CSI 5180: Topics In AI: Virtual Assistants

Winter 2023

Assignment 2 - Building a paraphrase classifier

Submitted To
Professor Caroline Barrière (Ph.D)

Submitted By

Shubham Kulkarni - 300276475
Snehal Sudhir Bhole - 300318105

Introduction

Problem
The task of binary classification of paraphrases involves determining whether two given sentences have the same meaning or not. Paraphrasing is the process of expressing a statement in a different way while retaining its original meaning.

For this assignment, this task has been treated as a Supervised Learning problem. Given a pair of sentences, we have been provided the scores from 5 judges, with each judge indicating whether they find a pair of sentences paraphrased or not. 3 Machine Learning Algorithms have been trained on the data provided and their results in terms of accuracy, precision and recall have been calculated.

Link Between Paraphrase Detection and Intent Detection:
The link between these two tasks is that paraphrase detection can be helpful in improving intent detection. In a question-answering system, for instance, if a user asks the same question in different ways, the system must understand that the user's intent is the same and respond accordingly, regardless of how the question is worded. The system can deliver a consistent response by determining the underlying intention behind various ways of wording the same question by looking for paraphrases.

Overall, paraphrase detection can be a useful tool for enhancing intent detection in systems that use natural language processing. The system can give users more precise and consistent responses by understanding the link between several formulations of the same statement or query.

Dataset

Parts of Dataset Used
From the dataset provided on the github link, we have used *train.data, test.data and dev.data* for working on the models. The number of rows in train, test and dev datasets are 13063, 972 and 4727 respectively. The files have been imported as csv into a dataframe. Since the data is in a tab-separated format, we have used '\t' as the separator. As per the information available on github, the dataset has been assigned following headings - 'ID', 'Topic_name', 'Sent_1', 'Sent_2', 'Label', 'Sent_1_tag', 'Sent_2_tag'. From the available headings, we have used Sent_1, Sent_2 and Label. [1]

```
df_train= pd.read_csv('~/path_to_train/train.data', sep='\t', header=None)
df_test = pd.read_csv('~/path_to_test/test.data', sep='\t', header=None)
```

Changing Graded Evaluations to Binary
In training data, the 'Label' column is in the format of (X, Y) where X indicates the number of judges that agree that the sentences are paraphrased and Y indicates vice-versa. As per instructions, if the value of X is greater than 3, the classification label is changed to 1, if the value of X is equal to 3, the entry is discarded from the dataset and if the value of X is less than 3 the label is changed to 0.
In the testing data, the 'Label' column is in a numeric format with a score between 0 and 5. Using a similar approach, if the score is more than 3, the label is set to 1, if the score is equal to 3, the row is discarded and if the score is less than 3, the label is set to 0.

This entire operation is performed by writing a function that handles all the testcases and updates the data frame. After performing pre-processing steps of removing the rows, the train dataset has 11,532 rows, test dataset has 838 rows and the dev dataset has 4142 rows.

Examples of paraphrases and non-paraphrases

Sent_1	Sent_2	Paraphrase
EJ Manuel the 1st QB to go in this draft	Can believe EJ Manuel went as the 1st QB in the draft	**Yes**
59th pick Patriots take Aaron Dobson wr Marshall	Aaron Dobson from Marshall to the Patriots	**Yes**
Love the Patriots Aaron Dobson pick	Aaron Dobson to the patriots good to see	**Yes**
EJ Manuel is the 1st QB in the draft	1st QB of the board is	**No**
EJ Manuel 1st QB taken in the NFLDraft	Is this the 1st QB pick	**No**
Love the Patriots Aaron Dobson pick	Aaron Dobson is a playmaker to say the least	**No**

Subjectivity of the class and the agreement between the 5 judges
Looking at the data, we understand that for 13,063 there are 6459 (49.45%) instances where all the judges either agree that a pair of sentences is a paraphrase or not, there are 3742 (28.65%) instances where the ratio is 4:1 or 1:4 on agreement between the judges, and finally there are 2862 (**only 21.9%**) instances where the agreement between judges is either in the ratio of 2:3 or 3:2. The instances with 2:3 agreement are eventually discarded as there is no clear indication on the paraphrases (note 3:2 agreements are accepted). The statistics suggest that for over 80% of the instances, the judges reach a majority conclusion

