CS6120 – NLP

Spring 2022

Final Project

Group Members: Kimberly Boynton O’Brien, Bill Collins

Russian Troll Tweet Identification

Introduction

In the summer of 2018, the site FiveThirtyEight made a set of 3 million tweets available that were issued from the Internet Research Agency (IRA) which runs a Russian “troll factory”. This group was a defendant in Robert Mueller’s investigations into Russian interference in American politics.

For this project we chose to try to distinguish these types of “troll” tweets from normal user tweets. This exercise may uncover important indicators of rhetoric which is attempting to psychologically manipulate people through social media.

Using these tweets and an equal sized set of normal user tweets we will perform feature analysis to determine which aspects of the text are most salient. After this, we’ll explore different methods of processing the text and converting it into a format that is consumable by downstream models. We will then test and evaluate different models on the binary classification task of distinguishing troll from non-troll tweets. Finally, we will present reflections on our findings and suggest future improvements.

Data

Source and Description

The troll data consists of 2,973,371 tweets from 2,848 Twitter handles from the Internet Research Agency “troll factory” sent between February 2012 and May 2018, with the vast majority in 2015-2017. The data was originally compiled by Clemson University researchers Darren Linvill and Patrick Warren using a Salesforce tool called Social Studio.

The tweets have been labeled by Linvill and Warren under several different subclasses of troll tweet; RightTroll, LeftTroll, Newsfeed, HashtagGamer, FearMonger, Commercial, and Unknown. RightTroll and LeftTroll respectively play the parts of far right and far left political posters, with the aim of increasing divisiveness between the two parties. The other types are more benign in nature.

The regular user data comes from Cheng et al., who provided over 5 million tweets between September 2009 and January 2010, originally used in the paper “You Are Where You Tweet: A Content-Based Approach to Geo-locating Twitter Users in CIKM 2010”.

The two sets of tweets provide different fields, but we are only using the text field and the subclass labels (using “NormalUser” for non-troll tweets). These subclass labels may be used for a multiclass class analysis of the troll data on its own.

Surface Observations and Hypotheses About Key Features

There are some immediate obvious features that seem promising. Tweets often contain other user mentions (tokens starting with “@”), hashtag topics (tokens starting with “#”), and links to other tweets (tokens starting with “http”). Since the users and links are overly specific, we will replace them with the tags “<USER>” and “<LINK>” respectively. Hashtags provide more general topic information so we will leave them as is and later examine their influence on model performance.

Textual style and topical differences are obvious between troll and non-troll (which we will just call “user”) tweets. As mentioned, the troll tweets are largely about politically divisive topics whereas the user tweets have a more mundane topic range. The spelling is noticeably better in the troll tweets as well. This indicates that a binary dictionary word lookup feature may be relevant.

Specific famous people, retail businesses, and places are frequently mentioned in the tweets, so named entity recognition will likely be important.

For the reasons mentioned above, it seems likely that topic, named entities, user mentions, links, and dictionary word presence will provide a strong information signal for the models to work with.

Cleaning and processing to get features

To prepare the data, the “tweet\_type” (“NormalUser” or one of the troll types) and “tweet\_text” were extracted from the raw data. User mentions, links, and emojis were replaced with “<USER>”, “<LINK>”, and “<EMOJI>” respectively. Non-English language tweets were skipped. The spacy library (Honnibal et al.) was used to tokenize the text from which lemmas, POS tags, phrases, and entities were then extracted.

Using this textual information, an additional set of features was derived which can be seen in the example below. We selected a balanced subset of a million tweets for our experiments.

