APPLIED MACHINE LEARNING SYSTEMS II (ELEC0135 19/20) REPORT

SN: 19132770

ABSTRACT

In this report, two LSTM neural networks are designed to solve subtask A and subtask B of SemEval 2017 Task4 challenge. These two tasks are sentiment analysis tasks, in which classifying the sentiment of tweets into different polarities. Moreover, this report is focusing on the designing, selection and implementation of model used to solve two tasks. The accuracy score for Task A model is 61%, and 86% for Task B model. These two model are not performing good enough in solving these two tasks, and the reasons behind and improvements are discussed in this report. Lastly, the code of these models can be found in the GitHub link[5].

1. INTRODUCTION

This project is choosing to work on subtask A and subtask B of "Sentiment Analysis on Twitter" challenge. Subtask A is a three-class message polarity classification task, which includes classifying whether a message is of positive, negative, or neutral sentiment. On the other side, subtask B is a two-class topic-based message polarity classification task, which includes classifying whether a topic-based message of positive or neutral sentiment.

There are three key steps in solving the above two subtasks. These key steps are text processing, training word embeddings, and designing neural networks. Firstly, the text processing step is essential for Natural Language Processing (NLP) tasks, since informal writing and misspelling words are confusing for classifiers to learn. Secondly, word embeddings express words as vectors that help machines to understand the relationship between words. Lastly, Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) are potential solvers for these NLP tasks. CNN models are fast and easy to train, and RNN models can process text sequentially (understand word orders).

This report is organized as follows. In section 2, some potential approaches to the tasks are discussed. In section 3, models using for each task are described and explained. In section 4, the detailed implementation of models is provided. In section 5, experimental results are described and analysed. Lastly, section 6 summarizes findings and suggests future improvements.

2. LITERATURE SURVEY

There are many potential approaches to solve these two subtasks, for example, Support Vector Machine (SVM) and Bayesian classifier. However, these two example classifiers are heavily relying on hand-crafted features. In other words, essential features in a sentence need to be extracted first before feeding into such classifiers. On the contrary, Artificial Neural Networks (ANN) perform feature learning, which filters out the redundant features and extracts essential features. Therefore, two state-of-the-art ANN models that solve subtasks A and B are introduced in this section.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (1)$$

The first introduced ANN model is designed by team *DataStories*. They ranked the first in subtask A with an average recall of 0.681. The recall is defined in equation 1, which can be interpreted as the ratio between true positives and predicted results.

This model consists of an embedding layer, two bidirectional Long Short Term Memory (Bi-LSTM) layers, and an attention layer. With the input of a sequence of words, the embedding layer project words into low-dimensional vectors. In this case, *DataStories* trained the word embeddings using the Global Vector (Glove) model on a collected dataset (containing 330M English Twitter messages) [1]. Furthermore, the weights of the embedding layer of this model are initialized with the pre-trained word embeddings.

Next, the LSTM cells take words-converted vectors from twitter messages and produce the word annotations $h_1, h_2, \dots, h_N \in \mathbb{R}^h$, which contain encoder hidden states at each time step. For each LSTM cell, the previous hidden state is concatenated with the current input vectors. Therefore, the LSTM cell has access to all the information in a twitter message up to the current input word from the previous hidden state. Furthermore, the Bi-LSTM cell consists of a forward LSTM cell and backward LSTM, which allows word annotations to be produced by summarizing information from both directions. *DataStories* stack two layers of BiLSTMs to better summarise abstract features in a twitter message [2].

$$a_t = \sum_{i=1}^{N} a_i^t h_i \in \mathbb{R}^h \quad (2)$$

Lastly, the attention layer is added on top of BiLSTM layers to find the relative contribution of each word. The attention layer assigns a weight a_i to each encoder's hidden state. From equation 2, a_t as attention output takes a weighted sum of encoder hidden states, where h_i is encoder hidden state and a_i^t is attention distribution for each encoder hidden state. As

a result, the contribution of important elements in a sentence is amplified through the attention mechanism.

