Project Title: Irony Detection on Twitter

Project Number: 18

Group Members:

Surname, First Name	Student ID	STAT 441	STAT 841	CM 763	Your Dept. e.g. STAT, ECE, CS
Mengyun, Zheng	20550326	☐	✓	☐	CS
Ningsheng, Zhao	20550707	☐	✓	☐	STATS
		☐	☐	☐	
		☐	☐	☐	

Your project falls into one of the following categories. Check the boxes which describe your project the best.

1. ☐ **Kaggle project.** Our project is a Kaggle competition.
   - This competition is active ☐ inactive ☐.
   - Our rank in the competition is ... .
   - The best Kaggle score in this competition is ... ..., and our score is ....
2. ☐ **New algorithm.** We developed a new algorithm and demonstrated (theoretically and/or empirically) why our technique is better (or worse) than other algorithms.
3. ✓ **Application.** We applied known algorithm(s) to some domain.

   ✓ We applied the algorithm(s) to our own research problem.
   ☐ We tried to reproduce results of someone else's paper.
   ☐ We used an existing implementation of the algorithm(s).
   ☐ We implemented the algorithm(s) ourself.

Our most significant contributions are (List at most three):

(a) Implement multiple useful pre-processing techniques on the noisy raw Twitter text.
(b) Experiment with all most used word library and lexicons.
(c) Achieve comparatively better result with more limited dataset

List the name of programming languages, tools, packages, and software that you have used in this project:
NLTK, emoji, word2vec, sklearn, xgboost, gensim, bs4, enchant, numpy, pandas
.

1 Introduction and Related Work

Twitter is one of the most popular social media which is now widely used to share opinions and thoughts about any topic in surrounding world. A tweet, a piece of message limited to most 140 characters, is possible to reflect the user's daily life event, personality, and preference towards a specific topic.

We participated in the International Workshop on Semantic Evaluation 2018(SemEval 2018)[1]Task-3A: Irony Detection in English Tweets, which is binary classification task to predict if an English tweet is ironic or not. Irony detection problem is an important component of sentiment analysis in the field of Natural Language Processing(NLP).

Merriam-Webster[2] provides definitions of irony as follows: *a: the use of words to express something other than and especially the opposite of the literal meaning; b: a usually humorous or sardonic literary style or form characterized by irony; c: an ironic expression or utterance.* To detect irony, some prior knowledge is required, e.g. politics, reasoning, literature and so on. In our project, we will not deal with such complex conditions, neither the different types of irony(verbal irony and situational irony) which is the second subtask of the competition. In this project, we define a tweet is ironic if its literal meaning is emotionally opposite to what the user wants to express. Take an example labeled ironic from raw data:"Oh, thank GOD - our entire office email system is down... the day of a big event. Santa, you know JUST what to get me for xmas." Even though the user expresses 'thank GOD' , 'Santa', and gift for 'xmas', the user says the entire email system is down, so it is an irony to express a negative sentiment.

Twitter is a frequent dataset in irony detection since it is easy to collect data from Twitter API and raw data is self-labeled through hashtags(#irony, #sarcasm, and #notsarcasm). Many researchers extract features from tweets contents and train classifiers using classical machine learning methods.Riloff et al used emoticons and punctuation marks to detect if a positive sentiment exists in a negative situation for irony detection [3] .Ptcek et al used n-grams, word-shape patterns, and pronunciations as features[4]. Gonzalez-Ibanez et al used a combination of unigrams, lexical features, and emotions as the feature to train classifiers through support vector machine (SVM) and logistic regression[5]. Cliche used n-grams and topics to extract features and trained SVM and Naive Bayes classifiers on self-collected tweets source. He claimed that SVM gives him better result than previous work.[6]

2 Method

2.1 Datasets

SemEval[1] provides training datasets which are collected by searching hashtags #irony,#sarcasm, and#not on Twitter. Moreover, each tweet in the dataset is manually labeled using a fine-grained annotation scheme for irony [7]. to minimize the noise. In order to keep the consistency of annotation, we only use the given data in our project, which contains 3834 instances(1911 are ironic and 1923 are nonironic). In the competition, the test data will be provided in evaluation period(Jan 2018), so we split training data into training and validation set. Specifically, we have 3067 tweets in training set of which 1520(49.56%) are labeled as ironic, and 1547(50.44%) as nonironic, 767 tweets in the validation dataset.