Also, the research paper from the authors suggests the integrated judgment achieves a F1-score of 0.823, precision of 0.752 and recall of 0.908 against expert annotations [1]. Looking at the data manually (over 50 random samples), it seems the judges are able to correctly classify the sentence pairs as paraphrases or non paraphrases for the majority of the samples. It is to be noted that the level of agreement can vary depending on the sentence pairs, ranging from moderate to high agreement. Overall, we agree with the judges.

Algorithms/Methods

Baseline Algorithm: The Baseline Algorithm uses the bag-of-words model and Support Vector Machine (SVC) Classifier.

The bag-of-words model represents each sentence pair as a bag of words (i.e., a frequency count of all the words in the document). The CountVectorizer() function in the code is used to convert the text data into a matrix of token counts, where each row represents a document and each column represents a word. The matrix is called a document-term matrix, and it is used as input to the SVC.

The SVC is trained on the document-term matrix (train_X) and the corresponding labels (df_train_updated["Label"]). Once the classifier is trained, it is used to predict the labels for the test set (df_test_updated["Label"]) and dev set (df_dev_updated["Label"]) using the document-term matrix for the test set (test_X) and (dev_X) respectively. Finally, the performance of the classifier is evaluated using four metrics: accuracy, precision, recall, and F1 score.

Example: Consider a case where we have 2 sentences, S_1 = *'EJ Manuel the 1st QB to go in this draft'* and S_2 = *'Can believe EJ Manuel went as the 1st QB in the draft.'*

The algorithm will use the CountVectorizer function to transform the text data into matrices of word frequencies. The model is then used to predict the label of new sentences, and its performance is

evaluated based on metrics such as accuracy, precision, recall, and F1-score. In the case of the given sentences, the algorithm will predict that they are similar based on shared keywords and context.

Algorithm B: This algorithm uses Roberta with cosine distance to determine similarities between two sentences.The labels are predicted using cosine similarity score.
Code Snippet below shows the Roberta model used:

```
#Using Roberta Model
model = SentenceTransformer('paraphrase-distilroberta-base-v1')
```

Roberta a transformer based language model used to get linguistic complexities from the input dev dataset. After deriving the sentence embeddings, we are finding cosine similarities in the two embeddings and predict the final label. Roberta is a pre-trained Language model, we use it to encode the df_dev_updated['Sent_1'] and df_dev_updated['Sent_2'] columns from dev dataset. After deriving their embeddings, cosine similarities for sentences is calculated to determine Final Label using threshold value of 0.50. For all the records with similarity score above 0.5 will be predicted as Paraphrase with Label 1 otherwise it will be predicted as 0.
Code Snippet below shows the method used for sentence embeddings and predicting final labels.

```
#Calculating Sentence Embeddings and their Cosine similarity Score
for index, row in df_dev_updated.iterrows():
    e1 = model.encode(row['Sent_1'])
    e2 = model.encode(row['Sent_2'])
    #Compute cosine similarity between my sentence, and each one in the corpus
    cos_sim = util.cos_sim(e1, e2)
    x = cos_sim.item()
    df_dev_updated.at[index,'Similarity'] = x
#Predicting Labels using Similarity score obtained after encoding sentences with
Roberta
threshold = 0.50
for index, row in df_dev_updated.iterrows():
    x = row['Similarity']
    if x > threshold:
        df_dev_updated.at[index,'Predicted'] = int(1)
    else:
        df_dev_updated.at[index,'Predicted'] = int(0)
df_dev_updated['Predicted'] = df_dev_updated['Predicted'].astype('int')
```

Finally, the performance of the classifier is evaluated using four metrics: accuracy, precision, recall, and F1 score on dev dataset.