The second introduced ANN model is an ensemble of LSTMs and CNNs with multiple convolution operations. This model is designed by team *BB_twtr* and ranked the first in subtask B with an average recall of 0.882.

$$C_{max} = max(c) \quad (3)$$

Firstly, Convolutional Neural Networks (CNNs) are used in this model. CNNs take words as input and map them into a matrix $s \times d$, where s is the number of words in the tweet, and d is dimensions of word embeddings [3]. The zero-padding strategy is applied to standardize the lengths of twitter as it does add extra meaning to tweets. Next, the max-pooling operation is conducted to each convolution to extract the most important features. From equation 3, C_{max} as the most important feature is extracted through the max-pooling layer to represent the pool-size matrix.

Secondly, LSTMs are also used to construct this model. *BB_twtr* chose to use BiLSTM to solve the post information failure problem, where LSTMs can only read in one direction. Also, dropout layers with a probability of 50% are used before and after LSTM layers to reduce overfitting.

Lastly, the formation of this ANN model is through soft voting on a combination of 10 CNNs and 10 LSTMs, where these models ensembled are initialized randomly and trained with different epochs, different filter sizes, and different pre-trained word embeddings [3]. These procedures are applied to reduce variance and further increase the average recall and accuracy of the ensembled model.

3. DESCRIPTION OF MODELS

The models used to solve both Task A and Task B are Recurrent Neural Networks (RNN). In fact, RNN and CNN are both capable of completing sentiment analysis tasks with satisfying results. In comparison, CNN is easier and faster to train. Moreover, CNN can better capture the important features of text through a list of pooling layers compared to RNN. However, RNN is more suited to perform text classification tasks because RNN can process text sequentially.

$$h_t = f_W(X_t, h_{t-1}) \quad (3)$$

In other words, RNN is learning text input using hidden states with timestep. Equation 3 can be interpreted as the hidden state of timestep t h_t is dependent on the weights W of the model, X_t the input text, and h_{t-1} the previous hidden state.[2] On the other side, It is difficult for CNN to make sense from important features extracted because CNN has no notion of word order. Therefore, the LSTM network, as a variant of RNN, is chosen to solve both two tasks.

To design the best fit model of each task, 27 models with different layer sizes and numbers of hidden layers are first built. To be specific, these 27 models are built from [1, 2, 3] BiLSTM layers, [0, 1, 2] dense layers and [32, 64, 128] layer sizes. After the best-performed model is selected, the selected model is then improved by choosing to add Dropout layers, Gaussian Noise layers, and others.

3.1. Task A: Message Polarity Classification

Task A is a three-class classification task in which a BiLSTM model is trained to classify message sentiment into positive, negative, and neutral.

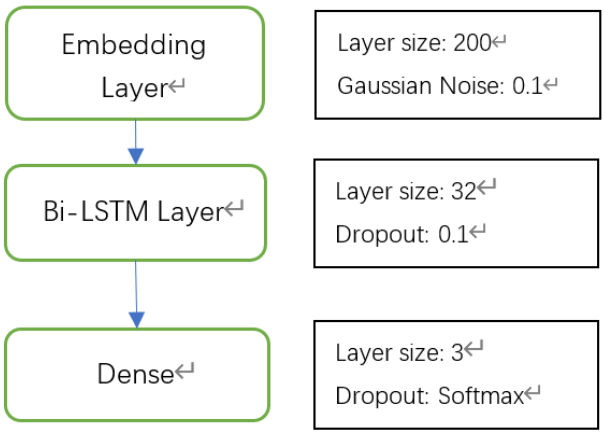

Figure 1. Flowchart of Task A model

The flowchart in Figure 1 illustrates that Task A model is made from three layers. The first layer is an embedding layer with 200 nodes and a 0.3 Gaussian noise rate. The second layer is a bidirectional LSTM layer with 32 nodes and a 0.1 dropout rate. The final layer is a dense layer with three nodes and activated using the Softmax function, which is commonly used in multi-class classification task models.