Example tweet record:

{'type': 'NormalUser',

'text': 'RT: Good to see Junie Browning back to being Junie Browning (VIDEO): ',

'tokens': 'RT : Good to see Junie Browning back to being Junie Browning ( VIDEO ): ',

'lemmas': 'rt : good to see junie browning back to be junie browning ( video ): ',

'pos': 'NNP : NFP NNP XX JJ TO VB NNP NNP RB IN VBG NNP NNP -LRB- NNP : NFP NNP ADD',

'phrases': 'Junie\_Browning Junie\_Browning',

'entities': ['Junie\_Browning:PERSON'],

'ent\_types': ['PERSON'],

'hashtags': [],

'oov\_words': 'rt junie junie ):',

'emoji\_ratio': 0.0,

'link\_ratio': 0.047619047619047616,

'user\_ratio': 0.047619047619047616,

'oov\_ratio': 0.3076923076923077}

Data Statistics

Stats and charts about these features showing differences between the classes

Data Representation for Models

We plan to explore permutations of several different components, including the choice of text processing method, word and document level vectorization methods, and models.

Text processing

Text processing choices will include the simple tokenized text with a vocabulary size limited by frequency, and the extracted lemmas which have a naturally smaller vocabulary.

Vectorization

The document level vectors are created from several methods in the scikit-learn library (Buitinck, et al.) and distributed embedding methods in the gensim library (Radim et al.). These include CountVectorizer, TfidfVectorizer (TF-IDF), Latent Semantic Analysis (LSA) applied to TF-IDF vectors for dimensionality reduction, Doc2vec from gensim, Fast Sentence Embedding (FSE), and Summed Word2Vec & FastText vectors weighted by TF-IDF score.

Data Representation Results

We evaluated the merit of different data representations by testing them with baseline models, and by visualization of the class clustering using the t-SNE algorithm from the scikit-learn library (Buitinck, et al.). This was very important as it gives a very strong indication of the difficulty of the discrimination task. We can get an idea of the class overlap from these visualizations and choose the vectorization method which displays the most promising class separability.

**TF-IDF**

TF-IDF vectorization was trained with a maximum vocabulary size of 50K, with English stopwords excluded. Given the transformed data, we can extract the TF-IDF scores for each word, which can be interpreted as representing their importance in the tweet.

For example, the tweet “'rt : good to see junie browning back to be junie browning ( video ):'”

We would expect low scores for more generic words like “good” and higher scores for more specific words (“junie”), which turns out to be the case:

|  |  |
| --- | --- |
| word | score |
| good | 0.1275 |
| junie | 0.7282 |

Using these values as “importance scores”, we can additionally use them as weight coefficients applied to a summation of word level vectors. This will allow the vectors for the most important words to best influence the direction in vector space.

**LSA Applied to TF-IDF**

Applying Singular Value Decomposition to the TF-IDF document matrix serves to compress the vocabulary-sized dimension (50K) down to a smaller value, while at the same time extracting the dominant components in vector space. In our experiments, we found that this operation reduced the accuracy of models by around 10%. We believe it’s because the tweets are so short that providing direct information about each token is necessary for the discrimination task.

**Doc2Vec**

Doc2Vec is trained with targets included for each document. This allows the model to implicitly integrate class information while learning the embeddings. This also allows us to use the model directly as a classifier, by finding the class that’s closest to the vector inferred for each sample. Using doc2vec on its own proved to be a poor classifier (0.7 accuracy).

**Fast Sentence Embedding**

Fast Sentence Embedding (Borchers, et al.) combines word2vec representations into a single document level vector in a more mathematically sophisticated way than simple averaging. These vectors display a stronger structure than doc2vec.

**TF-IDF Weighted Sum of Word2Vec & FastText Vectors**

These document vectors were created by summing word2vec or fasttext word vectors weighted by TF-IDF score. This was a slow process, so it was only run on a subset of the data in order to compare clustering with other document vectorization methods.

T-distributed Stochastic Neighbor Embedding (T-SNE) Visualizations of Document Vectors

T-SNE (van der Maaten, et al.) is an algorithm which iteratively reduces high dimensional data to 2D so that it can be plotted. It’s an excellent way to get an idea of class separability and clustering.

