Attempts to solve 2 GLUE-benchmark tasks based on Statistical rules and word embeddings

Xiaoyuan Liu
lxy9843@sjtu.edu.cn

Abstract

The General Language Understanding Evaluation (GLUE) benchmark[1] is a set of tasks which useful to test the ability of a program to classify or rate sentence or sentence-pair. In this course work, we pick 2 of its 11 task, CoLA and STS-B to evaluate the performance of some existing methods. We first try some naïve statistical rule-based approaches to get familiar with the dataset and learn about the structure of the tasks. Then state-of-the-art methods utilizing pretrained word embeddings like BERT are implemented.

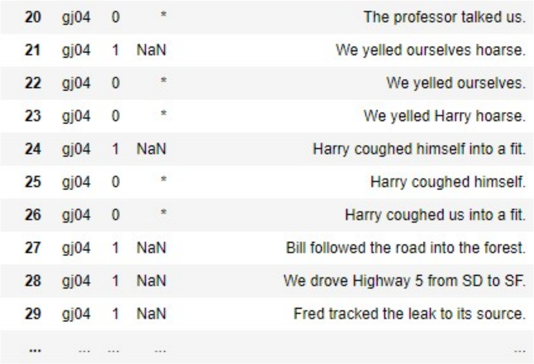
It is important to note that ***this work is more like a report rather than a paper because it is part of the course work****. As a result, the paragraphs below will focus more on the implementation details and the result of the experiments instead of innovative ideas.*

1. Introduction

The Corpus of Linguistic Acceptability (CoLA)[2] is a set of labeled English sentences purposed to evaluate the ability of machine learning methods in judging grammatical acceptability (see 错误!未找到引用源。). It is a typical binary classification problem for text sequence.

The Semantic Textual Similarity Benchmark[3] is a set of English sentence-pairs rated between 0 to 5 dues to the semantic similarity between two sentences. The source of the sentences is given and the program being tested are expected to give relatively coherent rating to the standard rating judged by Pearson correlation coefficient (see Table 2). It can be treated as a regression problem for its result is in a continues space.

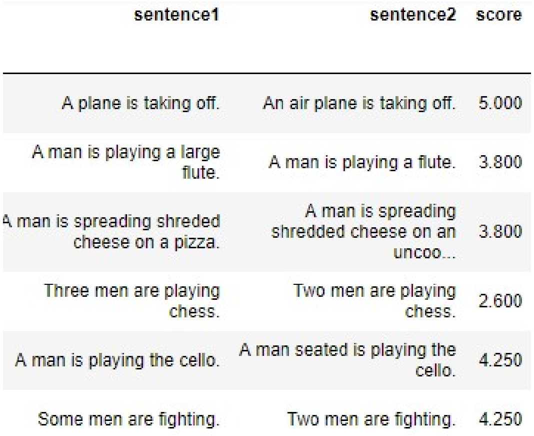
The 2 chosen problems from GLUE benchmark are all about capturing some properties of one or two short sequence of English texts. One focus on lexical level and the other focus on semantic level. To solve these two tasks using computer programming techniques, there are two obvious ways. One is to observe the feature of the task and construct human understandable rules based on statistics to classify or rate the examples. Another one is to utilize language models learned from large unsupervised corpus and build neural networks to complete all tasks.



20	gj04	0	*	The professor talked us.
21	gj04	1	NaN	We yelled ourselves hoarse.
22	gj04	0	*	We yelled ourselves.
23	gj04	0	*	We yelled Harry hoarse.
24	gj04	1	NaN	Harry coughed himself into a fit.
25	gj04	0	*	Harry coughed himself.
26	gj04	0	*	Harry coughed us into a fit.
27	gj04	1	NaN	Bill followed the road into the forest.
28	gj04	1	NaN	We drove Highway 5 from SD to SF.
29	gj04	1	NaN	Fred tracked the leak to its source.
...	...	...	...	...

