Detecting Flaming in Social Media

Federico Montori
Computer Engineering Department
Bogazici University
Istanbul, TURKEY
federico.montori@boun.edu.tr

Mehmet Durna
Computer Engineering Department
Bogazici University
Istanbul, TURKEY
mehmet.durna@boun.edu.tr

Abstract

This document contains the results of our detection experiment in conversations from micro blogs. The main feature that we want to recognize amongst these conversations is the concept of flaming, which is completely attributable to a fight among internet users, often involving the use of profanity. Our system aims to collect data from Twitter, gather it into conversations and, applying a variety of features such as vulgarity and disagreement, decide whether these conversations contain flaming or not. As this is the first attempt in this sector, our aim is to provide a strong baseline for a future work.

1 Introduction

Social web and social platforms are currently the most powerful tool to share informations and to express opinions. Sometimes these opinions can be wrong, disrespectful or simply annoying to other users. In these cases, the situation can evolve in a serious discussion made out of comments on comments, as if in a real fight, but with tweets. Sometimes the consequences can be serious with one tweet that was carelessly posted, and a major flaming wave may run across the globe. (i.e. the tweet posted few weeks ago by Justine Sacco[1]). Those fights are what we call "flames", which mean "a hostile and insulting interaction between Internet users, often involving the use of profanity" according to Wikipedia , which are the central element of the project described in this report. What our project does is collecting conversations from twitter about some predetermined "risky topics" that we marked, and identifying which of conversations actually contain flaming. As the definition of flaming is somehow loose, we made use of a human annotator to evaluate our model.

Flaming is, most of the times, a good indicator of the most discussed topics on the internet and, combining this characteristic with what nowadays the most famous microblog website, Twitter, provides, we can claim that it will determine correctly on which arguments the attention of people is mainly focused. After observing tweets and making social semantics analysis on tweets, we came to the conclusion that people arguing on Twitter often don't know each other and are involved in those discussions according to the topics that are following. Furthermore, as we personally observed, a discussion often originates from a post tweeted by a journalistic institution.
The project code is available on github at `https://github.com/memedum90/TurkuazTurchese-`.

The system is designed as a pipeline of programs and tasks as shown in figure 1, until the state "Tweets classified". First of all we retrieve tweets and we store them as conversations, as explained in section 3. Secondly we pre-process the texts contained in the tweets in order to make them readable to our dictionaries and lexicons, as explained in section 4. Then, we apply our features on all the texts to assign them a flaming score to decide whether a conversation can be considered as a flame or not. To evaluate those features we made use of multiple tests on them, as explained in section 5.

2 Related Work

Since the topic we are treating in this paper is strongly related with sentiment analysis, we examined what kind of previous works there are on that sector. We started considering the work performed in (Jurafsky et al, 2013) on classifying posts in forums like Wikipedia and Stack Ex-

[1] http://www.express.co.uk/news/world/450183/IAC-fire-Justine-Sacco-after-racist-tweet-about-trip-to- South-Africa