When plotting the 4 vectorization types mentioned above, we find that doc2vec classes display massive overlap, FSE displays much better separability of classes, and both TF-IDF methods also display decent separability (with FastText being slightly superior). The overlap in these visualizations also highlights the difficulty of the discrimination task. One additional point of note is that NormalUsers appear to overlap more with LeftTroll than RightTroll which may indicate that the average Twitter user is not strongly right leaning.



Models

We used sklearn (Buitinck, et al.) for the lightweight modeling using LogisticRegression, SVM, and MultiLayerPerceptron, and we built a custom torch model for more sophisticated feature targeting and control. The torch model has separate embedding for text, POS, entity types, and hashtags.

Lightweight Models

For vectorization, TFIDFVectorizer was used to vectorize both tokens and lemmas individually. This was done to see if tokens or lemmas would return a better result. We used max features of 50000 words, and English stop words, then separated the data into training and test sets, after shuffling it, with training being everything but the last 10k and test being the last 10k.

Both LinearSVC and Logistic Regression models were used to evaluate both tokens and lemmas using supervised learning, separating the data based on user or troll.

It was found that the precision from the texts were more accurate than the lemmas, though between Linear SVC and Logistic Regression there was no noticeable difference when comparing the text, however

Experiments with Text Representation and Vectorization Method

|  |  |  |  |
| --- | --- | --- | --- |
| Tokens/Lemmas | Count/TF-IDF | Model | Accuracy |
| lemmas | count | Logistic regression | 0.89342 |
| tokens | count | **Logistic regression** | **0.90816** |
| lemmas | tfidf | Logistic regression | 0.88984 |
| tokens | tfidf | Logistic regression | 0.90018 |
| lemmas | count | svm | 0.89394 |
| tokens | count | **svm** | **0.907** |
| lemmas | tfidf | svm | 0.89576 |
| tokens | tfidf | svm | 0.90296 |

From these tests we can see that tokens and count vectorizer are the preferred text features. This indicates that individual words are very important for the classification task, and any reduction in the word-level information is harmful. This is likely due to the shortness of the tweets.

Using Model Coefficients to Score Most Relevant Words

After training Logistic Regression or SVM on TF-IDF features we can observe the relative importance of words by examining the coefficients learned by the models. The results were typically not that informative but were clear when we trained the model to only distinguish LeftTroll from RightTroll. Below are the more politically charged words corresponding to the largest coefficients:

enlist 4.967805844812494  
realdonaldtrump 4.271653083374999  
wiunion 3.7685324282743036  
dineshdsouza 3.519830736532676  
antifa 3.3886375955439094  
bb4sp 3.3689422085758447  
iamonfire 3.2949869615021914  
thingsmoretrustedthanhillary 3.111873659794456  
amb 3.103703950396433  
maga 3.095712685658919

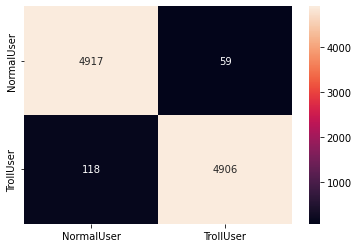
Linear Multi-feature Model

Our top performing model features input embedding channels ("EmbeddingBag" with mean aggregation) for text (whose Embedding layer was populated with pretrained 128 dim Word2Vec vectors), POS, Entity types, and Hashtags. The OOV/user/link/emoji tag ratio features are included and are run through a linear transformation and concatenated to the embedding results.

Dropout is applied to this concatenated vector which is then passed through 2 hidden layers (512 and 128 dimensions respectively) with ReLU activations and Batch Normalization. An Adam optimizer is used for calculating parameter updates. Initial learning rate is set to 1e-3 and a Learning Rate Scheduler reduces the LR by one magnitude when the cross-validation (CV) accuracy plateaus. Log Softmax is performed on the output, and the criterion is Negative Log Likelihood.

