CSI - 5180 Virtual Assistants

ASSIGNMENT - 2
Binary classification of Paraphrases

Assignment Report Submitted by

Edwin Thomas - 300278402
Saranya Krishnasami - 300321456

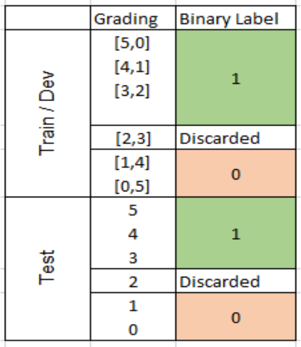
1. Introduction

A paraphrase is a restatement of a text, passage, or speech that retains the original meaning while using different words, sentence structure, or phrasing. The goal of binary classification of paraphrases is to accurately identify pairs of sentences with semantic equivalence.The difficulty in paraphrasing is the disparate nature of the text, change in tone and style and sentence structure.
To analyze a sample sentence pair from the dataset: "*Shonda Rhimes is a great writer*" and "*Shonda Rhimes needs to write a book*". The two sentences semantically convey the same meaning that the person is a great writer that she has to write a book.
Paraphrasing aids in identifying the underlying meaning of the request by restating in a way that makes it clearer and concise for better intent detection. By paraphrasing, intent detection algorithms can more accurately classify the user's intent and provide more relevant responses. For example, when the user says "*Hey Google, play some acoustic noise to sleep*" the intent identification algorithm might paraphrase it to "*Can you play me some plain white noise to sleep?* " for better intent spotting.

2. Dataset

The SEMEVAL 2015 [1] dataset is divided into train/dev/test with 13063, 4727, 972 sentence pairs respectively which are tweets. The data format is Topic_Id | Topic_Name | Sent_1 | Sent_2 | Label | Sent_1_tag | Sent_2_tag.The sentence tags are grouped into NER (BIO tagging), POS tags, Chunk Tags (BIO tagging), Event phrase (BIO tags) for every word in both the sentences. The validation(dev) dataset is used for evaluation of the baseline algorithms and test dataset for the prediction. While the train dataset is used for training the Model B.

The Label column for dev/train data is in a format like "(1, 4)", which means among 5 votes from Amazon Mechanical Turks only 1 is positive and 4 are negative. The Label column for test data is in a format of a single digit between 0 (no relation) and 5 (semantic equivalence), annotated by expert.

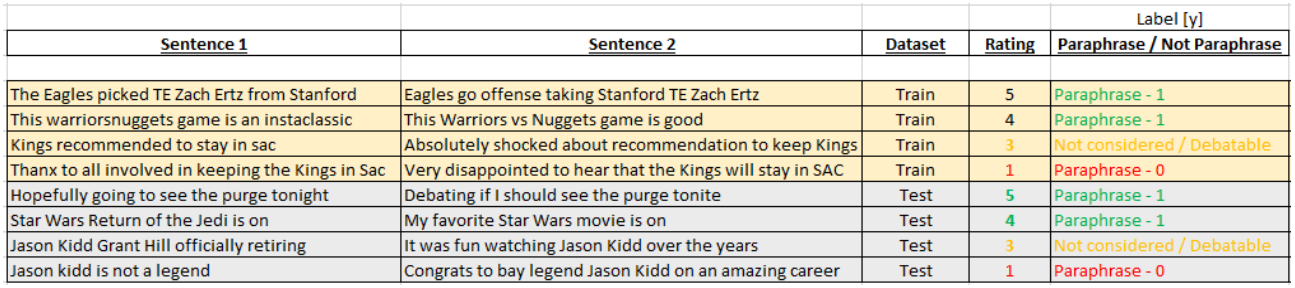


	Grading	Binary Label
Train / Dev	[5,0] [4,1] [3,2]	1
	[2,3]	Discarded
	[1,4] [0,5]	0
Test	5 4 3	1
	2	Discarded
	1 0	0

For the chosen task of binary classification, the graded evaluations by the mechanical turks (the ground truth) for train/dev the debatable ratings (2,3) are discarded and the other higher ratings such as (5,0),(4,1),(3,2) are marked as paraphrases with Label y = 1 and (1,0) as non paraphrase with Label y = 0 to mark the labels for binary classifier. For the test set, the gradings are marked as 5,4,3,2,1 with 5 denoting sentence pairs showing high semantic equivalence and 0 meaning no relation between the sentence pairs. The sentences are considered as paraphrases Label y =1 if the grading by Turks is greater than 3 , while rating 2 is discarded and if ratings are in (1,0) the sentences are considered non paraphrased.