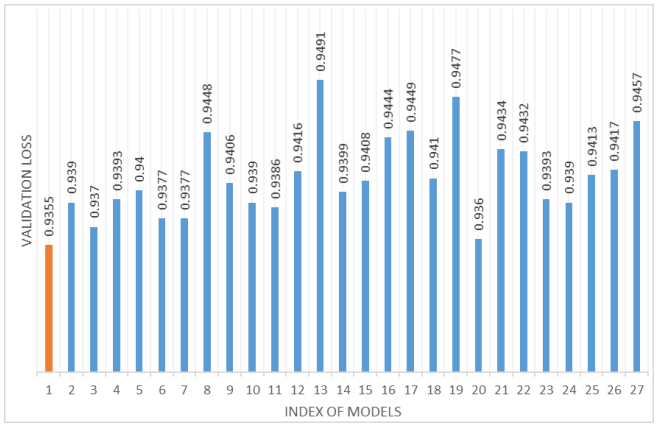

Figure 2. Validation loss of 27 test models for Task A

As introduced at the beginning of this section, 27 differently-structured models are built and compared. These 27 models are trained by running through the training dataset ten times (10 epochs) with a learning rate of 0.0001. Most of the

test models start to converge after the 2nd or the 3rd epoch and getting overfitted after the 6th epoch. Among 27 test models, the first model has the lowest validation loss (0.9355) after the 4th epoch. To be more specific, this selected model is made from an embedding layer of 200 nodes, a bidirectional LSTM layer of 32 nodes, and a dense layer of 3 nodes.

	Validation Loss
Test Model 1	0.9355
Test Model 1 + Gaussian Noise (0.1)	**0.927**
Test Model 1 + Gaussian Noise (0.3)	0.9318
Test Model 1 + Gaussian Noise (0.5)	0.944

Table 1. Validation loss adding Gaussian Noise layer

Moreover, the selected model is tried to be further improved by adding additional layers and adjusting hyperparameters. The first layer considered to add is a Gaussian noise layer. By adding a Gaussian noise layer, overfitting problem can be mitigated. As can be seen from Table 1, there is a decrease in validation loss when a Gaussian Noise layer of 0.1 or 0.3 is added to the model. The Gaussian Noise layer of 0.1 is added to the model after the embedding layer since adding such a layer can further decrease validation loss from 0.9355 to 0.927.

	Validation Loss
Test Model 1 + Gaussian Noise (0.1)	0.927
Test Model 1 + Gaussian Noise (0.1) + Dropout (0.1)	**0.9255**
Test Model 1 + Gaussian Noise (0.1) + Dropout (0.2)	0.9267

Table 2. Validation loss adding Dropout layer

Lastly, dropout layers are added before and after bidirectional LSTM layers to mitigate the overfitting problem. From Table 2, a dropout layer of 0.1 can help further reduce the validation loss of the selected model to 0.9255.

3.2. Task B: Topic-Based Message Polarity Classification

Task B is a binary classification task which is solved by training a BiLSTM model to classify topic-based message into positive and negative classes.

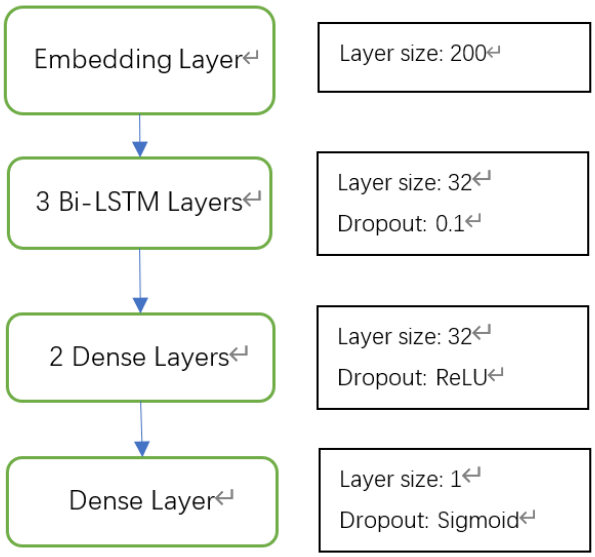

Figure 3. Flowchart of Task B model

The flowchart in figure 3 shows how the best-fit model for solving task B is built. The first layer in this model is an embedding layer of 200 nodes, and the next three layers are bidirectional LSTM layers of 32 nodes with a dropout rate of 0.1. Furthermore, two dense layers of 32 nodes are added and activated using the ReLU function. Lastly, a dense layer with one node is used to classify output into 0 and 1 with the help of a sigmoid function.