TABLE 1

Some examples from CoLA *train.tsv*. All grammatical unacceptable sentences are labeled 0 and others are labeled one in the third column.



sentence1	sentence2	score
A plane is taking off.	An air plane is taking off.	5.000
A man is playing a large flute.	A man is playing a flute.	3.800
A man is spreading shreded cheese on a pizza.	A man is spreading shredded cheese on an uncoo...	3.800
Three men are playing chess.	Two men are playing chess.	2.600
A man is playing the cello.	A man seated is playing the cello.	4.250
Some men are fighting.	Two men are fighting.	4.250

TABLE 2

Some example from STS-B *train.tsv*. Sentences with tiny differences are rated 5 and unrelated sentences are rated close to zero.

[1] Wang A, Singh A, Michael J, et al. Glue: A multi-task benchmark and analysis platform for natural language understanding[J]. arXiv preprint arXiv:1804.07461, 2018.

[2] Warstadt A, Singh A, Bowman S R. Neural network acceptability judgments[J]. arXiv preprint arXiv:1805.12471, 2018.

[3] Cer D, Diab M, Agirre E, et al. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation[J]. arXiv preprint arXiv:1708.00055, 2017.

2. Dataset

In this section, we will dive into details about two datasets and their evaluation metrics. Also, observations are given so that it may serve as the motivation of our methods improve the test performance.

2.1 CoLA (Table 3)

#Train	8551
#Dev	1043
#Test	1063
Evaluation	Matthews correlation coefficient

TABLE 3

■ Observation 1

Due to the construction method of the dataset, grammatically unacceptable sentence may be generated by changing a part of a correct sentence. The modification includes insertion, deletion, and replacement.

Example.

gj04	1		We yelled ourselves hoarse.
gj04	0	*	We yelled ourselves.
gj04	0	*	We yelled Harry hoarse.

■ Observation 2

The modification mentioned in observation 1 does not necessarily make the sentence unacceptable. There are cases that the modification happened but the sentences still make sense.

Example.

gj04	1		The building is tall and wide.
gj04	0	*	The building is tall and tall.
gj04	1		This building is taller and wider than that one.
gj04	1		This building got taller and wider than that one.
gj04	1		This building got taller and taller.

■ Matthews correlation coefficient

It is calculated by the following formula

$$MCC = \frac{TP * TN \; - \; FP * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}},$$

where

TP: True Positive	FP: False Positive
TN: True Negative	FN: False Negative.

It worth mentioning that this metric is “stricter” and more reliable than accuracy in binary classification problem, especially when the output result is not balanced.

2.2 STS-B (Table 4)

#Train	5749
#Dev	1500
#Test	1379
Evaluation	Pearson correlation coefficient

TABLE 4

■ Observation 1

The similarity rating is subjective, the rating may vary even for the same case.

Example.

A man swims underwater. A woman is swimming underwater.	2.000
A woman is slicing an onion. A man is cutting and onion.	3.200

■ Observation 2

Generally, two sentences with one losing some tiny details result in a high score.

Example.

A man is playing the guitar. A man plays an acoustic guitar.	3.750
A woman picks up and holds a baby kangaroo. A woman picks up and holds a baby kangaroo in her arms.	4.600

■ Pearson correlation coefficient

It is calculated by the following formula

$$PCC = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})}},$$

where X and Y are predictions and ground truth. It is also called linear correlation coefficient because its result won’t change if one of X or Y is linearly transformed with a positive scale.

3. Statistical methods

To better understand how the result of the dataset are given, we implement several naïve statistical methods and test their performance so that we will have a brief sense of how hard the tasks are.

3.1 CoLA – Blacklist approach

A naïve idea is that if part of a sentence is grammatically unacceptable, the whole sentence should be unacceptable as well. Following this intuition, the blacklist approach is introduced. The blacklist in this approach is a sequence of token that exists in the negative example but never in the positive example. By sequentially scanning the whole training set, we can generate such a list and use it to judge the example in the developing set as follow. If any token list in the developing example is in the blacklist, then we believe it to be a grammatically unacceptable, otherwise treat it as acceptable.

In this method, a very important decision is the choice of length of the target sequence, in other words the n in the n-gram if we describe it as the bag-of-word model.

