CSI5180

Topics in AI: Virtual Assistants

Module 3 Assignment
Building a paraphrase classifier

Submitted By:

Shubham Nishad 300102830

Submitted On: 8th March 2021

1. Introduction

The goal of paraphrase classification is to determine whether a pair of sentences have the same meaning. Paraphrase classification can be converted to a binary classification problem where the output classes are "paraphrase" and "not paraphrase". Identifying paraphrases is important for various virtual assistant tasks such as Intent detection, Fulfillment etc.

2. Dataset

We are using the SemEval-PIT2015 Dataset [1] for our study. The dataset contains pairs of tweets annotated by experts between a scale of 0 (no relation) and 5 (semantic equivalence).

2.1 Dataset Distribution

We use the original data provided by the author, with the **train set (13,063 sentence pairs)** and **dev set (4727 sentence pairs)** for training the models and **test set (972 sentences)** for the final evaluation of the selected model.

2.2 Label Conversion

The labels in the train/dev set are in the form (x, y) where x is the positive votes and y is the negative votes given by 5 experts. We need to convert them to binary form where 1 map to "paraphrase" and 0 map to "not paraphrase". We do the following conversion for the labels.

(3, 2), (4, 1), (5, 0) → 1 (paraphrase)

(2, 3), (1, 4), (0, 5) → 0 (not paraphrase)

For the test data, we convert output to binary form by doing the following conversion {0,1,2} → 0 and {3,4,5} → 1.

2.3 Examples

Paraphrases

	Tweet 1	Tweet 2	Output
0	EJ Manuel the 1st QB to go in this draft	Can believe EJ Manuel went as the 1st QB in the draft	1
1	Loving the Eagles pick of Stanford TE Zach Ertz	Zach ertz to the eagles is a great pick	1
2	AARON DOBSON to the patriots lets go good pick	Glad the Pats drafted Aaron Dobson	1

Table 1: Some examples of paraphrases in the train dataset.

Non-Paraphrases

	Tweet 1	Tweet 2	Output
0	With the 59th pick the Patriots select WR Aaron Dobson from Marshall	That catch highlight is sick for Aaron Dobson	0
1	50 Cent killing this track for the NFL Draft	50 cent is so lame now	0
2	It fits the larger iPhone 5	Should I get the iPhone 5 or an Android	0

Table 2: Some examples of non paraphrases in the train dataset.

3. Methods

We apply machine learning and deep learning techniques by using packages such as Pandas [2], Scikit-learn [3], Keras [4] and Tensorflow [5]. The code was executed on Google Collab for quicker training and testing.

Algorithm 1: Baseline

Algorithm 2: Machine Learning (SVM) Algorithm 3: Deep Learning (BiLSTM)

3.1 Baseline Algorithm

The baseline algorithm is based on the **ratio between the no. of common characters in both strings and the length of the smaller string**. This works on the assumption that in a paraphrase most of the words are common. The ratio is between 0 and 1, where 1 represents both strings are matching. This method is the modification of the idea provided by B. Roth [6].

$$val = \frac{len(set(s1, s2))}{min(len(s1), len(s2))}$$

Fig 1. Formula for calculating the ratio in the baseline algorithm

For our study if the value of the ratio is above a certain threshold, we classify the strings as paraphrase. Through experimentation, we set the threshold value at 0.8 . Any pair of tweets producing the value > 0.8 are classified as paraphrases. The accuracy of baseline was 52.27%.

3.1.1 Correctly classified

	Tweet 1	Tweet 2	Ratio	Baseline	Output
0	My first podcast drops on iTunes	iTunes deleted my voice memos	0.689	0	0
1	Do not Amber Alert me	Look Im tired if these damn Amber Alerts	0.952	1	1

Table 3. Correctly classified instances in the dev set

3.1.2 Incorrect classified

	Tweet 1	Tweet 2	Ratio	Baseline	Output
0	AAP is in the Adidas commercial	AAP in that Adidas Commercial lol	0.548	0	1
1	I hope to win a 50 Amazon Gift Code	According to Amazon s review	0.896	1	0

Table 4. Incorrect classified instances in the dev set

3.1.3 Illustration of Working

Step1: len (set (“Do not Amber Alert me”, “Look Im tired if these damn Amber Alerts”)) / min(21, 43)

Step 2: 20/21 → 0.952

Step 3: 0.952 > 0.8 → Output: 1

3.2 Machine Learning Classifier

To improve upon the accuracy of baseline algorithm and make use of the large training dataset, we choose three machine learning models: **Multinomial Naïve Bias, Support Vector Machines (SVM) and Random Forest**. Along with the three models we also experimented with two different vectorization techniques: **TF-iDF** and **Word Embeddings.**

Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation. This helps us to improve performance over TF-iDF. For our study we choose the Glove embeddings trained on Wikipedia text [7].

From the possible six combinations, we selected **SVM with word embeddings** as it gave better accuracy and F1-score.

3.2.1 SVM Working with Example

S1 = “*Do not Amber Alert me*” S2 = “*Look Im tired if these damn Amber Alerts*”

SVM works by creating a plane/line of separation between the two classes. A new instance is labelled according to the side of the plane it lies to.