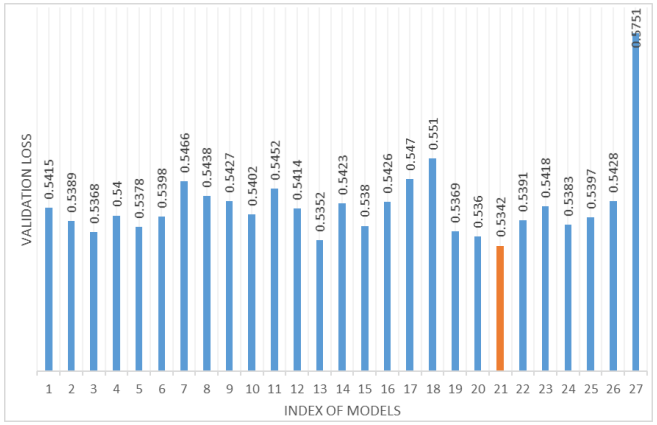

Figure 4. Validation loss of 27 test models for Task B

The lowest validation loss of 27 test models is shown in Figure 4. These 27 test models are trained by running the training dataset 20 times (20 epochs) with a learning rate of 0.00008. Most test models in Task B are converged after the 5th epoch, but there are few test models only start converging after the 10th epoch. This late convergence can be explained from the size of the dataset and the small learning rate. Among these 27 test models, model 21 has the lowest validation loss of 0.5342. This model is made of an embedding layer of 200 nodes, three bidirectional LSTM layers of 32

nodes, two ReLU function activated dense layers of 32 nodes, and Sigmoid function activated final dense layer of 1 node.

	Validation Loss
Test Model 21	0.5342
Test Model 21 + Gaussian Noise (0.1)	0.5419
Test Model 21 + Gaussian Noise (0.2)	0.5508
Test Model 21 + Dropout (0.1)	**0.5321**
Test Model 21 + Dropout (0.2)	0.5367

Table 3. Validation loss adding Gaussian Noise layer and Dropout layer

From Table 3, it can be seen that adding the Gaussian Noise layer to this task model cannot reduce the validation loss of the model. However, the adding of dropout layers of 0.1 can reduce validation loss from 0.5342 to 0.5321. Therefore, a dropout layer of 0.1 is added before and after the bidirectional LSTM layers.

4. IMPLEMENTATION

The implementation details of two task models are similar since both two tasks are solved using the RNN learning algorithm. These similar implementation details are external libraries, hyper-parameter selection, key functions, datasets and splits, and data preprocessing.

4.1. External libraries

The external libraries used for building text classification models are pandas 1.0.3, tensorflow-gpu 2.1.0, and ekphrasis 0.5.1. Starting from pandas, texts in the given dataset are transformed into data-frames for easier management and manipulations. Next, tensorflow as an open-source deep neural network library, it allows user to build and train complex model easily. In this case, tensorflow helps to design and implement RNN models needed to perform both sentiment analysis tasks. Lastly, ekphrasis, as a text processing tool, is used in this project to perform word tokenization, spell correction, and other preprocessing tasks.

4.2. Hyper-parameters

There are many hyper-parameters to be determined in the deep neural networks, and these parameters can be split into two categories in general, hidden layer parameters and model parameters. Hidden layer parameters include the number of layers, number of neurons in each layer, activation functions, and others. On the other side, model parameters include loss function, learning rate, batch size and others.

4.2.1. Hidden Layer parameters

There are three steps in determining hidden layer parameters. Firstly, select a well-performed general model. In this case, 27 basic bidirectional LSTM models are built with a different number of LSTM layers [1, 2, 3], number of dense layers [0, 1, 2], and number of nodes in LSTM layers and dense layers [32, 64, 128]. The model with the lowest validation loss is then selected to be the well-performed general model. Next, Tuning hyperparameters in each layer of this model carefully to achieve better-fitted model. The tuned parameters include the number of nodes in embedding layers, rate of Gaussian noise layers, rate of dropout layers, activation functions. The result of fine-tuned hidden layer parameters can be viewed from Figure 1 and Figure 3 in section 3.

