Detection of Sarcasm in Twitter

Arnav Kansal

CSCI-GA 3033-008: Statistical Natural Language Processing - Final Project

Introduction

Aim

- Given a text, classify it as *'sarcastic'* or *'non-sarcastic'*
- Extract features characterizing sarcasm in natural languages.

Evaluation Metric

Average F-Score

$$\boxed{F_{avg} = \frac{2 \times rec_{avg} \times prec_{avg}}{rec_{avg} + prec_{avg}}} \tag{1}$$

Data Source, Collection Method

Data source

- **Data mined from twitter using Twitter Search API.**

Collection

- **Queried for # sarcasm and # sarcastic tweets. (Labeled sarcastic)**
- **Queried general tweets of the day - already labeled sarcastic. (Labeled Non Sarcastic)**

Preprocessing

- Removed all '#' hashtags, '@' mentions and also urls/images from the tweets.
- Tokenized the data by using NLTK's tokenizer.

Sarcastic:
('if', 'evolution', 'was', 'real', 'why', 'do', 'we', 'still', 'have', 'dumbasses', '?')
Non Sarcastic:
('Great', 'eastern', 'restaurant', 'has', 'amazing', 'Dim', 'Sum', '🙂')

Total Data points	380093
Sarcasm	55468
Non Sarcasm	324625
Avg length(Data Pt)	13.5 words*

Table: Data Statistics

Bouzizi et. al. A pattern-based approach for sarcasm detection on twitter

Bouzizi et. al.

1. **Sentiment Related Features:** Four Featrues Extracted based on inconsistency between sentiments of words.
    1. **pw** - Number of positive words.
    2. **nw** - Number of negative words.
    3. **PW** - Number of highly emotional positive words.
    4. **NW** - Number of highly emotional negative words.
2. **Punctuation Related Features**
3. **Syntactic and Semantic or Pattern Related Features:** Parts of speech based grammar features.*

Joshi et. al.: Are word embedding-based features useful for sarcasm detection?

Joshi et. al.

- **Idea:**
  - **Use word embedding based features.**
  - **Compute cosine similarity scores of two word vectors.**
- **Unweighted Similarity Features:**
  - **Maximum score of most *similar* word pair**
  - **Minimum score of most *similar* word pair**
  - **Maximum score of most *dissimilar* word pair**
  - **Minimum score of most *dissimilar* word pair**
- **Distance-Weighted Similarity Features:**
  Instead of using raw cosine similarity scores of two words, weigh them inversely by the square of the distance between two words and proceed as before.

Vector embedding features contd.

Example showing feature calculation

Sentence: $word_1, word_2, word_3, ..., word_n$.
Content Word Sentence: $word_{a_1}, word_{a_2}, word_{a_3}, ..., word_{a_k}$

	$word_{a_1}$	$word_{a_2}$	...	$word_{an}$
$word_{a_1}$	-	0.4	...	0.2
$word_{a_2}$	0.7	-	...	0.6
...	...	...	-	...
$word_{an}$	0.2	0.3	...	-

Table: Similarity Table

	$word_{a_1}$	$word_{a_2}$	...	$word_{an}$
$word_{a_1}$	-	0.263	...	0.745
$word_{a_2}$	0.732	-	...	0.690
...	...	...	-	...
$word_{an}$	0.455	0.637	0.177	-

Table: Distance Weighted

$$S_{i,j} = cos(v[word_{a_i}], v[word_{a_j}])$$

$$S'_{i,j} = S_{i,j}/|(i-j)^2|$$

$$cos(u, v) = u.v/|u||v|$$

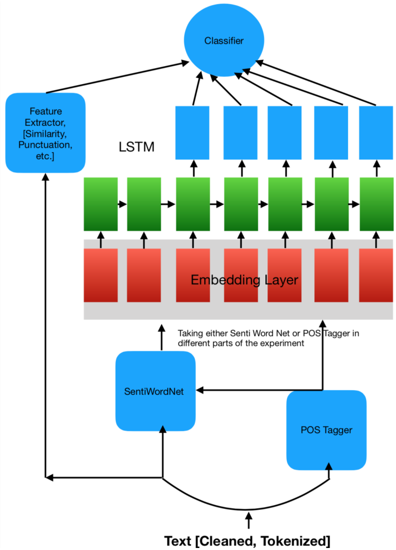
Proposed Solution

Motivation behind our contribution

- Learn more from syntactic features (POS tags) and their sequence.
  - Improve from the hard coded features from POS tags as done previously.
- Utilise sequence modeling for predicting sarcasm with sequence of sentiments.

Implementation

- Trained 2 LSTM's - 1 on sequence of sentiments, and one one sequence of POS Tags.
- Pulled the final layer output of the LSTM, augmented this output with precomputed baseline features and trained classifier over this new feature set.