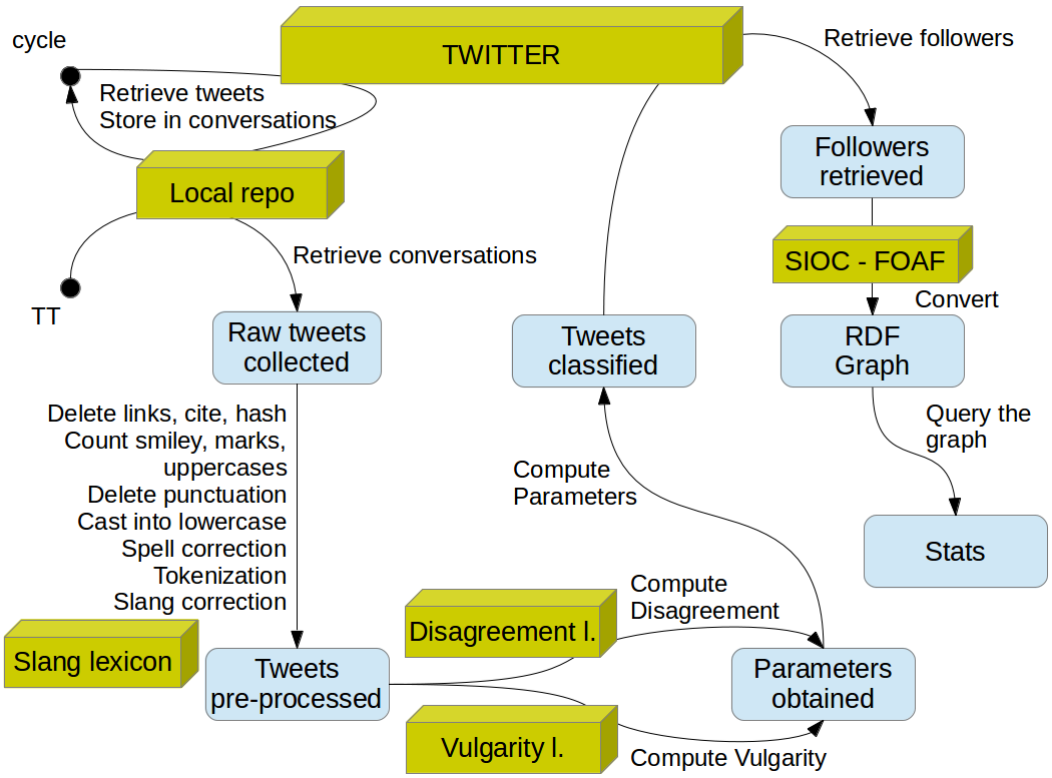

Figure 1: Overall design of our system. It contains also a SNA analysis in the end, which is not specifically part of this task and, therefore, not explained in this paper.

change on their politeness level using a manually annotated corpus. We initially used the same corpus as well, as explained in section 5, since politeness can be considered a valuable feature in identifying discussions. Another interesting work from ScienceDirect tries to detect features like friendship and other characteristic behavior in linguistic expressions (Ranganath et al, 2013). A group of researchers from the University of Michigan (Hassan et al, 2010; Hassan et al, 2012; Hassan et al, 2012) based their work on analyzing attitude in user conversations, often using lexicons and subjectivity analyzers from the MPQA project[2]. From those analysis they retrieved an oriented graph of relationships and they build systems capable of identifying subgroups and predict attitude among users. The latter is very similar to our task, despite the fact that flaming is a different concept, since it includes a certain amount of vulgarity of opposition among the participants of a discussion.

To our knowledge this is the very first attempt to try to analyze flames from social networks, therefore we were not able to start from a strong baseline, neither to have some relevant previously annotated data.

3 Data Retrieval

We retrieved data from Twitter simply using the tweepy[3] python library, so we didn't make use of another ready-to-use tool. The reason of this choice is that, basically, we needed to retrieve conversations, and there is no tool which allows us to do it.

We chose 15 different topics for our research which are more likely to contain a valuable percentage of flames and debates and less characterized with sarcasm: abortion, church, communism, erdogan, euthanasia, feminism, gaymarriage, gender, holocaust, islamist, jesus, massacre, morsi, muslim, racism, religion, syria, terrorist. The data retrieved has a form of object with multiple fields from which we extracted and renamed the ones we needed:

- **username:** The name of the user who posted, called screen name from the Twitter API

[2]http://mpqa.cs.pitt.edu/

[3]https://pypi.python.org/pypi/tweepy/

- **topic:** The keyword we user for the research.
- **user_id:** The ID of the user.
- **fvt_cnt:** The number of likes that tweet collected.
- **text:** The plain text of the tweet.
- **reply_to:** The tweet ID for which this tweet is a reply.
- **rep:** Manually added field. The value is 1 if this tweet is a reply, 0 otherwise.
- **id:** Unique ID of the tweet.
- **rtw_cnt:** Number of retweets of the tweet.

Basically, to retrieve conversations, we get some tweets making a search query using tweepy to Twitter API using the topics given. Every time we obtain a tweet which is a response, we look to the "reply_to" field, if it has another tweet id, we make the "rep" field of the tweet 1 and extract that tweet too making another query, then we proceed this way until we catch the head of the conversation and we put 0 on the field "rep". If "reply_to" is None, the tweet is ignored. This way is easy to collect whole conversations because we have to parse out tweet list and just store a new conversation when we reach the 0. Of course, due to the twitter API limit, we encountered some problems:

1. With a single keyword search we can retrieve about 300 tweets at a time and our program has periodically to sleep until the next time window available.
2. Is given a conversation made out of 3 tweets and structured as follows: $A \rightarrow B \rightarrow C$ (C is a reply to B, which is a reply to A). If with our keyword we extract B, we can retrieve A, but there's no way in which we can retrieve C.

