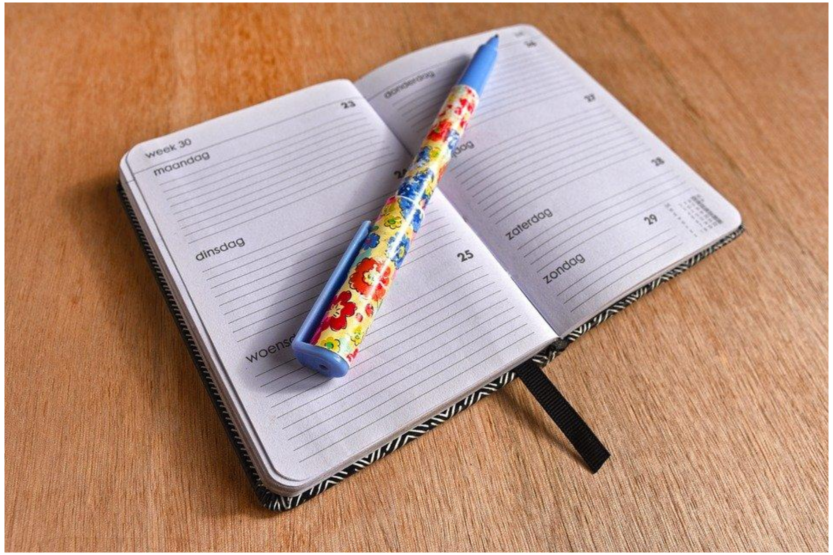
AmazingS3 Team

CS410 Text Classification Competition

Sahan	sithira2@illinois.edu
Santosh	kore3@illinois.edu
Suraj	surajb2@illinois.edu

Topics Covered

- Exploratory Data Analysis
- Models and respective best achieved F1 score
- Final selection: BERT model
- Adapting the BERT model to our problem
- Code flow and algorithm
- Code structure and walkthrough
- Graphs with additional insight
- Procedure
- Challenges
- Future Improvements
- References
- Q&A



Background

Problem Definition

Text Classification: Twitter Sarcasm Detection

Dataset format

Each line contains a JSON object with the following fields :

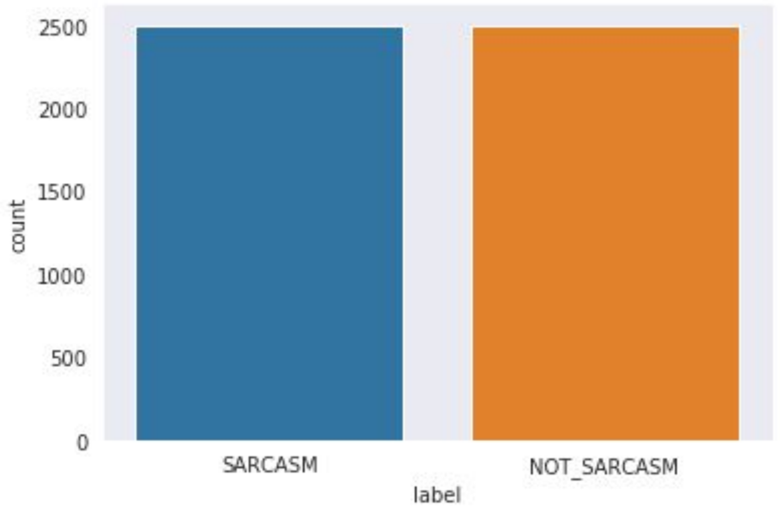
- *response* : the Tweet to be classified
- *context* : the conversation context of the *response*
  - Note, the context is an ordered list of dialogue, i.e., if the context contains three elements, `c1`, `c2`, `c3`, in that order, then `c2` is a reply to `c1` and `c3` is a reply to `c2`. Further, the Tweet to be classified is a reply to `c3`.
- *label* : `SARCASM` or `NOT_SARCASM`
- *id*: String identifier for sample. This id will be required when making submissions. (ONLY in test data)