2.2 Evaluation

Our work will be evaluated using standard evaluation metrics provided by the organizer as follows:

$$Accuracy : \frac{number of correctly predicted instances}{total number of instances}$$
$$Precision : \frac{number of correctly predicted instances}{number of predicted labels}$$
$$Recall : \frac{number of correctly predicted labels}{number of labeled in the gold standard}$$
$$F_1 - score : 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

2.3 Pre-processing

In order to extract suitable features for classifiers, we clean up the text as following steps using Python NLTK library[8]

- Replace Emojis.
  It is very common to use emojis to express users' true sentiments. Emojis only appear in 2017 testing dataset, and we detect none in training dataset and 2016 testing dataset. Therefore, the method we choose to handle emojis will affect the test accuracy. Instead of removing them, we replace emojis(Unicode) with their names by using emoji library. For example, a emoji whose unicode is $'U0001f308'$ will be replaced with "face blowing a kiss" after pre-processing.

- Replace URLs, usernames and remove numbers and stopwords.
  Hyperlinks are replaced with "URL" and "usernames" are replaced with "USER" in tweets to show the presence. Hashtags have a symbol as a prefix of a word or a phrase used to recognize the topic a tweet towards, so the "#" character is removed and the topic content is kept. Also, we remove numbers and stopwords and maintain each tweet in English only.

- Handling misspelled words.
  In tweets. It is common for users to express feelings through misspelled words like "coooool", which cannot be detected in words lexicon. For each word in a tweet, we check if it has repeated letters and if exits in Wordnet library through NLTK[8]. If so, extra letters are removed until the word can be detected in the library.

- Handling abbreviation and negations.
  For each tweet, the abbreviation in English like its, havent is checked one by one and is transformed into the original form like it is , have not. Negations play a significant role in irony detection since it can flip the real meaning of a sentence easily. We detected if negations exist in each tweet and if followed by an adjective. If so, the negation and the adjective are replaced with the antonym using NLTK corpus[8]. For example, not happy will be replaced with unhappy.

- Using lowercase, tokenization and stemming.
  The final tweets in our training datasets are lowercased, splitted into token by using NLTK library. The library is also used to apply stemming algorithms to strip off prefixes and suffixes like "shown" and "showing" becomes "show".

2.4 Feature Extraction

We experimented a number of standard features typically used in sentiment analysis as follows:

- N-grams.
  Through using unigram, bigram, and trigram(n = 1-3), we generate sparse vector representations. These are collections of one word, two words, and three words. Since each tweet was tokenized, lower-cased and stemmed in pre-processing part, we can simply extract those by using NLTK toolkit[8].

- Sentiment polarity lexicons.
  We use SentiWordNet lexicons developed by Baccianella et al. 9 to assign each word or gram in a tweet a sentiment score through its Part Of Speech Tag(POS-tagging) and corresponding sentisynet in lexicons. SentiWordNet is available from NLTK toolkit[8].

- Bags of Words.
  The Bag of Words is a model extract features by counting the number of times each word appears in a tweet.We use scikit-learn[10] package to extract a vocabulary with 3000 most frequent word. TF-IDF("Term Frequency, Inverse Document Frequency") is applied to score the importance of each word in a tweet by using scikit-learn package.

- Pre-trained word embeddings.
  Word2Vec[11] is one of the most popular word embedding algorithms to obtain dense vector representation of each term. Since our training data is not sufficient enough to train our own embedding, we use the publically available pre-trained model on tweets from Word2Vec. The Word2Vec model we choose was trained on 400 m tweets in English with 400-dimension representation for each word in vocabulary. Final vector representation of a tweet is calculated as an average of the sum of vectors of all the words in a tweet.

3 Experiment and Analysis

3.1 Experimental Result

Based on the features extracted as described above, we experiment with three classical machine learning algorithms form the scikit-learn[10] and xgboost package[12]: Boosting, Support Vector Machine, and Logistic Regression which have been reported having good performance in sentiment analysis. To train each classifier, all training data is being used and 5-fold cross-validation is performed to choose hyperparameter. Eventually, we use majority voting method among all predictions obtained above to get our final prediction.

More precisely, we use XGboost, Support Vector Machine with a linear kernel and Logistic Regression from scikit-learn package[10]. We evaluate all predictions through Accuracy, Precision, Recall and F1-Score computed through built-in function in scikit-learn package[10] as follows.

