Sarcasm detector using machine learning algorithms

By Project group 15

Motivation

Sarcasm is a sophisticated form of irony that is commonly found in social networks. It highly disrupts computational systems that harness this data to perform tasks such as sentiment analysis, opinion mining, author profiling, and harassment detection.

Sarcasm detection is an essential tool in the text classification for dealing with this sophisticated emergent. The performance of sarcasm classifier is important for accurate predictions when processing expressions in the textual data.

Our goal is to develop a new model to implement the sarcasm detection with good performance.

Dataset

	tweet	sarcastic	rephrase	sarcasm
0	The only thing I got from college is a caffein...	1	College is really difficult, expensive, tiring...	0.0
1	I love it when professors draw a big question ...	1	I do not like when professors don’t write out ...	1.0
2	Remember the hundred emails from companies whe...	1	I, at the bare minimum, wish companies actuall...	0.0
3465	I’m finally surfacing after a holiday to Scotl...	0	NaN	NaN
3466	Couldn't be prouder today. Well done to every ...	0	NaN	NaN
3467	Overheard as my 13 year old games with a frien...	0	NaN	NaN

Sarcasm labels are provided by the authors themselves.

Authors also rephrased the sarcastic text to convey the same intended message without using sarcasm.

Task

The main goal of this project is to determine the sarcasm in a tweet.

The task is to build up a model to determine sarcasm given a sarcastic text and its non-sarcastic rephrase while two texts convey the same meaning

	tweet	sarcastic	rephrase	sarcasm
0	The only thing I got from college is a caffein...	1	College is really difficult, expensive, tiring...	0.0
1	I love it when professors draw a big question ...	1	I do not like when professors don't write out ...	1.0
2	Remember the hundred emails from companies whe...	1	I, at the bare minimum, wish companies actuall...	0.0

Baseline

Several common mature methods on binary classification are listed as follows:

1. LogisticRegression
2. GaussianNB
3. KNeighborsClassifier
4. SVC
5. DecisionTreeClassifier
6. Random

Experiment and Result

1. Data Augmentation

```
paired = data[data['rephrase'].notnull()][['tweet', 'rephrase']]

part1 = pd.DataFrame(paired.head(paired.shape[0] // 2))
part1['sarcastic'] = 0

part2 = pd.DataFrame(paired.tail(paired.shape[0] - paired.shape[0] // 2))
part2.rename({'tweet': 'rephrase', 'rephrase': 'tweet'},
             inplace=True, axis='columns')
part2['sarcastic'] = 1

paired_shuffled = shuffle(pd.concat([part1, part2]), random_state=seed)
paired_shuffled
```

	tweet	rephrase	sarcastic
361	All the shade i have been hearing about Ben Platt being unbelievable as a teenager in the @DearEvanHansen movie boggles me. Does nobody have any memories of Grease??? Name 1 actor on that film who believably looked high school age!	Dear Evan Hanson cast the same type of older actors as Grease.	0
422	you would think the odds of me sitting next to the exact same stinky man three bus trips in a row is highly unlikely. alas I seem to be a lucky gal	I can't believe the same man ended up sitting next to me on the bus three times in a row. He stinks, so I'm unlucky.	0
67	Billy Gunn sorta relevant for the first time in years...	Billy Gunn is actually in the spotlight now.	0
779	Donald Trump asked the Chief Medical Officer if people should drink bleach to treat covid 19 and now zoflora, dettol and toilet duck are trending on twitter.	I see Dettol, Toilet Duck and Zoflora are all trending on Twitter. Now why could that be... #TrumpPressBriefing	1

Experiment and Result

2. Data Preprocess

```
class DataPreprocess:
      ...
      def __call__(self, tweet):
         tweet = self.replace_emojis(tweet)
         tweet = self.replace_hashtags(tweet)
         tweet = self.replace_link(tweet)
         tweet = self.replace_mentions(tweet)
         return tweet

preprocess = DataPreprocess()
```

```
preprocess("Hi, we are @team15 😄. My personal page "
           "is http://6301.com/team15/. 6301 is"
           "nice.")
```

```
Output:
Hi, we are @user . My personal page is #link. 6301 is nice.
```

Experiment and Result

3. Baseline Accuracy before/after Preprocessing

```
def task_baseline(data_df, X_train, X_col1_test, X_col2_test,
                  y_train, y_test):
    # training
    classifiers = [LogisticRegression(solver='lbfgs', ...),
                   GaussianNB(),
                   KNeighborsClassifier(),
                   SVC(gamma='scale', probability=True,...),
                   DecisionTreeClassifier(random_state=seed)]
    names, accuracies = [], []

    for clf in tqdm(classifiers):
        ...

    # random
    ...

    res_df = pd.DataFrame({'classifier': names,
                           'accuracy': accuracies})
    display(res_df.style.highlight_max(['accuracy']))
```

	Classifier	Non-preprocess Acc	Preprocess Acc
1	LogisticRegression	**0.730**	0.756
2	Gaussian	0.443	0.431
3	KNeighborsClassifier	0.092	0.132
4	SVC	0.701	**0.770**
5	DecisionTree	0.402	0.460
6	Random	0.500	0.501