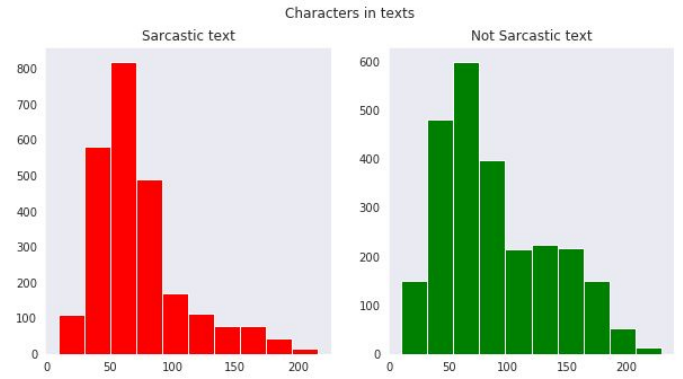
The response tweet, "@USER @USER @USER I don't get this..." is a reply to its immediate context "@USER If your child isn't..." which is a reply to "A minor child deserves privacy...".

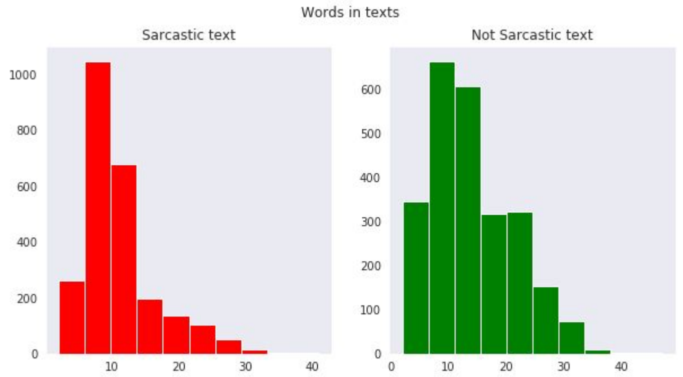
The goal is to predict the label of the "response" while optionally using the context (i.e, the immediate or the full context).