It is observed that the train, dev and the train datasets have class imbalance. The train dataset has paraphrase class (Label y=1) 3996 and non paraphrased at 7534. While Dev set the paraphrase (Label y=1) at 1470 and non paraphrased at 2672 and test set paraphrase class at 175 and non paraphrase at 663 denoting polarity.

The details about the Amazon Mechanical turkers are not mentioned and details about their qualification are not mentioned by SEMEVAL 2015. Some examples where we differ with the gold labels are <*Just saw the box chevy video , Box Chevy video is finally out - (3, 2)*> where 3 trucks accepted it to be paraphrase and 2 marked it as negative class. While both the sentences are more likely to be the paraphrase of each other. For instance, <*MC Hammer is at the GS game, MC Hammer is sitting court side at the WarriorsAndNuggets game - (5, 0)*> all 5 turks accepted it to be paraphrase, in my opinion the tweets are ambiguous with several name entities and unless the turks are trained the labeling is not credible. Table1 shows few sorted examples for Paraphrase and non paraphrase from the dataset ,



				Label [y]
Sentence 1	**Sentence 2**	**Dataset**	**Rating**	**Paraphrase / Not Paraphrase**
The Eagles picked TE Zach Ertz from Stanford	Eagles go offense taking Stanford TE Zach Ertz	Train	5	Paraphrase - 1
This warriorsnuggets game is an instaclassic	This Warriors vs Nuggets game is good	Train	4	Paraphrase - 1
Kings recommended to stay in sac	Absolutely shocked about recommendation to keep Kings	Train	3	Not considered / Debatable
Thanx to all involved in keeping the Kings in Sac	Very disappointed to hear that the Kings will stay in SAC	Train	1	Paraphrase - 0
Hopefully going to see the purge tonight	Debating if I should see the purge tonite	Test	5	Paraphrase - 1
Star Wars Return of the Jedi is on	My favorite Star Wars movie is on	Test	4	Paraphrase - 1
Jason Kidd Grant Hill officially retiring	It was fun watching Jason Kidd over the years	Test	3	Not considered / Debatable
Jason kidd is not a legend	Congrats to bay legend Jason Kidd on an amazing career	Test	1	Paraphrase - 0

Table 1 : Sample paraphrases and non paraphrases from training and test set

3. Algorithms / Methods

3.1 Baseline Algorithm - WER

The baseline algorithm used is plain sentence comparison by calculating the WER between the sentences using the python libraries *[2]* and then classifying them based on the threshold. The steps followed are,

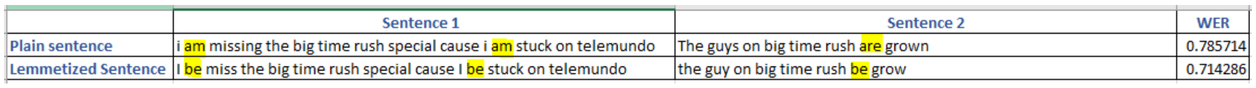
1) The sentences are compared using the WER evaluation with the longer sentence as the ground truth and the other sentence as the sentence to compare. The distance used in the calculation of WER is the Levenshtein (edit) distance *[3]*
2) Since this is error distance the sentence pairs with WER less than threshold (t=0.5) are considered as paraphrases and the sentence pairs with WER more than the threshold are classified as non paraphrases
    a) WER > t , where t = 0.5 implies Not Paraphrase
    b) WER < t , where t = 0.5 implies Paraphrase
3) The ground truth is calculated for the dev set as below and the sentences are binary classified,
    a) Ranking [5,0] [4,1] [3,2] - Considered Paraphrase(1) , Ranking [2,3] - Discarded , Ranking [0,5] - Non Paraphrase (0)
4) Once the label (y) is identified for the dev set the evaluation metric is calculated comparing the prediction against the ground truth

Example : For example as shown in the figure below, the two sentences are compared for the number of substitutions, insertions and deletions and the WER is calculated. With heuristic threshold set to 0.5, the WER (WER > t , where t = 0.5 - Not paraphrase) is compared against the threshold and the sentences are binary classified as paraphrased or not.

Ground Truth - 'A walk to remember is the definition of true love'
Sentence to Compare - 'A walk to remember is the cutest thing'
Word Error Rate (WER) - (Substitution + insertion + deletion) / Number of words spoken
WER = 5 + 0 + 0 / 10 = **0.5**

3.2 Baseline Algorithm on lemmatized sentences with WER

To extend the base algorithm the sentence pairs are lemmatized before calculating the WER following all the previous steps to perform the binary classification. Since lemmatization aims at giving the base or dictionary form of the word, the lemmatized WER approach shows a slight improvement in the reduction of WER as in Table 2 example.