Step 1: The input strings are concatenated to a single input string S.

Step 2: The string S is then vectorized using the pre trained word embeddings. Eg. [“hello”] → [0 0 54 0 11 67]

Step 3: SVM tries to form the hyper plane using mathematical formulas so that the two classes are separated.

Step 4: The position of the plane changes as data changes, once finalized the SVM can be used for classification.

Step 5: The strings are given as input in vectorized form and the plane decides the final output class for the pairs.

3.3 Deep Learning Classifier

As we have large amount of training data deep learning is a good option. Deep learning has been found to the improve performance of text-based classifiers. We apply Bidirectional LSTM with Max pooling. For word embeddings we again use Glove. The model is trained for 10 epochs with a batch size of 512.

Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems such as text classification.

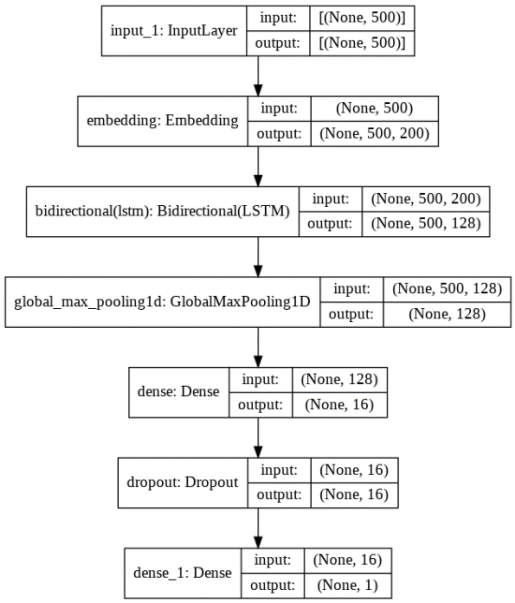

Fig 2. Architecture of the BiLSTM model for this study

3.3.1 BiLSTM Working with example

S1 = “*Do not Amber Alert me*” S2 = “*Look Im tired if these damn Amber Alerts*”

Step 1: The input strings are concatenated to a single input string S.

Step 2: The string S is then vectorized using the pre trained word embeddings. Eg. [“hello”] → [0 0 54 0 11 67]

Step 3: We apply BiLSTM along with Maxpooling to reduce the dimension of the data

Step 4: Finally using the sigmoid function, we convert the data to binary output of 1 (paraphrase) and 0 (!paraphrase)

4. Results

Performance of Baseline on the Dev set

Baseline	Accuracy	Precision	Recall	F1 Score
Dev Set	52.27	0.440	0.440	0.440

Performance of Machine Learning (SVM + Glove) on the Dev set

SVM + Glove	Accuracy	Precision	Recall	F1 Score
Dev Set	68.90	0.5	0.344	0.407

Performance of Deep Learning (BiLSTM + Glove) on the Dev set

BiLSTM + Glove	Accuracy	Precision	Recall	F1 Score
Dev Set	69.22	0.492	0.486	0.455

We compare the three algorithms based on a combination of performance metrics. While accuracy is important, we also focus on Precision and Recall. Low precision results in incorrect answers. Low recall results in the model not understanding the task properly.

We want to maximize both the recall and the precision, so we introduce F1 score. F1 score is the harmonic mean of precision and recall, it can capture both the properties.

For selecting the best model based on dev data, we see the accuracy and the F1 score. The deep learning BiLSTM algorithm is the best in terms of accuracy and F1 score. Therefore, we run the test data on BiLSTM for the final evaluation.

Performance of Deep Learning (BiLSTM + Glove) on the Test set

BiLSTM + Glove	Accuracy	Precision	Recall	F1 Score
Test Set	64.91	0.496	0.490	0.452

On running the Deep learning BiLSTM model on the Test Data we see that it produces results similar to the Dev set.

Overall, it can be said that using Complex architecture like deep learning and using different kinds of word embeddings can significantly improve the result over the baseline model.

5. Conclusion

Paraphrase classification is a difficult problem to solve. Through the study we have seen that just like any other NLP problem, we can see huge improvements in the performance by applying deep learning techniques.

In the future to improve the performance we can introduce some text preprocessing techniques such as lemmatization. Also, we can look at ways to include the POS tags in the deep learning process. We believe that the combination of preprocessing, deep learning and POS tagging can increase the accuracy tremendously.

6. References

1. SemEval-2015 Task 1: Paraphrase and Semantic Similarity in Twitter (PIT) https://github.com/cocoxu/SemEval-PIT2015
2. pandas - Python Data Analysis Library https://pandas.pydata.org/
3. scikit-learn Machine Learning in Python https://scikit-learn.org/stable/
4. Keras: the Python deep learning API https://keras.io/
5. TensorFlow : Machine Learning Platform https://www.tensorflow.org/
6. Paraphrase Identification; Numpy; Scikit-Learn Benjamin Roth https://cla2019.github.io/paraphrases_scikit_numpy.pdf
7. GloVe: Global Vectors for Word Representation https://nlp.stanford.edu/projects/glove/

Code Available at: https://github.com/shubhamnishad97/CSI5180-assignment-3