4.2.2. Model parameters

Start with loss function; sparse categorical cross-entropy is used in Task A model because Task A is a multi-class classification problem. The sparse categorical cross-entropy loss function provides sparsity in neuron network (most of the weights are 0), which increases time and space efficiency. On the other side, the binary cross-entropy loss function is selected for the task B model to cope with a binary classification problem

Next, the learning rate is determined to be 0.0001 for Task A model and 0.00008 for task B model. The difference between two learning rate choices is because task B has a smaller dataset and more easily to be overfitted. To help model find the convergence point, a lower learning rate is selected. The learning rate selected ensures two task models converge quickly enough with satisfying results.

Lastly, the batch size is determined to be 64 for both task models because a larger batch size can lead to an increase in loss, and smaller batch size can increase training time.

4.3. Key functions

There are two key functions created for solving Task A and Task B, and they are preprocessing function and model function.

The preprocessing function takes four inputs (test size, validation size, buffer size, batch size), and returns four outputs (training data, test data, validation data, vocab size). The test size and validation size are referring to the size of the test dataset and validation dataset, and these two parameters are used to split the given dataset into training, testing, and validation parts. Next, the buffer size is needed to randomize the dataset properly. In this case, a buffer size close to the size of the entire dataset is chosen to randomize the dataset uniformly. Lastly, the batch size is also needed to be determined and feed to the preprocessing function because the length of tweets is different. For better training results, tweets are processed in batches and padded with zeros. The outputs of this

function are then taken as inputs of the model function to develop the deep neural network model.

The model function takes four inputs (vocabulary size, training dataset, validation dataset, test dataset). The vocabulary size is needed to initialize the embedding layer of the testing model. Moreover, the training dataset and the validation dataset are used to train test models. Lastly, the test dataset is used to test the trained model with some data it has never seen before.

4.4. Datasets and split

The dataset used to train sentiment analysis models for Task A and Task B is provided by SemEval challenge organization. To be specific, the dataset used for training task A model is "SemEval2017-task4-dev.subtask-A.english.INPUT.txt", which can be found on the website of SemEval 2017 Task 4 [4]. Furthermore, the dataset used to train Task B model is "SemEval2017-task4-dev.subtask-BD.english.INPUT", which also can be found on the same website.

	Positive	Negative	Neutral
Numbers	7059	3231	10342

Table 4. Distribution of Task A dataset

Table 4 shows the distribution of Task A dataset, and this dataset is highly biased. To train a well-performed model, the size of the dataset for each category should be similar. Among these three categories, negative labeled data is only accounted for about 16 percent of the entire dataset. On the contrary, neutral labeled data accounts for almost 50 percent of the entire dataset for a three-class classification problem.

	Positive	Negative
Numbers	8212	2339

Table 5. Distribution of Task B dataset

From Table 5, positive labeled data accounts for 78 percent of the entire dataset, and the negative labeled data accounts for the rest. The model trained using this dataset can be highly biased, with low accuracy and high loss.

The dataset of both tasks is split into three different sets, training set, validation set, and test set. Training set and validation set are used during the training phase, where the training set is used to train the model, and the validation set is used to reflect the real loss and accuracy of the model. This is because the model will try to remember the entire training dataset to have higher training accuracy, and that is where overfitting begins. The test set is to test how good is the trained model by showing the model data it has never seen during the training phase.

The separation of both tasks is 70% for the training set, 15% for the validation set, and 15% for the test set. Such separation choice is made to ensure the model is trained with the large and diverse dataset, and test the model with different categories inputs.

4.5. Data preprocessing

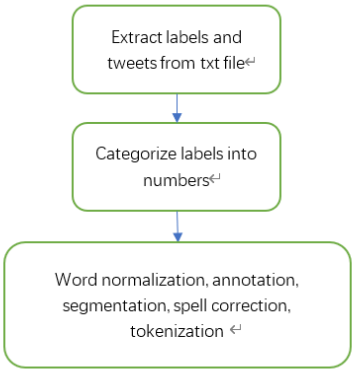

Figure 5. Flowchart of data preprocessing

Figure 5 shows data preprocessing procedures for both tasks. There are three steps in data preprocessing, extraction, categorizing, word tokenization, word normalization, spell correction, and word annotations.