Experiment and Result

4.1 Bert

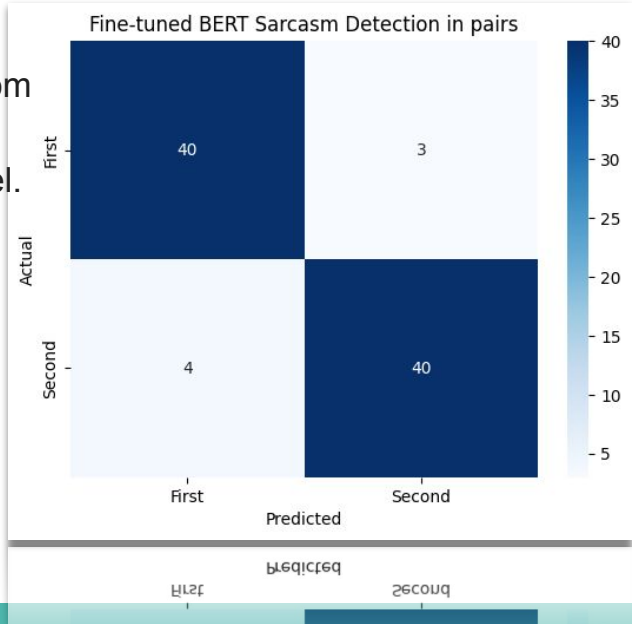
```
bert_model = BertForSequenceClassification \
      .from_pretrained('bert-base-cased') \
      .to(device)
bert_trainer = Trainer(...) # 8 epochs

bert_trainer.train()
bert_trainer.evaluate()
bert_result = bert_trainer.evaluate(
      eval_dataset=test_bert,
      metric_key_prefix='test')


ax = sns.heatmap(cf_matrix, annot=True,
      cmap='Blues', fmt='d')
```

BERT can consider the full context of a word under long context.

We used BERT base cased model for the sequence classification and fine-tuned with preprocessing and BertTokenizer.

The Transformers library from Hugging Face was used to train and evaluate the model.

Sample Accuracy ~= 93%



Experiment and Result

4.2 GPT2

```
gpt_model =
GPT2ForSequenceClassification.from_pretrained(
      'remotejob/tweetsDISTILGPT2fi_v4', num_labels=2)
gpt_model.config.pad_token_id =
gpt_model.config.eos_token_id
gpt_model = gpt_model.to(device)

gpt_trainer = Trainer(
   model=gpt_model,
   args=training_args,
   train_dataset=train_gpt,
   eval_dataset=val_gpt,
   compute_metrics = compute_metrics
)

gpt_trainer.train()
gpt_trainer.evaluate()
```

GPT-2 has 1.5 billion parameters, used to be one of the largest and most powerful language models.

We used the GPT-2 model for sequence classification and fine-tuned on the sarcasm detection task.

The library from hugging face named "tweetsDISTILGPT2fi_v4" and the corresponding tokenizer were used to train and evaluate the performance of model on the datasets.

Experiment and Result

4.3 Logistic Regression / SVM with BertTokenizer

```
logreg = LogisticRegression(solver='lbfgs',
random_state=seed)
logreg.fit(X_train, y_train)
svc = SVC(gamma='scale', probability=True,
random_state=seed)
svc.fit(X_train, y_train)
clf_evaluate(logreg, X_col1_val, X_col2_val, y_val)
svc_result = clf_evaluate(svc, X_col1_test, X_col2_test,
y_test)
```

```
{'test_accuracy': 0.9080459770114943,

 'test_f1': 0.9111111111111111,

 'test_precision': 0.9318181818181818,

 'test_recall': 0.8913043478260869}
```

1. Compared to TF-IDF, the BERT tokenizer is better at capturing the complex semantic information of words.

2. The BERT model has been pre-trained on a large dataset, the tokenizer designed for it has better generalization capabilities as well.

Overall Result

Our best-performing model is BERT, which achieves an accuracy of 0.9195.

The fine-tuned GPT-2 model achieved the lowest performance than any other BERT related method, which is still competitive in baseline with and without preprocessing.

The logistic regression model achieved an accuracy of about 0.8966.

The support vector machine (SVM) model also achieved an accuracy of 0.8966.

When applying the BERT tokenizer to the SVM and logistic regression models, we observed an improvement in accuracy, which suggests that the use of BERT em -beddings helped to result in better performance.

For comparison: The raw performance of GPT-3.5-Turbo among valid response is 86%

	test_accuracy	test_f1	test_precision	test_recall
BERT	0.919540	0.919540	0.930233	0.909091
GPT2	0.632184	0.515152	0.772727	0.386364
Logistic Regression	0.896552	0.888889	0.972973	0.818182
SVC	0.896552	0.891566	0.948718	0.840909

Conclusion

From the results, it is clear that fine-tuned BERT and the logistic regression model using embeddings from BERT performed the best in detecting sarcasm in paired tweets. This indicates that contextualized word embeddings and fine-tuning on task-specific data are effective in capturing the nuances of sarcasm in paired tweets.

Overall, the results suggest that fine-tuning on task-specific data and using contextualized word embeddings are effective approaches for detecting sarcasm in paired tweets.

	Classifier	Non-preprocess	Preprocess
1	Logistic Regression	**0.730**	0.756
2	Gaussian	0.443	0.431
3	KNeighbors Classifier	0.092	0.132
4	SVC	0.701	**0.770**
5	DecisionTree	0.402	0.460
6	Random	0.500	0.501

	test_accuracy	test_f1	test_precision	test_recall
BERT	0.919540	0.919540	0.930233	0.909091
GPT2	0.632184	0.515152	0.772727	0.386364
Logistic Regression	0.896552	0.888889	0.972973	0.818182
SVC	0.896552	0.891566	0.948718	0.840909

References

- https://huggingface.co/bert-base-cased
- Computing metrix github https://github.com/huggingface/notebooks/blob/6b19898fe599e1b2bd401fc260ee5a45824f1420/sagemaker/06_sagemaker_metrics/scripts/train.py#L51
- Bert Dataset github https://github.com/aws-samples/aws-ai-ml-workshop-kr/blob/147e49d71b6fe093a6190b748bfbae44418d2e0a/sagemaker/sm-kornlp/question-answering/scripts/train.py#L53