The model is saved every time the CV accuracy improves. The training stops early when the CV accuracy doesn’t improve for 5 epochs. The last saved model is the one with the highest CV accuracy, which is reloaded and run on the test set. This model hit its best CV accuracy at epoch 7.

precision recall f1-score support  
  
 NormalUser 0.9766 0.9881 0.9823 4976  
 TrollUser 0.9881 0.9765 0.9823 5024  
  
 accuracy 0.9823 10000  
 macro avg 0.9823 0.9823 0.9823 10000  
weighted avg 0.9824 0.9823 0.9823 10000



Ablation Studies on the Multi-Feature Model

We are interested in learning which features are the most important to the success of the model, so we perform ablation studies by masking out the influence of the different feature inputs and noting the effect on model accuracy.

|  |  |
| --- | --- |
| **Fields Present (text, POS, entity type, hashtags, ratio feats)** | **Accuracy at 3 Epochs** |
| Text, POS, entity, hashtag, ratio (all) | 0.9813 |
| POS, entity, hashtag, ratio (text removed) | 0.9623 |
| Text, entity, hashtag, ratio (pos removed) | 0.9512 |
| Text, POS, hashtag, ratio (entity removed) | 0.9805 |
| Text, POS, entity, ratio (hashtag removed) | 0.9740 |
| Text, POS, entity, hashtag (ratio removed) | 0.9807 |
| Text only | 0.9226 |
| POS only | 0.9155 |
| Entity only | 0.6261 |
| Hashtag only | 0.7539 |
| Ratio only | 0.5972 |
| **Text, POS Only (effect of POS in addition to text)** | **0.9736** |
| Entity, Hashtag, Ratio (Text, POS, most important features, removed) | 0.8298 |
| Text Only, without Word2Vec initialization | 0.9182 |

These results indicate that unsurprisingly text is the most important field, followed closely by POS. A small advantage is gained by initializing the text embeddings with pretrained Word2Vec vectors. It’s very interesting that by using only POS we can get 0.91 accuracy! It’s also clear that including POS makes a very significant contribution to the accuracy (0.92 -> 0.97) by providing additional information about the text, much needed in short tweets. Even when removing text and POS the model is still able to extract enough information from entity types, hashtags and user/link/emoji/OOV ratio to get to 0.83 accuracy.

Reflection and Future Improvement

Explain strengths and weaknesses of each feature type with each model.

Evaluation of hypotheses

Evaluation of hardest data

Evaluation of key features via model (logistic regression)

Possible Improvements

Noting how

Key takeaways and conclusion

Acknowledgements/Citations

**Acknowledgments**

Troll Tweet dataset: <https://github.com/fivethirtyeight/russian-troll-tweets>

Normal Tweet dataset: <https://archive.org/details/twitter_cikm_2010>

<https://fivethirtyeight.com/features/why-were-sharing-3-million-russian-troll-tweets/>

<https://towardsdatascience.com/3-essential-ways-to-calculate-feature-importance-in-python-2f9149592155>

**Citations**

Borchers, O. “Fast sentence embeddings.” <https://github.com/oborchers/Fast_Sentence_Embeddings>. (2019)

Lars Buitinck, et al. "API design for machine learning software: experiences from the scikit-learn project." *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013.

Z. Cheng, J. Caverlee, and K. Lee. “You Are Where You Tweet: A Content-Based Approach to Geo-locating Twitter Users.” *In Proceeding of the 19th ACM Conference on Information and Knowledge Management (CIKM), Toronto*, Oct 2010.

Honnibal, M., & Montani, I. “spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.” (2017)

Rehurek, Radim et al. "Gensim–python framework for vector space modelling". *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* 3. 2(2011).

van der Maaten, L.J.P.; Hinton, G.E. “Visualizing High-Dimensional Data Using t-SNE.” *Journal of Machine Learning Research* 9:2579-2605, 2008.