Text Classification Competition: Twitter Sarcasm Detection

Artsiom Strok
astrok2@illinois.edu

Peter Zukerman
peterz2@illinois.edu

Dheeraj Patta
npatta2@illinois.edu

***Abstract*—We are exploring different machine learning models and NLP techniques to classify tweets as sarcastic or not. We tried classical algorithms such as Naive Bayes, Gradient Boosting, SVM, Random Forest, and deep learning algorithms such as LSTM, CNN, using various word embeddings. And finally, we used the BERT-large model to achieve the best results. We also explored different text pre-processing techniques, experimented with the context of the tweets, and tried to use additional information such as the tweet's sentiment.**

***Index Terms*—Sarcasm, NLP, Figurative Language, Deep Learning, BERT, CNN, LSTM, Word2Vec.**

I. Introduction

Due to the growing volume of available textual information, there is a great demand for Natural Language Processing (NLP) techniques that can automatically process and manage texts, supporting the information retrieval and communication in core areas of society (e.g. healthcare, business, and science). NLP techniques have to tackle the often ambiguous and linguistic structures that people use in everyday speech. As such, there are many issues that have to be considered, for instance slang, grammatical errors, regional dialects, figurative language, etc. Figurative Language (FL), such as irony, sarcasm, similes, and metaphors, pose a serious challenge to NLP systems. FL is a frequent phenomenon within human communication, occurring both in spoken and written discourse including books, websites, forums, chats, social network posts, news articles and product reviews. [**?**]

II. Data

A. Dataset format

Each line contains a JSON object with the following fields:

- ***response*** : the Tweet to be classified
- ***context*** : the conversation context of the
- ***response*** - Note, the context is an ordered list of dialogue, i.e., if the context contains three elements, `c1`, `c2`, `c3`, in that order, then `c2` is a reply to `c1` and `c3` is a reply to `c2`. Further, the Tweet to be classified is a reply to `c3`.
- ***label*** : `SARCASM` or `NOT_SARCASM`

B. Example

```
"label": "SARCASM",
"response":
    "@USER @USER @USER I don't
    get this .. obviously you do care
    or you would've moved right along ..
    instead you decided to care and
    troll her ..",
"context":
    "A minor child deserves privacy and
    should be kept out of politics.
    Pamela Karlan, you should be ashamed
    of your very angry and obviously
    biased public pandering, and using a child
    to do it .",

    "@USER If your child isn't named Barron ...
    \#BeBest Melania couldn't care less. Fact "
```

The response tweet,

`@USER @USER @USER I don't get this ...`

is a reply to its immediate context

`@USER If your child isn't ...`

which is a reply to

`A minor child deserves privacy ...`

TABLE I
DATASET SIZE STATISTICS

Train	Test
5000	1800

III. Experiments

The source code of all experiments is available on GitHub.

A. Project plan

Our team had clarity on the roles/responsibilities and categorized the task of classifying twitter sarcasm detection as follows:

- Understanding Data
- Data enrichment and pre-processing
- Implementing classic algorithms
- Research on SoTA models and Glue Leaderboard[1]
- Implementing State-of-the-Art Models
- Evaluation and Fine-tuning
- Current Leaderboard

[1]https://super.gluebenchmark.com/leaderboard/

B. Data enrichment and pre-processing

We've performed comprehensive Exploratory Data Analysis (EDA) on the given training dataset using traditional methods as well as the dedicated pre-processing python package "ekphrasis" to understand the dataset attributes like distribution, tokenization, lemmatization, normalization, annotation, spell correction, label encoding and word segmentation. This enriched training data is then used to train classic and SoTA models to predict the sarcasm responses.

C. Initial Modeling and Thoughts

The language of our choice is Python, and we started with simple yet powerful framework/library "sklearn" to implement classic models like Naïve Bayes (Multinomial), linear models like SGD (Stochastic Gradient Descent) and ensembles like Gradient Boosting. We've utilized stratified "k-folds" cross validation and split the dataset into 5 folds for cross training and validation.

```
skf = StratifiedKFold(
    n_splits=5,
    random_state=123,
    shuffle=True)
```

The initial round of classical methods used are Count Vectorizer, TF-IDF transformer and Multinomial Naïve Bayes.

```
text_clf = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', MultinomialNB()),
])
```

Using Macro average precision, here are the results for our initial round of modeling. We've introduced tokenization and word lemmatization to study the effect of the same on dataset and with ensemble methods like boosting. (XGB and GB)

```
Pipeline([
    ('vect', CountVectorizer(
        strip_accents = 'unicode',
         stop_words = 'english',
         lowercase = True)),
    ('tfidf', TfidfTransformer()),
    ('clf', GradientBoostingClassifier()),
])
```

TABLE II
CLASSICAL ML ALGORITHMS PERFORMANCE

Model	Precision	Recall	F1
Multinomial Naïve Bayes	0.7349	0.7808	0.7570
XGBoost	0.6548	0.7744	0.7095

Although the results are encouraging, we couldn't surpass the baseline.