	Sentence 1	Sentence 2	WER
Plain sentence	i am missing the big time rush special cause i am stuck on telemundo	The guys on big time rush are grown	0.785714
Lemmetized Sentence	I be miss the big time rush special cause I be stuck on telemundo	the guy on big time rush be grow	0.714286

Table 2 : Sample plain and lemmatized sentence pairs from training set

3.3 Baseline supervised Binary classification with Logistic Regression , SVC and KNN classifier

To perform binary classification in supervised setup, the precision, recall and f1 scores are calculated for the matching of unigram, bigram of between plain sentence pairs and unigram, bigram for lemmatized sentences pairs. The 9 features are fed into the machine learning models for binary classification. The models are fit using the training data and the evaluated on the validation set.

3.4 Transformer-based Language Models

In this section we present two approaches based on masked language models to solve the PI task.

3.4.1 Pre-trained Sentence Transformers

The first approach uses pre-trained MiniLM model [4] to obtain sentence embeddings. The model achieves comparable performances to BERT on SOTA benchmarks while using only a fraction of the parameters (~30M as opposed to ~109M). This is achieved by employing knowledge distillation (teacher student training methodology) to transfer self-attention behavior from large LMs such as BERT to a smaller parameter space [5]. It consists of 12 layers with a hidden dimension size of 384. We first pass the sentence pairs to this model followed by computing sentence embedding similarities using the Cosine Similarity measure [6]. A threshold of 0.45 is used to classify text into paraphrases and non-paraphrases.

3.4.2 Fine-tuned Sentence Transformers

As the pre-trained embeddings did not achieve the expected performance on the tweet type dataset, the model is further fine tuned using the siamese networks for sentence-pair regression tasks [7], using the sentence transformers library [8].

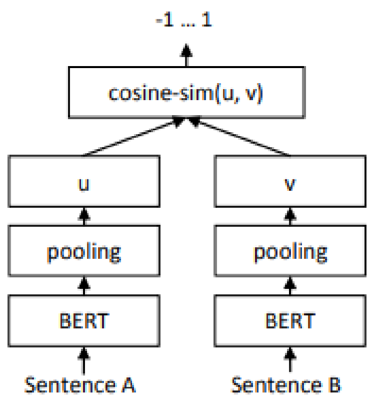


Fig 1. SBERT using cosine similarity loss

Fig. 1 shows the training setup, where the BERT networks have tied weights. The sentences are passed to these networks, followed by a mean-pooling layer which reduces *nx384* output embedding dimensions to *1x384* (n is the number of input tokens + the special tokens (CLS, SEP, PAD for a given sentence). This is followed by a normalization layer and a cosine similarity loss function [9] that computes the semantic distance between the two sentence embeddings to compute the loss for updating the model weights as shown below (Eq. 1):

||input_label - cosine_sim(u,v)||_2 — (1)

To train the model, we use the mini-batch gradient descent scheme (batch size = 16), and the Adam Optimizer with a learning rate of 2e-05. To ensure a smooth learning process, we employ a non-linear learning rate scheduler using 10% of the training data for performing warm-up cycles. 13063 sentence

