Sarcasm Suite: A Browser-based Engine for Sarcasm Detection and Generation

Aditya Joshi[1,2,3] **Diptesh Kanojia**[1,2,3]
Pushpak Bhattacharyya[1] **Mark Carman**[2]
[1]Indian Institute of Technology Bombay, India
[2]Monash University, Australia
[3]IITB-Monash Research Academy, India
{adityaj, diptesh, pb}@cse.iitb.ac.in, mark.carman@monash.edu

Abstract

Sarcasm Suite is a browser-based engine that deploys five of our past papers in sarcasm detection and generation. The sarcasm detection modules use four kinds of incongruity: sentiment incongruity, semantic incongruity, historical context incongruity and conversational context incongruity. The sarcasm generation module is a chatbot that responds sarcastically to user input. With a visually appealing interface that indicates predictions using 'faces' of our co-authors from our past papers, Sarcasm Suite is our first demonstration of our work in computational sarcasm.

Introduction

Sarcasm detection gained attention from the sentiment analysis (SA) community for the challenges that sarcasm poses to typical SA systems. Several approaches to detect sarcasm have been reported Joshi, Bhattacharyya, and Carman (2016). This demonstration deploys five of our papers on sarcasm detection and generation. The demonstration titled '*Sarcasm Suite*' is a browser-based engine that allows users to test these systems. At the time of writing this paper, Sarcasm Suite is available at: https://www.cfilt.iitb.ac.in/sarcasmsuite/.

Sarcasm Suite deploys four sarcasm detection modules and one sarcasm generation module. To the best of our knowledge, this is the first demonstration related to computational sarcasm. The demonstration will prove to be interesting as well as useful. We expect Sarcasm Suite to be pushed to its limits. However, errors made by Sarcasm Suite will provide ideas for future research in computational sarcasm.

Related Work

Sarcasm Suite is a deployment of the following past works by three out of four authors of this paper:

1. In Joshi, Sharma, and Bhattacharyya (2015), we use sentiment incongruity to detect sarcasm. We experiment with two sets of features: explicit incongruity features (which capture number of sentiment flips, sentiment subsequence lengths, etc.), and implicit incongruity features (which are phrases with implicit sentiment).

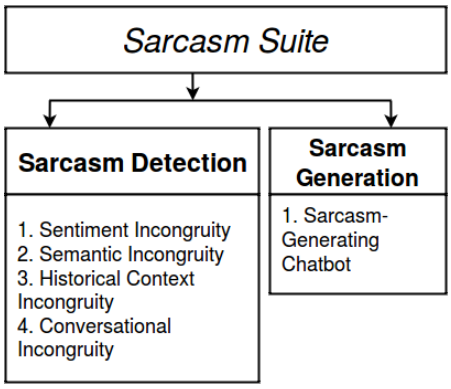

Figure 1: A Schematic Diagram of Sarcasm Suite

2. In Joshi et al. (2016b), we capture semantic incongruity via word embeddings in order to detect sarcasm. We experiment with two kinds of features: regular features and distance-weighted features.

3. In Khattri et al. (2015), we use an author's historical context in the form of their twitter timeline to detect sarcasm in their tweets. This is a rule-based technique that calculates surface sentiment of a tweet and compares it with the author's sentiment towards phrases in the tweet, in the past.

4. In Joshi et al. (2016a), we use conversational context using sequence labeling algorithms. The paper uses a dataset of transcripts from the TV show 'Friends'.

5. In Joshi et al. (2015), we present a sarcasm generation module that responds sarcastically to user input. The module implements eight sarcasm generators, each covering a peculiar form of sarcasm.

Architecture

Figure 1 shows a schematic diagram of Sarcasm Suite. There are a total of five modules in Sarcasm Suite. A video of Sarcasm Suite is attached. This section describes different aspects of Sarcasm Suite.

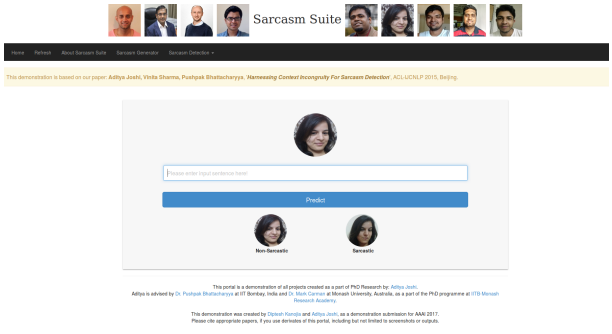

Figure 2: Snapshot of Sarcasm Suite

Interface Details

Figure 2 is a snapshot of Sarcasm Suite. The following are input/outputs of different modules of Sarcasm Suite:

1. **Sarcasm Generation**: Input is a user input while the output is an automatically generated sarcastic response.
2. **Sarcasm detection using sentiment incongruity**: Input is a sentence/set of sentences while the output is one among sarcastic and non-sarcastic. This is indicated by the face of the person. The legend indicates the sarcastic and non-sarcastic 'faces'.
3. **Sarcasm detection using semantic incongruity**: This is same as above.
4. **Sarcasm detection using conversational context incongruity**: The expected input is a conversation of the form of 'Speaker A: Text.*(newline)* Speaker B: Text.', etc. The output is a series of labels for each utterance by a speaker.
5. **Sarcasm detection using Historical Context Incongruity**: The input in this case is a sentence and a twitter id. We download 2000 recent tweets from the twitter ID (assuming it to be valid) and use it to obtain authors' historical context. This is then used to predict if the sentence is sarcastic.

The faces used in the respective pages are of co-authors of these papers. Sample inputs are provided.

Configuration details

For Joshi, Sharma, and Bhattacharyya (2015), we use features based on explicit and implicit incongruity. In case of Joshi et al. (2016b), we use the best feature configuration: Liebrecht, Kunneman, and van den Bosch (2015) with word vector-based features. In order to speed up the engine, we use the subset of Word2Vec that consists of common to the four kinds of word embeddings, as described in the original paper. Joshi et al. (2016a) is deployed using all features given in the original paper, and SVM-HMM to test the output. Khattri et al. (2015) and Joshi et al. (2015) are rule-based systems for sarcasm detection and generation respectively.

Implementation Details

We use server side scripting in PHP to create the engine. We have used Bootstrap CSS Framework released by Twitter Inc. for the front end, and jQuery-based AJAX requests that query the modules. The implementation of sarcasm generation was done in Java (Available at the github repository at: `https://github.com/adityajo/sarcasmbot/`). The implementation of sarcasm detection modules was done in Python, with calls machine learning tools: SVM-perf Joachims (2006), SVM-HMM Altun et al. (2003) and SVM-Light Joachims (1999).

Conclusion

Sarcasm Suite is a unique engine that demonstrates five of our past works related to computational sarcasm: four sarcasm detection approaches and one sarcasm generation approache. This is our first demonstration of these papers. The engine offers a variety of approaches that use sentiment flips, word vectors, historical context based on twitter timelines, etc. as cues for sarcasm detection. Sarcasm Suite will enable the research community working in sentiment analysis to identify challenges encountered in sarcasm detection.

References

Altun, Y.; Tsochantaridis, I.; Hofmann, T.; et al. 2003. Hidden markov support vector machines. In *ICML*, volume 3, 3–10.

Joachims, T. 1999. Svmlight: Support vector machine. *SVM-Light Support Vector Machine http://svmlight. joachims. org/, University of Dortmund* 19(4).

Joachims, T. 2006. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 217–226. ACM.

Joshi, A.; Kunchukuttan, A.; Bhattacharyya, P.; and Carman, M. J. 2015. Sarcasmbot: An open-source sarcasm-generation module for chatbots. *WISDOM Workshop at KDD*.

Joshi, A.; Tripathi, V.; Bhattacharyya, P.; and Carman, M. 2016a. Harnessing sequence labeling for sarcasm detection in dialogue from tv series friends. *CONLL 2016* 146.

Joshi, A.; Tripathi, V.; Patel, K.; Bhattacharyya, P.; and Carman, M. 2016b. Are word embedding-based features for sarcasm detection? In *EMNLP*.

Joshi, A.; Bhattacharyya, P.; and Carman, M. J. 2016. Automatic sarcasm detection: A survey. *arXiv preprint arXiv:1602.03426*.

Joshi, A.; Sharma, V.; and Bhattacharyya, P. 2015. Harnessing context incongruity for sarcasm detection. In *ACL-IJCNLP*, volume 2, 757–762.

Khattri, A.; Joshi, A.; Bhattacharyya, P.; and Carman, M. J. 2015. Your sentiment precedes you: Using an authors historical tweets to predict sarcasm. In *WASSA Workshop at EMNLP 2015*.

Liebrecht, C.; Kunneman, F.; and van den Bosch, A. 2015. The perfect solution for detecting sarcasm in tweets# not. In *WASSA Workshop*. New Brunswick, NJ: ACL.