Firstly, labels and tweets are extracted from tab-separated txt file using package pandas. Secondly, labels are converted from strings into numbers because machine learning cannot understand characters.

Thirdly, tweets are normalized, annotated, segmented tokenized, and corrected using text processing tool ekphrasis. URL links, email addresses, money, and date are normalized because these words do not have sentiment values. For instance, the text processing tool normalizes "www.google.com" into "<url>" to help model understand the sentiment of tweets. Next, word annotations can be used to hashtags, emphasised words, and repeated words. For example, "#machine learning" is annotated into "<hashtag> machine learning <hashtag>" to help models to learn tweets. Furthermore, word segmentation is used to separate the connected words. For example, "welovetheearth" can be segmented into "we love the earth" to help understand the sentiment of tweets. Moreover, this text processing tool is able to correct misspelled words. For example, "litle cat" can be corrected to "little cat" to train a better-performed model. Lastly, words in twitter are tokenized and indexed because machine learning algorithms can only deal with numbers instead of words.

4.5. Training convergence

The training convergence and stopping criterion of these two task models are discussed separately with reference of learning curves.

4.5.1. Task A: Message Polarity Classification

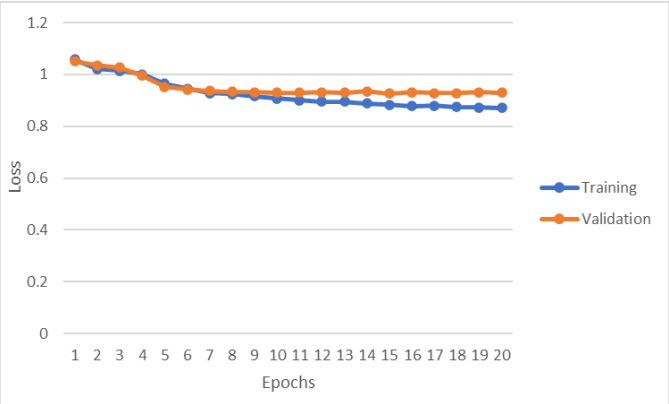

Figure 6. Training and validation loss of Task A model

Figure 6 shows the training loss and validation loss curves of 20 epochs on the same graph. The training loss continuously decreases until the 15th epoch and becomes steady after that. On the other side, the validation loss decreases dramatically in the first four epochs and decreases slightly and constantly in the following 11 epochs. The lowest validation loss of this model is 0.9255 and happens at the 15th epoch. Therefore, the training is stopped at the 15th epoch to reduce the effects of overfitting.

4.5.2. Task B: Topic-Based Message Polarity Classification

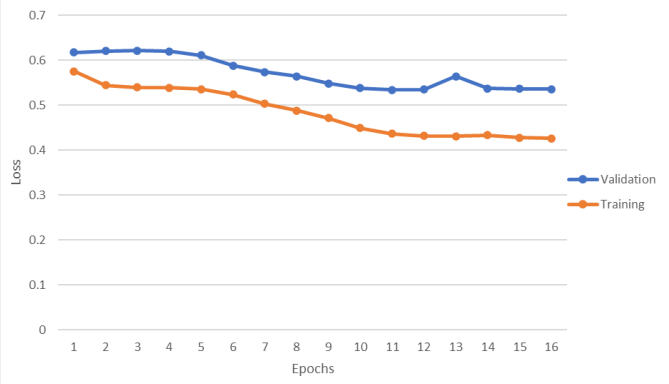

Figure 7. Training and validation loss of Task B model

Figure 7 shows how training and validation loss of Task B model changes during the training phase. The training loss of this model constantly decreases until the 12th epoch. On the contrary, the validation loss only starts decreasing after the 5th epoch and stops decreasing after the 12th epoch. This model is converged after learning the training dataset 12 times since the validation loss curve and the training loss curve are both becoming steady. Moreover, there is a slight increase in the validation loss at the 13th epoch. This increase is the result of the overfitting problem. To be specific, the model is starting to remember the training sets to have lower training loss. Therefore, training of this model is stopped at the 12th epoch with the lowest validation loss of 0.5321.

