Comments on Kim’s Paper

Violent Politial Rhetoric on Twitter

Discussant: David Yen-Chieh Liao

17 November 2021

Comments

- The Findings:
  1. The paper systematically examines violent contents expressed in Twitter by utilizing machine learning methods to find the rhetoric pasterns.
  2. The finding point to the evidence that the violent tweets closely occured the preceding the Capital Riot.
- Contribution:
  1. automated method to discover the pasterns in violent rhetoric.
  2. hard work: labeling the data and then training the classifiers (Kim's model/classifier)
  3. new approach to study political conflict.

A Few Concerns

- Performance (and Performance Measures)

TABLE A4 *The average performance of classifiers from 5-fold cross validation*

Model	Precision	Recall	F-1
Logistic Regression + Count Vector	68.64	32.78	44.33
Logistic Regression + TF-IDF Vector	80.75	9.91	17.62
Logistic Regression + GloVe	60.58	10.66	18.09
Random Forest + Count	77.83	19.17	30.67
Random Forest + TF-IDF Vector	80.69	17.34	28.50
Random Forest + GloVe	74.14	10.97	19.02
XGBoost + Count Vector	78.15	7.74	14.06
XGBoost + TF-IDF Vector	79.49	11.67	20.28
XGBoost + GloVe	68.15	14.18	23.46
BERT	74.02	59.05	65.69

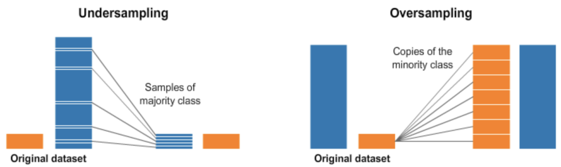
1. Confusion Matrix? (Precision: TP / TP+FP ; Recall: TP / TP+FN)

2. trade-off between a precision/recall: AUC, Precision versus recall

3. ROC curves?

The Classifiers

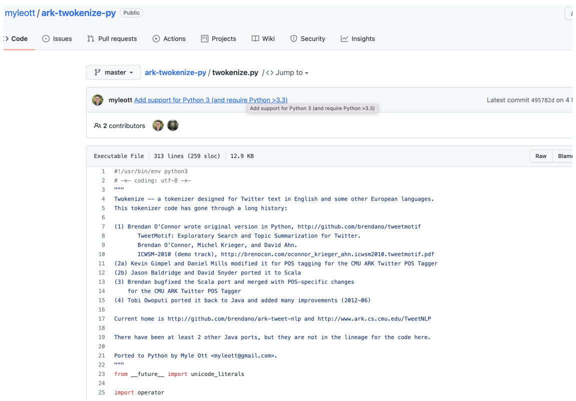
- Imbalanced training class (reference: https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/)



- use `imblearn` with `oversampling method` or `undersample method` to generate new samples to balance the classes before cv. (solution reference: https://kiwidamien.github.io/how-to-do-cross-validation-when-upsampling-data.html)
- feature scaling: min-max or standardization ideally.
- bruteforcefully generate more violent contents based on the domain knowledge

Difficulties in Training Twitter's Data

- Noisy words in twitter data and the tweet length is limited
  1. countvectorizing tweets might generate not much sense feature for the classifiers (symbols, emoji, emoticon, unformal expressions).
  2. Suggestion one: `CountVectorizer()` with `tokenizeRawTweetText()` from twokenize.py (https://github.com/myleott/ark-twokenize-py)

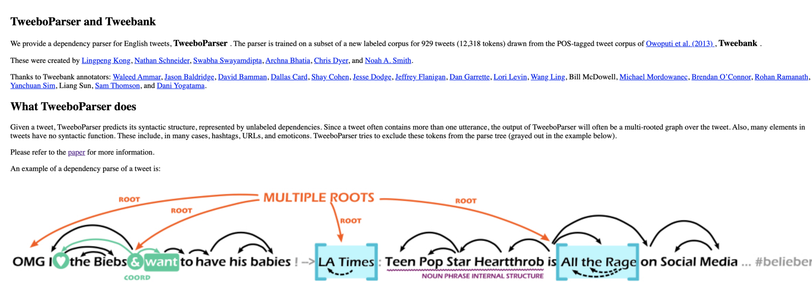


Difficulties in Training Twitter's Data

- Suggestion two:

  1. Use `TweeboParser` from **Tweet NLP** (https://www.cs.cmu.edu/~ark/TweetNLP/#parser_paper) at Carnegie Mellon: part-of-speech tagger for tweets, its training data of manually labeled POS annotated tweets, a web-based annotation tool, and hierarchical word clusters from unlabeled tweets.

  2. Trai selective feature like active verb, specific tags and emoji like `:o` `:/`` >.<` `XD` `-__-`



TweeboParser and Tweebank

We provide a dependency parser for English tweets, **TweeboParser** . The parser is trained on a subset of a new labeled corpus for 929 tweets (12,318 tokens) drawn from the POS-tagged tweet corpus of Owoputi et al. (2013) , **Tweebank** .

These were created by Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith.

Thanks to Tweebank annotators: Waleed Ammar, Jason Baldridge, David Bamman, Dallas Card, Shay Cohen, Jesse Dodge, Jeffrey Flanigan, Dan Garrette, Lori Levin, Wang Ling, Bill McDowell, Michael Mordowanec, Brendan O'Connor, Rohan Ramanath, Yanchuan Sim, Liang Sun, Sam Thomson, and Dani Yogatama.

What TweeboParser does

Given a tweet, TweeboParser predicts its syntactic structure, represented by unlabeled dependencies. Since a tweet often contains more than one utterance, the output of TweeboParser will often be a multi-rooted graph over the tweet. Also, many elements in tweets have no syntactic function. These include, in many cases, hashtags, URLs, and emoticons. TweeboParser tries to exclude these tokens from the parse tree (grayed out in the example below).

Please refer to the paper for more information.

An example of a dependency parse of a tweet is:

Performance Measures (Suggested):

- Saving holdout from tainset for validation
    - hold out part of the training set to evaluate several candidate models and select the best one.
    - if good number of the validation set can be extract from training.
- One-vs-the-one strategy (aka One-vs-the-rest, OVR).

`sklearn.multiclass`.OneVsRestClassifier

class sklearn.multiclass.OneVsRestClassifier(*estimator*, *, *n_jobs=None*) [source]

One-vs-the-rest (OvR) multiclass strategy.

Also known as one-vs-all, this strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes. In addition to its computational efficiency (only `n_classes` classifiers are needed), one advantage of this approach is its interpretability. Since each class is represented by one and one classifier only, it is possible to gain knowledge about the class by inspecting its corresponding classifier. This is the most commonly used strategy for multiclass classification and is a fair default choice.

OneVsRestClassifier can also be used for multilabel classification. To use this feature, provide an indicator matrix for the target `y` when calling `.fit`. In other words, the target labels should be formatted as a 2D binary (0/1) matrix, where [i, j] == 1 indicates the presence of label j in sample i. This estimator uses the binary relevance method to perform multilabel classification, which involves training one binary classifier independently for each label.

Closing Mark

- method: very promising and highly innovated
- thoughtful research design
- policy implication