To perform further analysis in future work, like Social Network Analysis, we needed also another relationship among users, especially we want to know which user follows each other. In the first place, we tried to extract all followers and friends of all the authors of at least one post in out dataset, but again, Twitter rate limits did not let us do it. The only solution we could adopt was checking for every couple of user in the same conversation if one user is following the other.

4 Data Preprocessing

To perform our analysis we had to work on the plain text we retrieved, doing some preprocessing. To achieve this goal, we worked on our data performing the following steps:

- We removed from the plain text every hashtag (word beginning with #), every citation (word beginning with @) and every hyperlink (word beginning with http://).
- We casted everything into lowercase to make our lexicons work homogeneously.
- We removed all the punctuation.
- We normalized every slang expression, replacing it with the actual translation. To achieve this, we obtained a huge list of slang terms with the respective meaning from an online vocabulary[4] and we manually modified that.
- We checked if the post is actually in english using the guess_language[5] library, which has an acceptable accuracy. We could not use the language field specified in every tweet because it denotes the mother language of the user, not necessarily the language used in the tweet.
- We performed spell correction of a word, checking if the word is in the english dictionary using enchant[6] and, if not, substituting it with the closest word in the dictionary if the edit distance is less than 2.

5 Data Features

According to the work performed by (Jurafsky et al, 2013), we considered to use the politeness corpus provided in their repositories[7] as a strong baseline, but we figured out that unfortunately it was not possible because the tweets' texts are too short to perform a proper Naive Bayes evaluation. So, both to have a good baseline for future work and to be able to evaluate our performances, we manually annotated more than 5000 tweets, classifying them into two categories (flames or not).

[4] http://noslang.com
[5] https://pypi.python.org/pypi/guess-language
[6] https://pypi.python.org/pypi/pyenchant
[7] http://www.mpi-sws.org/ cristian/Politeness.html

This dataset became our gold standard on which we performed our tests on the features.

The dataset is currently available in our project repository.

The features that we considered were:

1. Count of the number of uppercases in a tweet (before casting everything into lowercase). Normally a user wants to say something strongly and angrily is more likely to use uppercases for entire words or even sentences.

2. Count of the number of question and exclamation marks. In this case, this kind of punctutation is normally repeated when a question or an exclamation is intended to be stronger by its author.

3. Count of smiley, bad or good, they are always giving a sense of joke to the whole sentence. This is the easiest way to identify sarcasm in a sentence. As we will see in this section, sarcasm was one of the most frequent causes of false positives.

4. Count of vulgar words, as they are the main characteristic of flames. Normally they can be contained even in other kinds of conversations, especially ironic ones. For such task, we manually modified lexicon that was previously constructed[8]. Together with the previous features, we used the latter as characterizing our first baseline, from which we extracted statistics as shown at the end of this section.

5. Count of co-occurrences of vulgar words and second person singular pronouns like "you", "you're", "you'll". This is more likely to be present in flames rather than in other kind of conversations, because it denotes insults directed to the interlocutor. In particular, we used this feature replacing the pure count of vulgar words to perform some improvements on the baseline.

6. Count of expressions of disagreement and agreement. We annotated a lexicon by hand, present in our project repository, having agreement and disagreement expressions. Of course, expressions of disagreement like "that's not true" are indicators of debate, while the expressions of agreement are exactly the opposite. This feature was not present in our first baseline and it's again an improvement.

7. Polarity of the tweets. Lots and lots of works have been done on the sentiment analysis task, in particular, we are interested in classifying a tweet as positive or negative, what we call polarity. In our case, we made use of an online tool called Sentiment140[9] to calculate every tweet's polarity and report it directly in our corpus on each tweet under the field "polarity", denoted by a discrete value between 1 and -1. (1 meaning negative, 0 meaning neutral, -1 meaning positive).

6 Flaming Detection

As mentioned, we determined some features to be relevant and upon calculating the data as described above, we experimented with different approaches.