Algorithm B Plus: This algorithm uses SBERT with cosine distance to determine similarities between two sentences.The labels are predicted using cosine similarity score.
Code Snippet below shows the S_BERT model used:

```
# load our Sentence Transformers model pre-trained!!
model = SentenceTransformer('all-MiniLM-L6-v2')
```

SBERT is a transformer based language model used to get linguistic complexities from the input dev dataset. After deriving the sentence embeddings, we are finding cosine similarities in the two embeddings and predict the final label.SBERT is a pre-trained Language model, we use it to encode the df_dev_updated['Sent_1'] and df_dev_updated['Sent_2'] columns from dev dataset. This approach is similar to Algorithm B used here but uses S_Bert to embed Sentences and threshold is increased to

0.70.Finally, the performance of the classifier is evaluated using four metrics: accuracy, precision, recall, and F1 score.

Consider Figure below, for the sentences "*Walk Remember definite true love*" and "*Walk Remember cutest thing*" have similarity scores of 0.6040 and 0.6321 using Roberta and SBERT embeddings. For the Algorithm B, this record will be predicted as "*Paraphrase*" but for Algorithm B plus, the final prediction will be "*Not A Paraphrase*"

	A	B	C	D	E	I	K	M
1		ID	Topic_nan	Sent_1	Sent_2	Similarity	S_BERT_Similarity	
2	0	17	A Walk To	walk rememb definit true love	walk rememb im town im upset	0.354709	0.501483	
3	1	17	A Walk To	walk rememb definit true love	walk rememb cutest thing	0.604037	0.632093	
4	2	17	A Walk To	walk rememb definit true love	walk rememb abc famili your welcom	0.544427	0.583929	
5	3	17	A Walk To	walk rememb definit true love	walk rememb amaz inspir	0.577606	0.590004	
6	4	17	A Walk To	walk rememb definit true love	guy fave part walk rememb	0.423503	0.63584	
7	5	17	A Walk To	walk rememb definit true love	day made walk rememb	0.492264	0.653404	
8	8	17	A Walk To	walk rememb definit true love	nichola spark movi genuin like walk rememb	0.350997	0.577919	

Figure: SBERT And Roberta Cosine Similarities

Result Table generated using dev dataset for all the Algorithms:

Model	**Accuracy**	**Precision**	**Recall**	**Precision**
Baseline Algorithm	0.6588	0.4458	0.0728	0.1251
Algorithm B	0.7049	0.5836	**0.5836**	**0.5841**
Algorithm B Plus	0.**7329**	**0.7131**	0.4143	0.5241

From the above table, we can see that Algorithm B Plus has comparatively better performance overall than the algorithms considered. Good precision, recall and F1 score shows that models can learn the patterns in data more effectively than others. A high recall score indicates that the model is able to capture most of the positive samples.A high precision score indicates that the model is able to accurately identify positive samples without many false positives. On comparing Algorithm B and B Plus, they have similar performances over each other but Algorithm B plus has performed considerably well with good scores in all the 4 metrics used. Thus we have analyzed its performance on our test dataset which is as follows:

Result Table generated using Test dataset for algorithm B Plus:

Accuracy	**Precision**	**Recall**	**F1-Score**
0.8484	0.7105	0.4629	0.5606

Conclusion

The results obtained clearly show that Language based models have better performance analyzing the complex linguistic relationships from the text (Considering precision and recall values). Thus for further experimentation, Linguistic NLP models can be considered as base and further approaches can be implemented on them. This dataset can't be used for intent detection effectively because there is a probability of bias in the labels generated by judges. Moreover, the accuracy is moderate for any virtual assistant to be built on these models as there are limited sentences to train models on.

References

[1] SemEval-2015 Task 1: Paraphrase and Semantic Similarity in Twitter (PIT) - Github, Research Paper
[2] Scikit-Learn - Logistic Regression - Source
[3] Kaggle - Bag-of-Words Model for Beginners - Source
[4] Models: Source
[5] Kaggle - Code Implementation of the Assignment - Source