Exploratory Data Analysis

Data distribution for train.jsonl

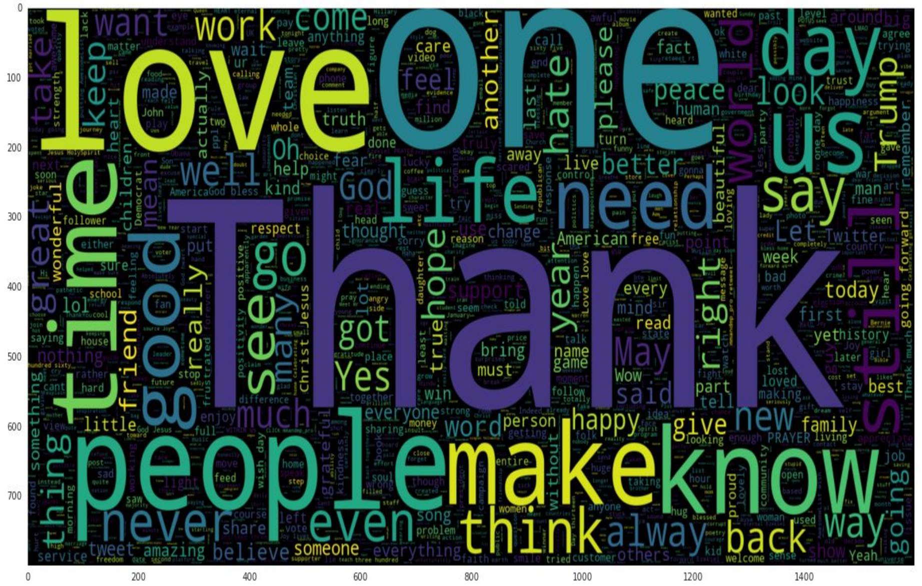




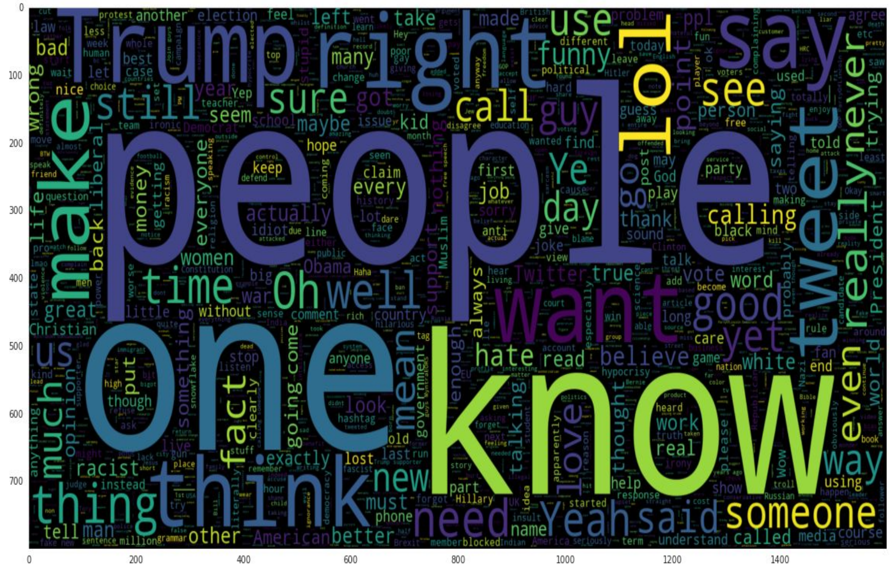


Exploratory Data Analysis (cont.)