TABLE III
AUTO SCIKIT LEARN RESULTS

Model	Accuracy
LibSVM (SVC)	0.7770
Random Forest	0.7624
Bernoulli NB	0.7545
SGD	0.7509
Extra Trees	0.7491
Passive Aggressive	0.7412
LibLinear (SVC)	0.7388
AdaBoost	0.7176
KNN	0.6921
DT	0.6321

D. Deep Neural Network (CNN, LSTM)

We've progressed to build a Deep Learning model and based on our study on research papers published in the domain of sentiment analysis and sarcasm analysis, we've chosen to use Convolution Neural Networks and Long Short-Term Memory (LSTM) deep learning techniques to classify the twitter sarcasm dataset. The framework of our choice is TensorFlow with Keras (a high-level abstraction layer) for building a Deep Learning model. We've followed a well established and documented process of DL model building using Keras. Below is the summary of our DL Model including Hidden layers.

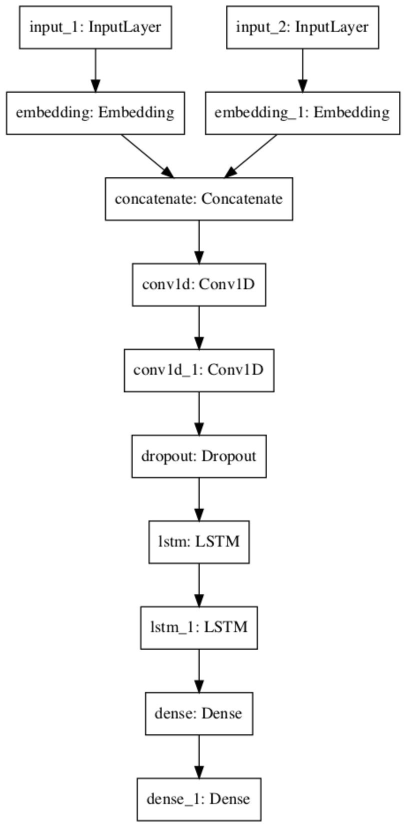
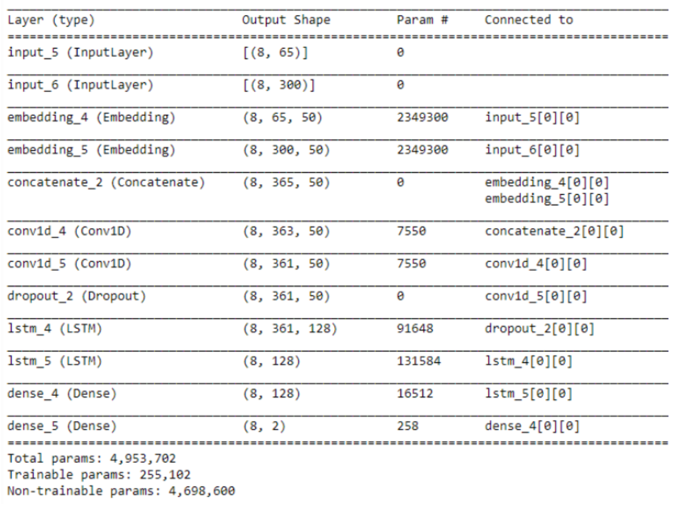


Fig. 1. Deep Learning model architechture



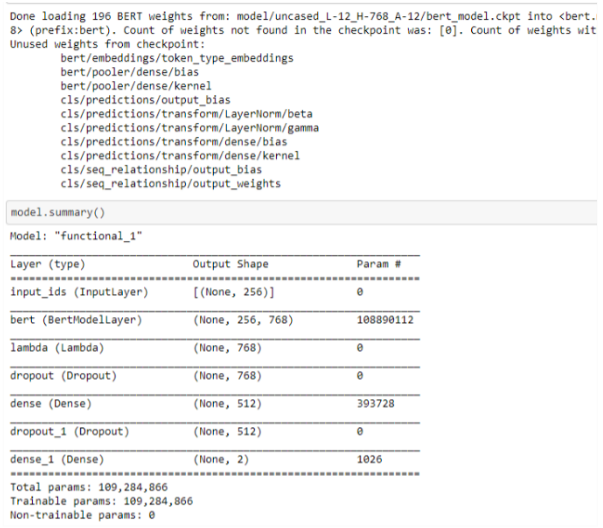
```
_____________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
=============================================================================================
input_5 (InputLayer)            [(8, 65)]            0
_____________________________________________________________________________________________
input_6 (InputLayer)            [(8, 300)]           0
_____________________________________________________________________________________________
embedding_4 (Embedding)         (8, 65, 50)          2349300     input_5[0][0]
_____________________________________________________________________________________________
embedding_5 (Embedding)         (8, 300, 50)         2349300     input_6[0][0]
_____________________________________________________________________________________________
concatenate_2 (Concatenate)     (8, 365, 50)         0           embedding_4[0][0]
                                                                 embedding_5[0][0]
_____________________________________________________________________________________________
conv1d_4 (Conv1D)               (8, 363, 50)         7550        concatenate_2[0][0]
_____________________________________________________________________________________________
conv1d_5 (Conv1D)               (8, 361, 50)         7550        conv1d_4[0][0]
_____________________________________________________________________________________________
dropout_2 (Dropout)             (8, 361, 50)         0           conv1d_5[0][0]
_____________________________________________________________________________________________
lstm_4 (LSTM)                   (8, 361, 128)        91648       dropout_2[0][0]
_____________________________________________________________________________________________
lstm_5 (LSTM)                   (8, 128)             131584      lstm_4[0][0]
_____________________________________________________________________________________________
dense_4 (Dense)                 (8, 128)             16512       lstm_5[0][0]
_____________________________________________________________________________________________
dense_5 (Dense)                 (8, 2)               258         dense_4[0][0]
=============================================================================================
Total params: 4,953,702
Trainable params: 255,102
Non-trainable params: 4,698,600
```