First of all, we found some coefficients by testing and trial and error. We multiplied obtained features by these coefficients and then use a predetermined threshold, again by observation. Coefficients used can be seen in Table 2

In the second run, we made some normalizations according to the number of tweets in the conversation and we calculate a score using some weights assigned to the features. And we selected three features to work with which are disagreement, vulgarity and polarity.Then we made an analysis on the manually annotated data set to get some ideas about the features, we noted the scores features for flaming and non flaming. The values of the features for flaming and non-flaming conversations can be seen in Table 1. Using these value, we take averages of the value of the same feature for flaming and non flaming and sum them to be our new threshold.

6.1 Baseline

In our baseline, we made use of five features which are:

- Count of the number of uppercases
- Count of the number of question and exclamation marks.

[8] http://www.cs.cmu.edu/ biglou/resources/bad-words.txt

[9] http://www.sentiment140.com/

Features	Flames	Non-Flames
Uppercases	5.442	4.217
Marks	0.383	0.436
Vulgarity	0.411	0.152
Good Smileys	0.031	0.042
Bad Smileys	0.086	0.195
Disagreement	0.110	0.036
Polarity	-0.026	0.036

Table 1: Average Scores

Features	Baseline	Our Method
Uppercases	0.1	0.1
Marks	2.0	2.0
Vulgarity	4	
Good Smileys	-10	-10
Bad Smileys	-0.5	-0.5
Insults		4
Disagreement		8
Polarity		2

Table 2: Assigned Weights.

- Count of good smileys
- Count of bad smileys
- Count of vulgar words

We make use of these features by multiplying it with some coefficients and summing them to compare it with a determined threshold. The coefficients used for the baseline can be seen in Table 2.

6.2 Our Method

We added two more features, and one improved feature to our baseline to improve it.

- Count of co-occurrences of vulgar words and second person singular pronouns.
- Count of expressions of disagreement and agreement.
- Polarity of the tweets.

We have two experiments with this method as described above. First is using the same approach in baseline but with more and improved features, and using different coefficients that are observed and found by testing, and trial and error.The coefficients used for the proposed method can be seen in Table 2. Second one is setting up a threshold using statistics of a previous run averaging the feature scores described in more detail above.

6.3 Results and Discussion

The results we got for baseline is as follows:

- For flaming we achieved a 0.137 precision, a 0.167 recall and a F-score of 0.15.
- For non-flaming we achieved a 0.891 precision, a 0.867 recall and a F-score of 0.879.

With our first approach, applying the proper weights to our features and running our tool on another manually annotated dataset we achieved the following results(Lower weights are given to the features that doesn't make a big difference.:

- For flaming we achieved a 0.241 precision, a 0.433 recall and a F-score of 0.31.
- For non-flaming we achieved a 0.92 precision, a 0.826 recall and a F-score of 0.87.

With our second approach, determining a threshold using the values of a previous run and just summing up the feature values as they were, the results were as follows.

- For flaming we achieved a 0.198 precision, a 0.45 recall and a F-score of 0.276.
- For non-flaming we achieved a 0.917 precision, a 0.769 recall and a F-score of 0.836.

So we can say we improved our baseline and our first proposed method is better than the second method.

By our experience we are aware of some well-known bugs. Our method can reach false positives easily when dealing with sarcasm (very difficult to handle), with excess of vulgarity in a common invective (a group of people angry at the same entity) and with too many uppercases in short conversations. On the other hand we can reach false negatives when dealing with very long conversations with very short replies.

7 Conclusions and future work

Due to our dataset, some of our results may seem disappointing, but the most relevant concept is that we developed a baseline for a solid future work. Annotating manually a huge amount of tweets, we provided a corpus that may be used to perform a Naive Bayes Classification to contribute to our analysis.

References

Hassan, A., Qazvinian, V., Radev, D.A. 2010. *What's with the Attitude? Identifying Sentences with Attitude in Online Discussions*. EMNLP 2010

Abu-Jbara, A., Hassan, A., Radev, D. 2012. *AttitudeMiner: Mining Attitude from Online Discussions*. NAACL 2012

Abu-Jbara, A., Hassan, A., Radev, D. 2012. *Detecting Subgroups in Online Discussions by Modeling Positive and Negative Relations among Participants*. EMNLP 2012

Danescu-Niculescu-Mizil, C., Sudhof, M., Jurafsky, D., Leskovec, J., Potts, C. 2013. *A computational approach to politeness with application to social factors*. ACL 2013

Ranganath, R., Jurafsky, D., McFarland, 2013. *Detecting friendly, flirtatious, awkward, and assertive speech in speed-dates*. SciVerse ScienceDirect 2013