However, a potential problem is that we cannot track all n-gram that violate the grammar rules. With a relatively small dataset, we cannot produce a classifier with good performance. An alternative may be using the part of speech as the sentence template and rule out those invalid part of speech sequence.

According to these principles, we wrote code to test how this approach work out.

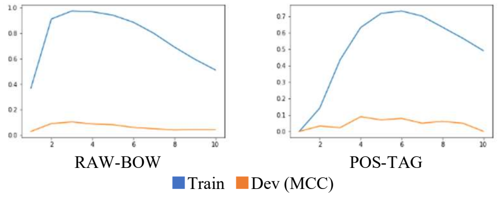
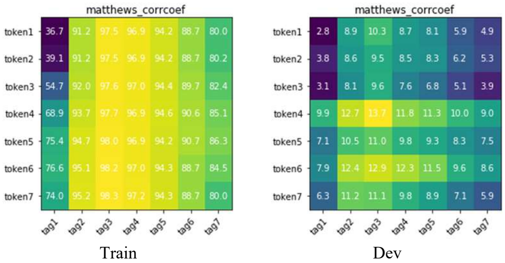


FIGURE 1

From Figure 1, we see that our purposed method have a reasonable performance in the training set but have poor generalization ability. We can also observe that the POS-TAG method works even worse than the RAW-BOW method. One possible explanation is that, in our code, we use a pretrained POS tagging model provided in *nltk*, but this model is not robust to sentence which are grammatically unacceptable, thus producing less accurate tagging result.

Finally, we test the performance combining two blacklists produced by RAW-BOW and POS-TAG (Figure 2). With proper hyper-parameter, our system achieve result in Table 5.



token-n: the length n of n-gram
tag-m: the length m of m-gram of pos tag.

FIGURE 2

Approach	MCC-DEV
blacklist	13.7

TABLE 5

3.2 STS-B – Based on co-existence

For the STS-benchmark, we test a relatively more intuitive method: counting the co-existence works in two sentences.

$$\text{similarity}: 5 * \frac{\frac{\#token\ in\ A\ also\ in\ B}{\#token\ in\ A} + \frac{\#token\ in\ B\ also\ in\ A}{\#token\ in\ B}}{2}$$

The result is shown in Table 6.

Approach	PCC-DEV
Based on co-existence	62.1

TABLE 6

4. Pretrained word embeddings

The third section has shown that the performance of some naïve statistical methods is unsatisfactory. Recently, neural networks are widely used to extract the features from examples for classification and regression problems. For language understanding, recent works like ELMo[4] and BERT[5] use large unsupervised corpus to train its contextualized word embeddings. These kinds of methods improved the performance of the language model on various tasks significantly.

In this paper, we implement the state-of-the-art model BERT on two chosen tasks and submit the result to GLUE benchmark website[6] to see test result.

We utilize the result in Pytorch-Pretrained-BERT[7] and fine tune to adapt it to our task. After careful selection of hyper-parameters, the result without ensemble is shown in Table 7.

CoLA	**MCC-TEST**
BERT-large-ft	62.3
STS-B	**PCC-TEST**
BERT-large-ft	87.8/86.7

TABLE 7

5. Conclusion

In this paper, both traditional statistical methods and state-of-the-art methods utilizing BERT with their augmented variants are implemented and tested. The result shows that for small datasets which need the program to understand the structure and the meaning of the language, traditional statistical methods lacks the information of the language itself thus cannot provide good results. On the other hand, pretrained language model is indeed a reasonable and powerful choice.

It worth mentioning that as the researches in word embeddings continue, more and more handy and powerful language models will be purposed in the future. Four days before this paper finished, a new pretrained model called XLNet[8] proved that it has better performance than BERT after ensemble in almost all datasets in GLUE benchmark.

[4] Peters M E, Neumann M, Iyyer M, et al. Deep contextualized word representations[J]. arXiv preprint arXiv:1802.05365, 2018.