Fig. 2. Layer Dimensions

However, the model did not perform well, likely due to the lack of data and performed poorly than classic models (our first iteration)

E. State-Of-The-Art (SoTA) Models

The GLUE benchmark leaderboard in NLP is dominated by BERT and incremental updates/flavors of BERT. This is an incredible opportunity for us to implement some of the state-of-the-art models. We've utilized Google Colab to make use of in-built NVIDIA GPUs and computational power to train our BERT model. The model is trained on GPUs for 15 epochs with a batch size of 24. We've saved the model weights, parameters and checkpoints to a hdf5 file format (as recommended by TensorFlow). The weights are then loaded to perform predictions and are saved to answer.txt. This is clearly a Winner. We've successfully beat the baseline and achieved a ranking of #2 in leaderboard.



```
Done loading 196 BERT weights from: model/uncased_L-12_H-768_A-12/bert_model.ckpt into <bert.
8> (prefix:bert). Count of weights not found in the checkpoint was: [0]. Count of weights wit
Unused weights from checkpoint:
        bert/embeddings/token_type_embeddings
        bert/pooler/dense/bias
        bert/pooler/dense/kernel
        cls/predictions/output_bias
        cls/predictions/transform/LayerNorm/beta
        cls/predictions/transform/LayerNorm/gamma
        cls/predictions/transform/dense/bias
        cls/predictions/transform/dense/kernel
        cls/seq_relationship/output_bias
        cls/seq_relationship/output_weights
```

```
model.summary()
```

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_ids (InputLayer)       [(None, 256)]             0
_________________________________________________________________
bert (BertModelLayer)        (None, 256, 768)          108890112
_________________________________________________________________
lambda (Lambda)              (None, 768)               0
_________________________________________________________________
dropout (Dropout)            (None, 768)               0
_________________________________________________________________
dense (Dense)                (None, 512)               393728
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 1026
=================================================================
Total params: 109,284,866
Trainable params: 109,284,866
Non-trainable params: 0
```

Fig. 3. BERT - Layer Dimensions

IV. Results

A. Closing thoughts

We are working towards fine-tuning the winning model by augmenting ”rich” data like sentiments, emoticons understanding, context and relationships. We are currently focusing on implementing the different flavors of BERT like Facebook's RoBERTa, ALBERT with NAS, ERNIE and Google T5 to see if there is any significant improvement in our prediction accuracy and F1 scores thereby pushing our team to #1 in the Leaderboard.

References

[1] Weitzel, Leila & Prati, Ronaldo & Aguiar, Raul. (2016). "The Comprehension of Figurative Language: What Is the Influence of Irony and Sarcasm on NLP Techniques¿‘ 10.1007/978-3-319-30319-2_3.

[2] Lee, H., Yu, Y., and Kim, G., "Augmenting Data for Sarcasm Detection with Unlabeled Conversation Context", *arXiv e-prints*, 2020.

[3] Dong, X., Li, C., and Choi, J. D., "Transformer-based Context-aware Sarcasm Detection in Conversation Threads from Social Media", *arXiv e-prints*, 2020.

[4] Pant, K. and Dadu, T., "Sarcasm Detection using Context Separators in Online Discourse", *arXiv e-prints*, 2020.

[5] González-Ibáñez, R.I., Muresan, S., & Wacholder, N. (2011). Identifying Sarcasm in Twitter: A Closer Look. ACL.

[6] SoTA Sarcasm Detection Benchmarks https://paperswithcode.com/task/sarcasm-detection/codeless

Appendix A
Submission history

TABLE IV
Submission history

Model	**Precision**	**Recall**	**F1**	**Beat the baseline**	**Rank**
Classic	0.681	0.684	0.683	No	34
BERT	0.691	0.845	0.761	Yes	2
BERT+Sentiment	0.692	0.845	0.761	Yes	5