Classifier
Feature Extractor, [Similarity, Punctuation, etc.]
LSTM
Embedding Layer
Taking either Senti Word Net or POS Tagger in different parts of the experiment
SentiWordNet
POS Tagger
Text [Cleaned, Tokenized]

Results

Results with Linear Kernel SVM Classifier (Sarcastic to Non-Sarcastic Test Class Ratio - 1 : 50)

Model / Features	Precision (%) S	Precision (%) N-S	Recall (%) S	Recall (%) N-S	F Score Average
Hacky Features	2.935	98.48	46.10	69.64	0.5405
Embeddings	2.068	98.25	55.58	47.60	0.508
Bouzizi et. al.	3.159	98.67	55.14	66.35	0.5540
Joshi et. al.	3.210	98.64	52.43	68.52	0.5529
Sentiment LSTM	3.423	99.03	70.14	60.60	0.5744
POS LSTM	4.132	99.18	72.02	66.73	0.5921
Sent + POS LSTM	4.322	99.22	73.37	67.66	0.5971
ALL FEATURES	4.674	99.24	72.93	70.38	0.6024

S $\rightarrow$ Sarcastic class

N-S $\rightarrow$ Non-Sarcastic class

Results

Results with XGBoost Classifier (Sarcastic to Non-Sarcastic Test Class Ratio - 1 : 50)

Model / Features	Precision (%) S	Precision (%) N-S	Recall (%) S	Recall (%) N-S	F Score Average
Hacky Features	3.163	98.78	60.86	62.91	0.5590
Embeddings	3.103	98.91	68.09	57.67	0.5632
Bouzizi et. al.	3.512	98.87	62.33	65.91	0.5693
Joshi et. al.	3.894	99.08	68.84	66.97	0.5860
Sentiment LSTM	3.481	99.03	69.88	61.43	0.5757
POS LSTM	4.212	99.17	71.56	67.60	0.5932
Sent + POS LSTM	4.455	99.24	73.46	68.63	0.5995
ALL FEATURES	4.818	99.29	74.80	70.58	0.6067

S $\rightarrow$ Sarcastic class

N-S $\rightarrow$ Non-Sarcastic class

Results

Results for various Sarcastic to Non-Sarcastic Test Ratios Classifier: Linear Kernel SVM

Sarcastic : Non-Sarcastic	Precision (%) S	Precision (%) N-S	Recall (%) S	Recall (%) N-S	F Score Average
1 : 50	4.674	99.24	72.93	70.38	0.6024
1 : 9	19.38	96.38	67.11	64.61	0.6436
1 : 3	44.66	88.91	77.82	66.85	0.6898
1 : 1	68.58	74.37	77.82	66.36	0.7128

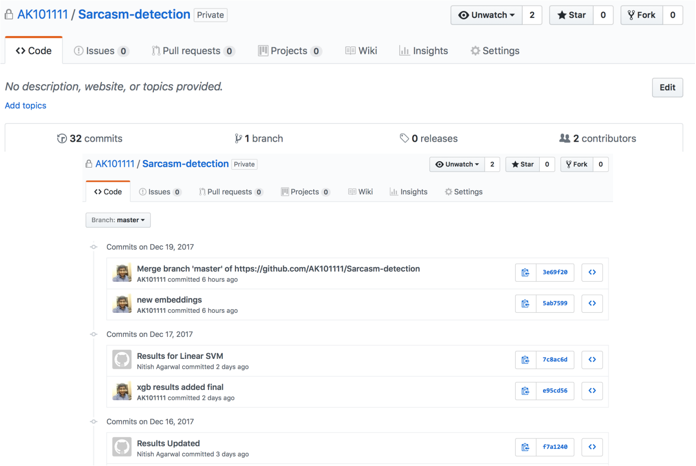
S $\rightarrow$ Sarcastic class

N-S $\rightarrow$ Non-Sarcastic class

REFERENCES:

- Mondher Bouazizi and Tomoaki Otsuki Ohtsuki. A pattern-based approach for sarcasm detection on twitter. IEEE Access, 4:5477–5488, 2016.
- Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. Are word embedding-based features useful for sarcasm detection? arXiv preprint arXiv:1610.00883, 2016.
- RobertoGonzález-Ibáñez, SmarandaMuresan, 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.
- Bird, Steven, Edward Loper and Ewan Klein (2009). Natural Language Processing with Python. O'Reilly Media Inc.
- Baccianella, S., Esuli, A. and Sebastiani, F., 2010, May. SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. In LREC (Vol. 10, pp. 2200-2204).
- Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N. and Smith, N.A., 2013. Improved part-of-speech tagging for online conversational text with word clusters. Association for Computational Linguistics.

Thank You!


AK101111 / Sarcasm-detection Private
Unwatch 2 Star 0 Fork 0
Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings
No description, website, or topics provided.
Edit
Add topics
32 commits
1 branch
0 releases
2 contributors
AK101111 / Sarcasm-detection Private
Unwatch 2 Star 0 Fork 0
Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings
Branch: master
Commits on Dec 19, 2017
Merge branch 'master' of https://github.com/AK101111/Sarcasm-detection
AK101111 committed 6 hours ago
3e69f20
new embeddings
AK101111 committed 6 hours ago
5ab7599
Commits on Dec 17, 2017
Results for Linear SVM
Nitish Agarwal committed 2 days ago
7c8ac6d
xgb results added final
AK101111 committed 2 days ago
e95cd56
Commits on Dec 16, 2017
Results Updated
Nitish Agarwal committed 3 days ago
f7a1240