[5] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

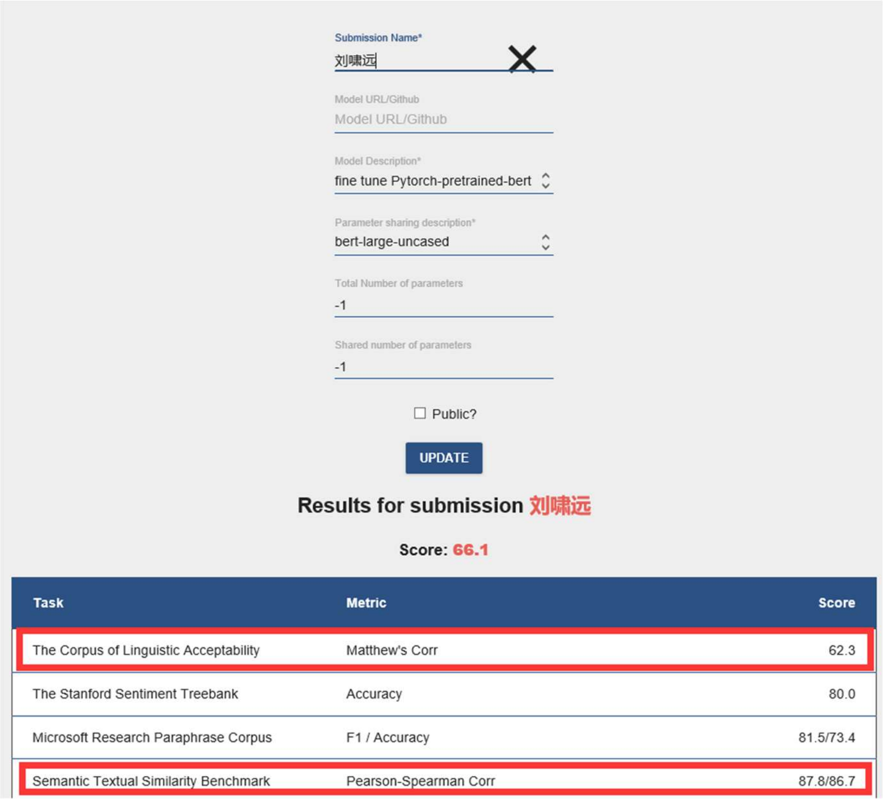
[6] https://gluebenchmark.com/

[7] https://github.com/huggingface/pytorch-pretrained-BERT

[8] https://github.com/zihangdai/xlnet

6. Appendix – Screenshot

Final result



Submission Name*

刘啸远

Model URL/Github

Model URL/Github

Model Description*

fine tune Pytorch-pretrained-bert

Parameter sharing description*

bert-large-uncased

Total Number of parameters

-1

Shared number of parameters

-1

☐ Public?

Results for submission 刘啸远

Score: 66.1

Task	Metric	Score
The Corpus of Linguistic Acceptability	Matthew's Corr	62.3
The Stanford Sentiment Treebank	Accuracy	80.0
Microsoft Research Paraphrase Corpus	F1 / Accuracy	81.5/73.4
Semantic Textual Similarity Benchmark	Pearson-Spearman Corr	87.8/86.7

Script used

```
# CoLA
python run_classifier.py \
--task_name CoLA \
--data_dir ./data/glue_data_all/CoLA \
--output_dir ./output/CoLA6/ \
--max_seq_length 64 \
--train_batch_size 24 \
--learning_rate 1e-5 \
--num_train_epochs 10.0 \
--bert_model bert-large-uncased \
--do_train --do_eval --do_test \
--do_lower_case

# STS-B
python run_classifier.py \
--task_name sts-b \
--data_dir ./data/glue_data_all/STS-B \
--output_dir ./output/STSB001/ \
--max_seq_length 64 \
--train_batch_size 24 \
--learning_rate 2e-5 \
--num_train_epochs 10.0 \
--bert_model bert-large-uncased \
--seed 1234 \
--do_train --do_eval --do_test \
--do_lower_case
```

All submission scripts can be found in submits.ipynb

All submitted labels can be found in data/my_submission

All trained model can be found in output