- XGboost

	Evaluation Scores			
	Accuracy	F1-score	Precision	Recall
TfIdf	0.6493	0.6492	0.6497	0.6499
Word2Vec	0.6193	0.6192	0.6198	0.6200
SentiScore	0.5502	0.5404	0.5533	0.5590

Table1:Result for XGboost classifier

- Support Vector Machine

	Evaluation Scores			
	Accuracy	F1-score	Precision	Recall
TfIdf	0.6232	0.6232	0.6234	0.6234
Word2Vec	0.6167	0.6167	0.6171	0.6171
SentiScore	0.5632	0.5511	0.5667	0.5758

Table2:Result for Linear SVM classifier with c=1

- Logistic Regression

	Evaluation Scores			
	Accuracy	F1-score	Precision	Recall
TfIdf	0.6389	0.6388	0.6393	0.6394
Word2Vec	0.6219	0.6217	0.6226	0.6232
SentiWord	0.5606	0.5480	0.5641	0.5732

Table3:Result for Logistic classifier wich c=1

- Majority Voting

Based on what we predict as described above, we use total prediction to do majority voting. Finally, we obtain a new prediction which achieves most votes among 9 previous predictions.

	Accuracy	F1-score	Precision	Recall
test score	0.6545	0.6535	0.6558	0.6584

Table4:Result after Majority Voting

3.2 Result Analysis

Generally, different classifiers have similar performance on the same extracted features, and XGboost performs comparatively best among three. For each classifier, the feature obtained from Bags of Words(TfIdf) gives the best result, and the features extracted through SentiWord library performs worst. Moreover, majority voting enhances the prediction performance among all classifiers and features. Comparing our best result obtained from the majority voting with Mathieu Clichs work[6], the prediction scores we achieved are more than 0.65 which are at least 0.05 higher than his result. Considering Clich trained classifiers on 120,000 data while we used 3000 only, we claim we achieve a better result comparatively.

3.3 Discussion and Future Work

The final result obtained was much improved from our initial trial, and we conclude the pre-processing of the raw data is the main reason. As Khodak's[13] claim, Twitter is a frequent source but with low-quality language and much noise. When we implemented few basic pre-processing techniques on raw data, there is too much information lost before extracting features and training a classifier. Our multiple pre-processing techniques play a significant role in increasing prediction scores.

We plan to explore more in extracting features which can keep the information of a sentences structure and extract lexicon-based features which can utilize the power of SentiWord in this case. On the other hand, we will implement deep learning algorithms like RNN and LSTM to see if the performance can be enhanced.

References

[1] Semeval-2018 task 3-irony detection in english tweets. URL https://competitions.codalab.org/competitions/17468#learn_the_details-data-annotation.

[2] Dictionary by merriam-webster: America's most-trusted online dictionary. URL https://www.merriam-webster.com/.

[3] Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. *Sarcasm as contrast between a positive sentiment and negative situation*, pages 704–714. Association for Computational Linguistics (ACL), 2013. ISBN 9781937284978.

[4] Ivan Habernal Tomás Ptácek and Jun Hong. Sarcasm detection on czech and english twitter. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, pages 213–223, Dublin, Ireland, August 2014.

[5] Roberto González-Ibáñez, Smaranda Muresan, and Nina Wacholder. Identifying sarcasm in twitter: A closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 581–586, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-88-6. URL http://dl.acm.org/citation.cfm?id=2002736.2002850.

[6] Mathieu Cliche. The sarcasm detector. URL http://www.thesarcasmdetector.com/about/.

[7] Cynthia Van Hee, Els Lefever, and Véronique Hoste. Guidelines for annotating irony in social media text, version 2.0. *Technical Report 16-01, LT3*.

[8] Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the COLING/ACL on Interactive Presentation Sessions*, COLING-ACL '06, pages 69–72, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. doi: 10.3115/1225403.1225421. URL https://doi.org/10.3115/1225403.1225421.

[9] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *in Proc. of LREC*, 2010.

[10] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:

2825–2830, November 2011. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1953048.2078195.

[11] Frderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van De Walle. Multimedia lab @ acl wnut ner shared task: Named entity recognition for twitter microposts using distributed word representations. *Proceedings of the Workshop on Noisy User-generated Text*, 2015. doi: 10.18653/v1/w15-4322.

[12] Scalable and flexible gradient boosting. URL http://xgboost.readthedocs.io/en/latest/.

[13] Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. A large self-annotated corpus for sarcasm. *CoRR*, abs/1704.05579, 2017. URL http://arxiv.org/abs/1704.05579.

[14] Andrea Esuli and Fabrizio Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *In LREC*, volume 6, 2006.