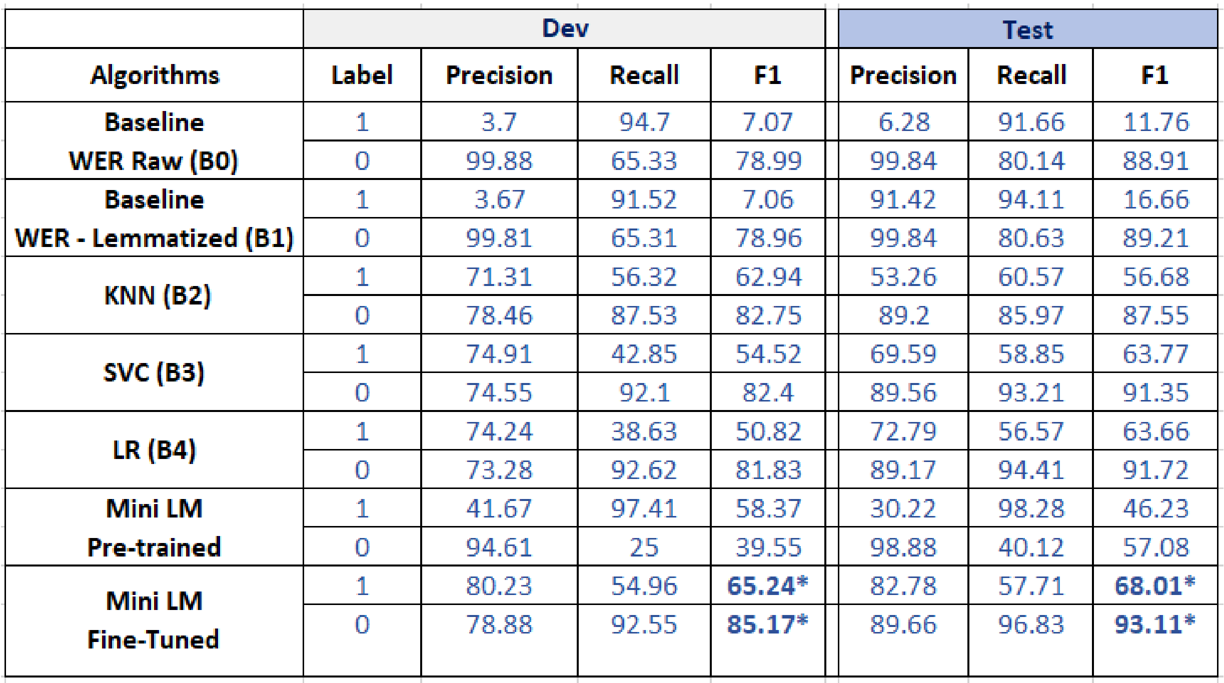
pairs are used for training with 4727 evaluation pairs. The Model is evaluated after every 100 steps in the training cycle and the best checkpoint is saved based on its performance on the validation set. Inference is performed in the same way as described in 3.4.1.
For eg: For the sentence pair <"God forbid lyknx Rafa Benitez", "How can chelsea fans still hate benitez">, the model outputs a score of 0.10, which is mapped to 0 (non-paraphrase) using a threshold of 0.45.

4. Results

4.1 Evaluation results on Validation set

Acknowledging the class imbalance in the datasets the metric used to evaluate the classification are Precision, Recall and F1.Table 3 shows the performance of baseline, supervised models KNN, SVC, LR and MiniLM (BERT-based) models on validation set.

From table 3 the baseline model (B1 and B2) did not show much improvement in scores except a slight improvement in recall for the lemmatized method. Baseline models are having good recall rates indicating the model classifying all positive samples correctly as positive also with a tradeoff of precision of 0.04 denoting a high number of false positives. While for the negative class the recall rates are at 0.65 in classifying the negative class here the non paraphrased text and a good precision of 0.99 in accurately identifying the negative class.
Whilst for the supervised models KNN, SVC and LR for positive class it is noted that the recall is at 0.56, 0.43 and 0.38 denoting the accuracy of model to predict the positive class and precision is at 0.70 to 0.74 meaning the 74% times with less false positives.



	Dev				Test		
Algorithms	**Label**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Baseline WER Raw (B0)	1	3.7	94.7	7.07	6.28	91.66	11.76
	0	99.88	65.33	78.99	99.84	80.14	88.91
Baseline WER - Lemmatized (B1)	1	3.67	91.52	7.06	91.42	94.11	16.66
	0	99.81	65.31	78.96	99.84	80.63	89.21
KNN (B2)	1	71.31	56.32	62.94	53.26	60.57	56.68
	0	78.46	87.53	82.75	89.2	85.97	87.55
SVC (B3)	1	74.91	42.85	54.52	69.59	58.85	63.77
	0	74.55	92.1	82.4	89.56	93.21	91.35
LR (B4)	1	74.24	38.63	50.82	72.79	56.57	63.66
	0	73.28	92.62	81.83	89.17	94.41	91.72
Mini LM Pre-trained	1	41.67	97.41	58.37	30.22	98.28	46.23
	0	94.61	25	39.55	98.88	40.12	57.08
Mini LM Fine-Tuned	1	80.23	54.96	**65.24***	82.78	57.71	**68.01***
	0	78.88	92.55	**85.17***	89.66	96.83	**93.11***

