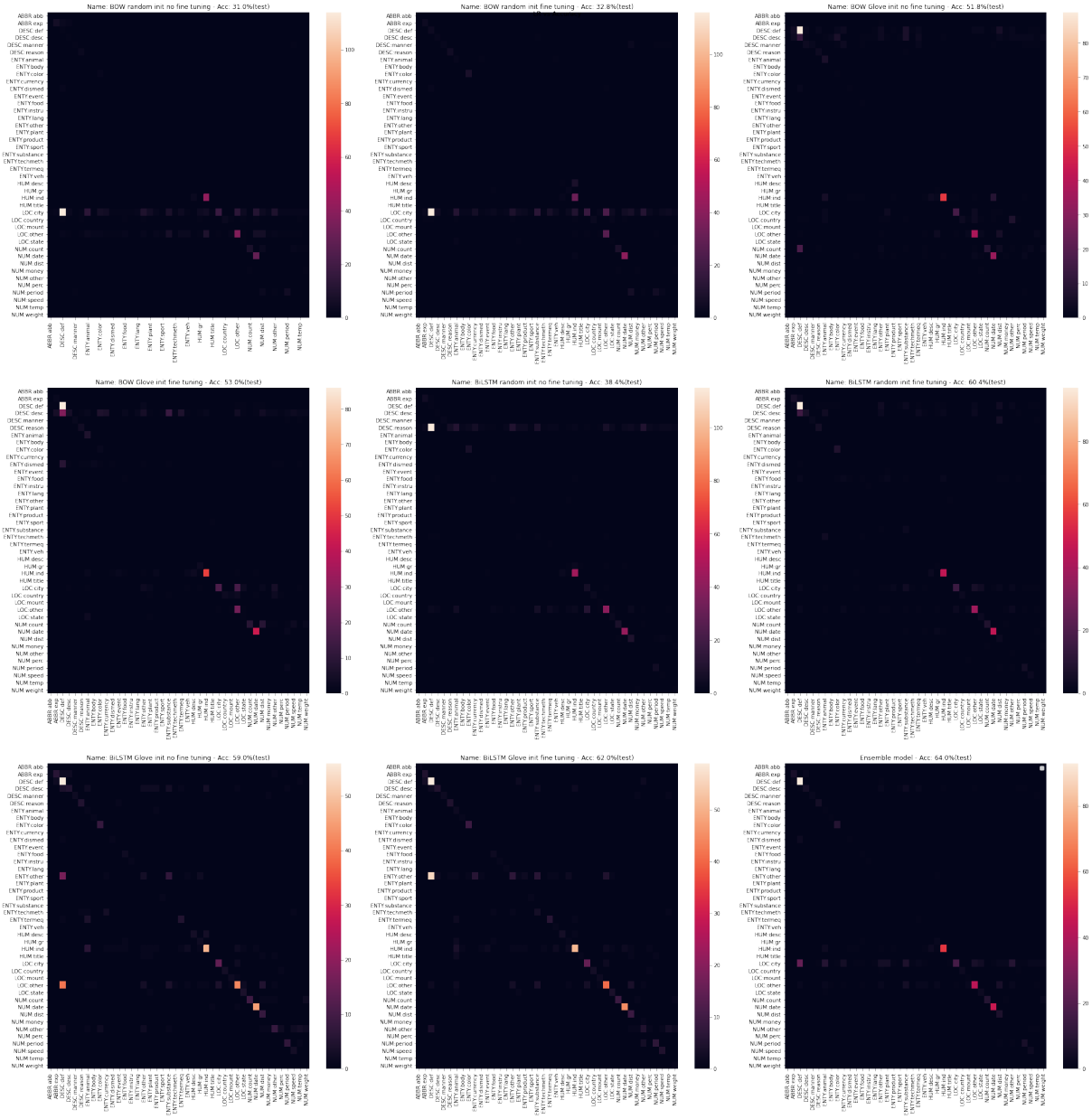

Figure 5: Left to right - BOW random initialization without freeze, BOW random initialization with freeze, BOW GloVe initialization without freeze, BOW GloVe initialization with freeze, BiLSTM random initialization without freeze, BiLSTM random initialization with freeze, BiLSTM GloVe initialization without freeze, BiLSTM GloVe initialization with freeze.

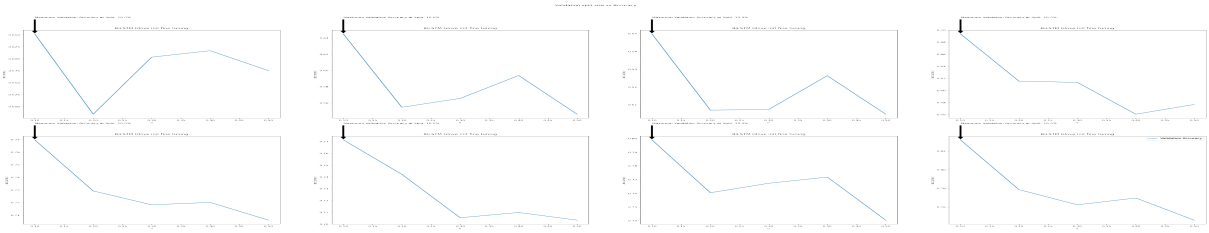

Figure 6: Left to right - BOW random initialization without freeze, BOW random initialization with freeze, BOW GloVe initialization without freeze, BOW GloVe initialization with freeze, BiLSTM random initialization without freeze, BiLSTM random initialization with freeze, BiLSTM GloVe initialization without freeze, BiLSTM GloVe initialization with freeze

References

[1] Savelie Cornegruta, Robert Bakewell, Samuel Withey, and Giovanni Montana. 2016. Modelling radio-logical language with bidirectional long short-term memory networks. pages 17–27

[2] Xin Li and Dan Roth. 2002. Learning question classifiers. In Proceedings of the 19th international conference on Computational linguistics - Volume 1 (COLING '02). Association for Computational Linguistics, USA, 1–7.

[3] Lee, Jaejun Tang, Raphael Lin, Jimmy. (2019). What Would Elsa Do? Freezing Layers During Transformer Fine-Tuning.

[4] Chollet, F., Allaire, J. J. (2018). Deep learning with R.

[5] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation

[6] Kudo, Taku Richardson, John. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. 66-71. 10.18653/v1/D18-2012.