5. EXPERIMENTAL RESULTS AND ANALYSIS

Task	Model	Train Acc	Val Acc	Test Acc
A	RNN	0.69	0.61	0.61
B	RNN	0.93	0.79	0.86

Table 6. Training, validation and test accuracy of Task A, B

Table 6 shows the accuracy of both task models on training sets, validation sets, and test sets. Training accuracy is the highest because models are experiencing overfitting. The trained model can have higher training accuracy by training more epochs. On the other side, the validation accuracy and test accuracy can reflect the real performance of the trained model. Therefore, trained model with higher validation accuracy and test accuracy is a better-fit model. Task B model performs better than Task A model because Task B is a binary classification problem, and Task A is a three-class classification problem.

5.1. Task A: Message Polarity Classification

The training accuracy of Task A model is 8 percent higher than validation and test accuracy because the model is slightly overfitted. The validation accuracy and test accuracy are the real indicators of model performance. The trained model for Task A can classify 61 percent tweets emotional correctly.

The highest test accuracy is 0.664 in this subtask challenge. In comparison, the test accuracy of this trained model is 5 percent lower than the best-performed model. The reasons behind this accuracy difference are model structures, datasets, data preprocessing. Firstly, Task A model can be improved by adding attention mechanisms or ensemble models. Secondly, the datasets used to train Task A model is the 2017 dataset only. The team that participated in this challenge are provided with previous year dataset. With a more balanced and bigger dataset, a better-performed model can be trained. The reason not to use the previous year model is to reduce the training complexity and training time of the model. Lastly, Team DataStories used 330M collected tweeter to train a word embedding for the embedding layer of the model. With pre-trained word embeddings, the model can perform better.

5.2. Task B: Topic-Based Message Polarity Classification

The training accuracy in Task B is 14 percent and 7 percent higher than validation accuracy and test accuracy, respectively. This huge difference in the accuracy of different datasets is the result of overfitting. The model is trained with too many epochs and starting to remember the training set. Moreover, the difference between validation accuracy and test accuracy is because of the imbalance of the dataset. As described before, this dataset contains 78 percent positive labeled data, and 22 percent labeled data. Therefore, the Task B model is more likely to categorize the sentiment of twitter

as positive. Therefore, low test accuracy is the result of the biased dataset.

6. CONCLUSION

In conclusion, the models designed to solve two subtasks achieved average accuracy scores comparing to the other teams' models. However, the adding of Gaussian Noise and dropout rate into LSTM model helps solving overfitting problem effectively. There is no obvious bounce back in validation loss after the convergence of both task models. Furthermore, bidirectional LSTM layer have better performance than LSTM layer in dealing text classification. Lastly, text preprocessing is essential for text classification tasks.

The model of both tasks can be improved from three aspects, dataset, preprocessing, modelling. Firstly, a better-performed model can be trained if a larger and unbiased dataset is used for training. Secondly, a word embedding can be trained using a lager collected tweeter dataset to help the model learn the relationship between different words. Lastly, attention mechanisms can be added into the model to help identify the sentiment contribution of each word.

7. REFERENCES

[1]"cbaziotis/datastories-semeval2017-task4", GitHub, 2017. [Online]. Available: https://github.com/cbaziotis/datastories-semeval2017-task4. [Accessed: 13- Apr- 2020].

[2]C. Baziotis, N. Pelekis and C. Doulkeridis, "DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis."

[3]M. Cliche, "BB twtr at SemEval-2017 Task 4: Twitter Sentiment Analysis with CNNs and LSTMs", 2017.

[4]"Data and Tools < SemEval-2017 Task 4", Alt.qcri.org, 2017. [Online]. Available: http://alt.qcri.org/semeval2017/task4/index.php?id=data-and-tools. [Accessed: 18- Apr- 2020].

[5] https://github.com/alleno-yu/AMLS_II_assignment19_20