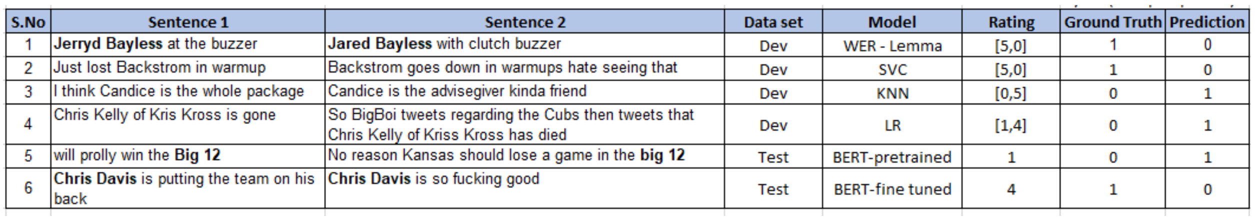
Table 3 : **Baseline, KNN, SVC, LR and Fine tuned MiniLM evaluation results on validation set**

4.2 Results on test set

The fine-tuned transformer model obtains the best performance in the test dataset achieving an f1-score of ~68% and ~93% on the two classes respectively. It can be seen that a significant improvement from the pre-trained version is observed after fine-tuning the weights to learn tweet text type characteristics, vocabulary range, semantics and context.

4.3 Error analysis

The qualitative analysis performed on dev set for baseline WER model as shown in Table 4 reveals that the spelling errors in sentence #1 like 'Jarryd' and 'Jared' impacted the similarity check as the WER looks at the exact word match.
For the supervised models the features are n-grams, the model identified n-gram matches in #4 like 'chris Kelly', 'kelly of' , 'kris kross' in both sentences and misclassified the pairs as paraphrased however this is a false positive case.



S.No	Sentence 1	Sentence 2	Data set	Model	Rating	Ground Truth	Prediction
1	**Jerryd Bayless** at the buzzer	**Jared Bayless** with clutch buzzer	Dev	WER - Lemma	[5,0]	1	0
2	Just lost Backstrom in warmup	Backstrom goes down in warmups hate seeing that	Dev	SVC	[5,0]	1	0
3	I think Candice is the whole package	Candice is the advisegiver kinda friend	Dev	KNN	[0,5]	0	1
4	Chris Kelly of Kris Kross is gone	So BigBoi tweets regarding the Cubs then tweets that Chris Kelly of Kriss Kross has died	Dev	LR	[1,4]	0	1
5	will prolly win the **Big 12**	No reason Kansas should lose a game in the **big 12**	Test	BERT-pretrained	1	0	1
6	**Chris Davis** is putting the team on his back	**Chris Davis** is so fucking good	Test	BERT-fine tuned	4	1	0

Table 4 : Quantitative analysis of models on dataset

It can be observed that the pre-trained MiniLM model associates a high similarity between sentences talking about the same entity, without giving weightage to the context of intent (for eg: due to the entity “Big 12” the model classifies the text as a paraphrase in #5). This high recall and low precision nature of the model can be attributed to the fact that the sentences contain slang words and have a non-grammatical structure specific to tweets. The fine-tuned model is able to overcome this challenge to some extent by learning patterns from the training data. However, it fails to recognize some of the paraphrases, which use completely different words to describe the same idea (as seen in #6 obscene words are used as adjectives with very little background context).

5. Conclusion

To summarize, the problem of binary classification of sentence pairs is done using baseline, supervised and deep learning models. The baseline models that used thresholding on WER did not seem very effective as sentences can be paraphrased using completely different sets of words, although lemmatization (or reducing words to their base forms) seemed to give slight improvements. Supervised models using n-gram features obtained much better results than baseline. Transformer based language models give the best results on the dataset after fine-tuning, as it learns to associate tweet specific vocabulary to short contexts more effectively. To further improve on the results, we can fine-tune the Transformer model using additional information such as their POS tags, Chunk tags and NER tags.

The given dataset is collection of tweets with limited context, extensive presence of informal language and lack of diversity in content. Due to these shortcomings this dataset is not very suitable for intent detection tasks.

6. References

[1] SEMEVAL dataset: Data set Link
[2] WER calculation: https://pypi.org/project/jiwer/
[3] Level : https://github.com/maxbachmann/Levenshtein
[4] https://huggingface.co/microsoft/MiniLM-L12-H384-uncased
[5] https://arxiv.org/pdf/2002.10957.pdf
[6] https://www.sciencedirect.com/topics/computer-science/cosine-similarity
[7] https://arxiv.org/pdf/1908.10084.pdf
[8] https://www.sbert.net/
[9] https://www.sbert.net/docs/package_reference/losses.html#cosinesimilarityloss
[10] https://github.com/edwinthomas444/paraphrase-identification.git