Word Cloud Representations



NOT_SARCASM



SARCASM

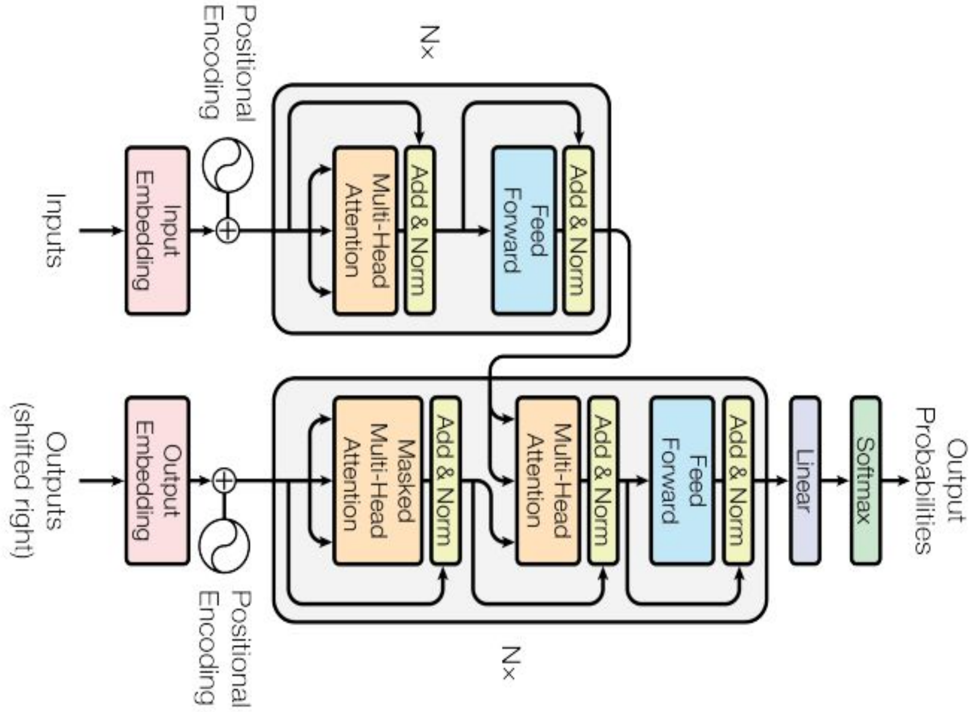
Models and F1 scores

Models	Optimization Step	F1 score
LSTM	BOW+TF-IDF	0.689009364218827
Bidirectional LSTM	BOW+TF-IDF	0.475915221579961 4
Sequence	BOW	0.720697167755991 4
RNN	BOW+TF-IDF	0.647596899224806 2
Spacy+LSTM	BOW+TF-IDF	0.677596899224806 2
CNN+GloVE Embeddings	BOW+TF-IDF	0.661259541984732 9
Logistic Regression	BOW+TF-IDF	0.648898001025115
Random Forest	BOW+default RF settings	0.696791685494803 4
SVC	BOW+TF-IDF	0.658588821250691 8
BERT		
	Using response column only	0.726504751847940 8
	Added immediate context	0.731707317073170 7
	Added full context	0.742857142857142 8
	Numbers and additional characters removed	**0.754296330701346 9**

BERT Model

Key Ideas

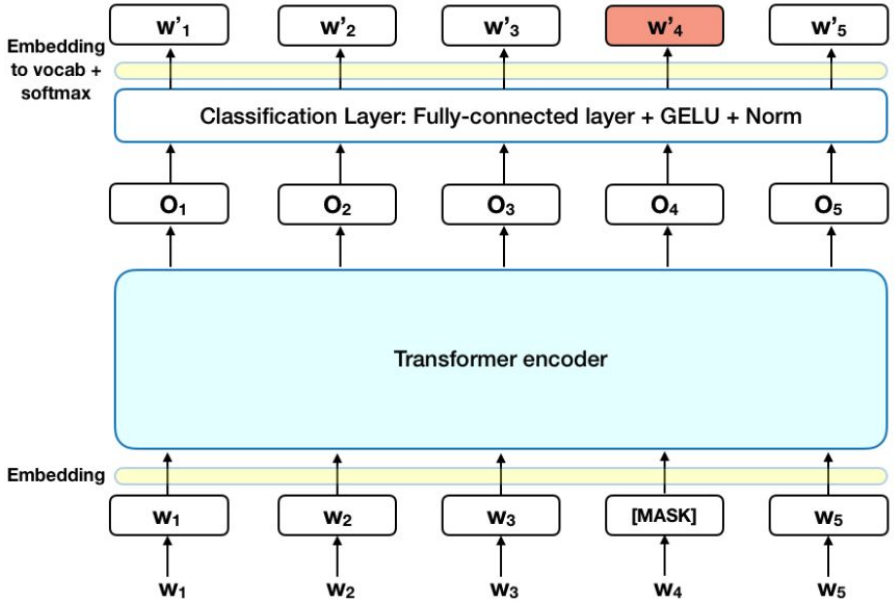
- Bidirectional - to understand the text you’re looking you’ll have to look back (at the previous words) and forward (at the next words).
- Transformers - The Transformer reads entire sequences of tokens at once.
- (Pre-trained) contextualized word embeddings



Source: https://nextjournal.com/chengchingwen/jsoc-2019-practical-implementation-of-bert-models-for-julia

Adapting BERT Model

- **Rationale**
  a. We observed that the the traditional ML models need to be either highly optimized and not enough to beat the baseline.
  b. The training dataset was not enough to train the DL models we tried (LSTM, CNN, RNN etc.)
  c. Preserving the sequence/order of words mattered and we needed a model that could understand the context.
- **Base model**
  a. bert-base-uncased was chosen to start with.
  b. Good enough to beat the baseline
  c. We observed that the accuracy can be improved by tuning many parameters.


w'1
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

Source: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

Code Flow and Algo

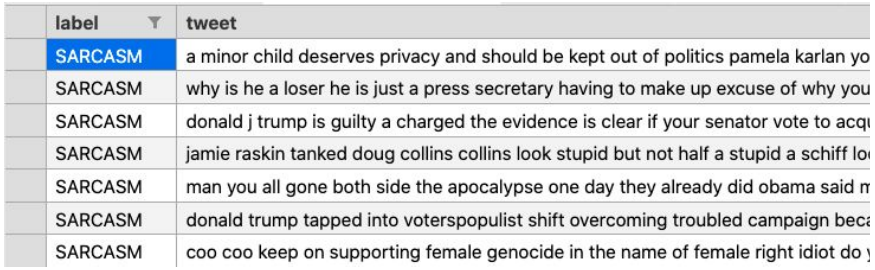
Cleaning – Pre-processing

1. Append all context to become one sentence and prefix it to the response.
2. Fix the tweet if it has special characters to support better expansion of contractions.
3. Remove all digits from the tweets.
4. Remove <URL> and @USER as they do not add any value.
5. Convert all tweets to lowercase.
6. Use NLTK's tweet processor to remove emojis, URLs, smileys and '@' mentions
7. Do hashtag segmentation to expand any hashtags to words.
8. Expand contracted words.
9. Remove all special symbols.
10. Perform lemmatization on the words.

Code Flow and Algo contd..

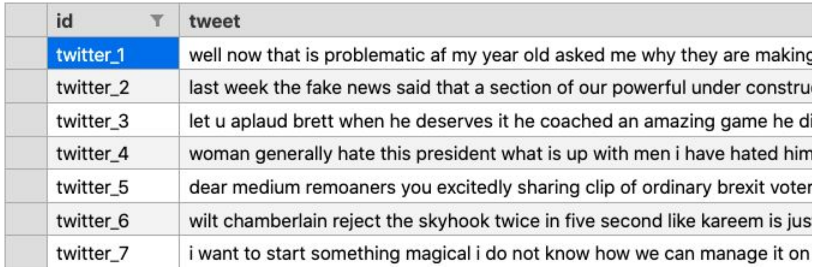
Data transformation

1. Train and Test datasets are loaded as Pandas DataFrames.
2. Each tweet preprocessed and appended to a new column named as "tweet".
3. Corresponding "label" or "id" and "tweet" columns of training and test datasets are written to CSV files.



label	tweet
SARCASM	a minor child deserves privacy and should be kept out of politics pamela karlan yo
SARCASM	why is he a loser he is just a press secretary having to make up excuse of why you
SARCASM	donald j trump is guilty a charged the evidence is clear if your senator vote to acqu
SARCASM	jamie raskin tanked doug collins collins look stupid but not half a stupid a schiff lo
SARCASM	man you all gone both side the apocalypse one day they already did obama said m
SARCASM	donald trump tapped into voterspopulist shift overcoming troubled campaign beca
SARCASM	coo coo keep on supporting female genocide in the name of female right idiot do y

train.csv



id	tweet
twitter_1	well now that is problematic af my year old asked me why they are making
twitter_2	last week the fake news said that a section of our powerful under constru
twitter_3	let u aplaud brett when he deserves it he coached an amazing game he di
twitter_4	woman generally hate this president what is up with men i have hated him
twitter_5	dear medium remoaners you excitedly sharing clip of ordinary brexit voter
twitter_6	wilt chamberlain reject the skyhook twice in five second like kareem is jus
twitter_7	i want to start something magical i do not know how we can manage it on

test.csv

Code Flow and Algo contd..

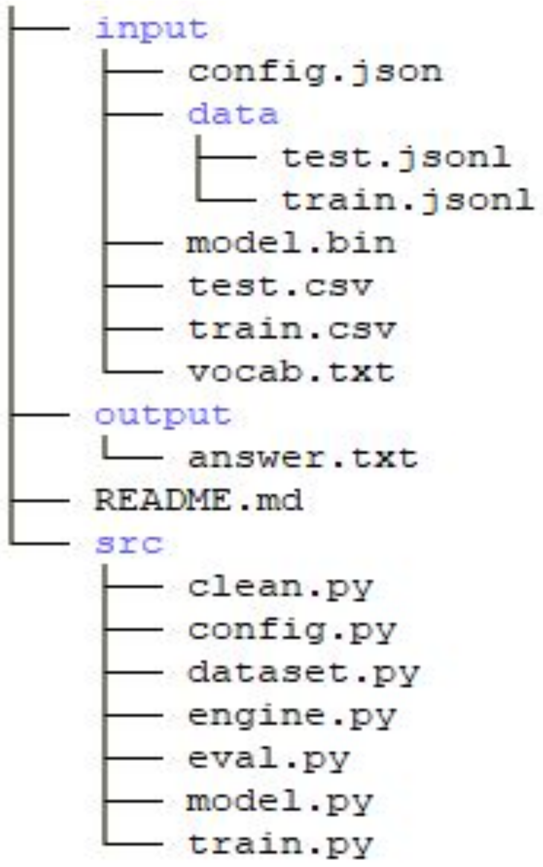
Training

1. Read in the train.csv from prior step.
2. Training dataset (5000 records) is split into training and validation as 80:20 ratio.
3. Feed in the parameters to the model.
4. Perform model training for the given number of epochs.
5. Calculate validation accuracy for each run and save the best model as a bin file

Evaluation

1. Load the test.csv file from data transformation step.
2. Load the best performing model from the training step.
3. Perform predictions for each test tweet (1800 total records)
4. Generate answer.txt that will be submitted to the grader to the "output" folder.

Code Structure and Walk-through



```
├── input
│   ├── config.json
│   ├── data
│   │   ├── test.jsonl
│   │   └── train.jsonl
│   ├── model.bin
│   ├── test.csv
│   ├── train.csv
│   └── vocab.txt
├── output
│   └── answer.txt
├── README.md
└── src
    ├── clean.py
    ├── config.py
    ├── dataset.py
    ├── engine.py
    ├── eval.py
    ├── model.py
    └── train.py
```

Environment Setup and execution Procedure

Prerequisites

- Anaconda 1.9.12
- Python 3.8.3
- PyTorch 1.7.0
- Transformers 3.0.0

Install dependencies

1. Libs
```
pip install tweet-preprocessor textblob wordsegment contractions tqdm
```
2. Download TextBlob corpora
```
python -m textblob.download_corpora
```
3. Install PyTorch & Transformers
```
conda install pytorch torchvision torchaudio cpuonly -c pytorch transformers
```

If it complains that the `transformers` lib's not installed, try this command:
```
conda install -c conda-forge transformers
```

Sequence of operations

First, `cd src` and run the following commands,

tl;dr

```
python clean.py && python train.py && python eval.py
```

Description of each step:

1. Clean the dataset ***python clean.py***
2. Train the model ***python train.py***
   Once the model is trained it will create an `input/model.bin` file which saves our model to a binary file. We can later load this file (in the evaluation step) to make predictions.
3. Make predictions & create the answer.txt file ***python eval.py*** The answer.txt file is created at output folder.

Challenges

- Trying various model and optimizing respective parameters with time constraint.
- Learning and implementing simultaneously.
- Cleaning-up data is most important task but not performed for all the models.
- BERT model requires lots of resource and takes around 2-3 hrs for model building.

Future improvements

- Cleaning data further
- Optimizing BERT model parameters and trying different BERT model (eg. RoBERTa)
- Re-use some of the tried models and optimizing to beat F1 scores
- Extract Emoji's to add more meaning to the sentiments of the tweets
- Data augmentation steps to prevent overfitting.
- Try an ensemble of models (eg. BERT + VLaD etc.)
- Run our model on different test data and compare results against state-of-art

References

- https://github.com/abhishekkrthakur/bert-sentiment
- https://curiousily.com/posts/sentiment-analysis-with-bert-and-hugging-face-using-pytorch-and-python/
- https://huggingface.co/transformers/pretrained_models.html
- https://textblob.readthedocs.io/en/dev/quickstart.html#words-inflection-and-lemmatization
- https://www.nltk.org/
- https://scikit-learn.org/

Q&A

THANK
YOU