Computational Linguistics -1
Project - 2 Report

Hate speech detection in Hindi

Vamshi Krishna , Vanshpreet Singh Kohli , Nukit Tailor

We use the paper linked here as reference to build a tool that searches a Hindi corpus (in this case, a collection of tweets - the corpus we used can be found here) for hate speech and then categorizes the tweets into “weakly hateful”, “strongly hateful” or “no hate” based on the amount of hate content present. The model we built uses a rule-based approach to use sentiment analysis techniques, especially subjectivity analysis, to identify and rate the polarity of sentiment expressions. We begin by shortening the corpus that we shall operate upon by removing objective sentences. Then using subjectivity, sentiment analysis of words and lexical features of hate speech (extracted from the corpus so as to maintain relevance with it), we create a lexicon that we use to build a classifier for hate speech. We then put the hate speech detector to the test, showing practical application in a real-world web discourse.

Introduction

Hate speech is extremely common on platforms such as Twitter, Facebook, comments sections and even blogs or biased online publications, and even though things like profanity filters exist, they only filter out obscenities and swear words. We find that the vast majority of hate speech consists of veiled attacks, or otherwise uses words that would in other contexts be completely innocuous but are being used to attack an individual or group. Most of even this is contextualized, so to know the context of each of the sentences and then gauge the hatefulness to a near-perfect accuracy would require some knowledge of the topic under discussion which is beyond the scope of our rule-based algorithm. However, we find we can achieve a healthy precision in not just the detection of hate speech, but also its segregation into weakly or strongly hateful speech.

The corpus

Political and social issues are heavily polarizing, and people seem to have a lot of emotion attached to their ideologies, which causes a lot of rudeness and hate speech in threads (on, say, twitter) that discuss political issues. Thus, our corpus deals with tweets of mostly a political or social nature and finds a large quantity of hate speech therein. As you will find if you head over to the corpus (linked above), it

has been pre-segregated into sub-categories that define the hatefulness of the tweets. This has been done using a comma-separated-values (csv) file where each row is made up of an index number, the tweet itself, and its “hatefulness” category. The categories are:

- Non-hostile
- Fake
- Defamation
- Offensive
- Hate

Naturally, not all of them are mutually disjoint, hence the corpus enlists various categories for some tweets instead of just one. For example, a tweet that’s fake could also be defamatory as well as offensive. It is important to note here that all the tweets marked “non-hostile” are devoid of any malintent and thus do not fall into any of the other categories.

The algorithm

Our approach consists of three major steps.

- Subjectivity analysis
- Building a hate speech lexicon
- Identifying theme-based nouns.

Let us look at the algorithm in more detail to understand what these steps signify.

Subjectivity analysis

Subjectivity analysis helps us whittle down the corpus, as only the tweets that are subjective shall have hate speech instead of, say, facts. Thus, it is of great help to look for objective tweets and mark them already as “not hate”. To do this, we must first give scores for subjectivity and filter out the objective sentences. We use SentiWordNet for this, using it to assign positive and negative scores to each word.

Then, for each key in the now-created SUBJCLUE lexicon, we match it to its occurrence in our dataset and assign negative and positive scores accordingly. We then use the scores to determine the total score for that tweet/sentence. We used a variety of values as the limiting values that make a tweet worth investigating further, and decided that all tweets with a score above 1.0 or below -0.5 were subjective enough to be heavy contributors towards hate speech. The lower limit is numerically lesser than the upper one because we found that the sentences marked negatively by the word sentiment analysis were more likely to contain hate speech.

Building a lexicon for hate speech

Now that we have the subjective sentences segregated, we can start building a lexicon that has words which help us more accurately identify hate speech. We first collect all words from the SUBJCLUE lexicon which have a total score less than -0.25 and them to a list of strong negative words and similarly collect weakly negative words which are negative but greater than –0.25. These lists help us detect negative polarity and hate content in the sentences or tweets. We then use an initial list of seed verbs, consisting of verbs manually selected after examining the verbs in the dataset which we believed to carry a hateful connotation or to be used mostly for spiteful remarks which were not included in the SUBJCLUE. We then use a SYNSET to find all the possible synonyms of these verbs, and add all of those to a list of hateful verbs. After having this hate-verb list built, we check which of those hate-verbs occur in the hate corpus and if they do then we add them into a final hate-verbs lexicon.

Theme-based nouns

Now that we have lexicons for negative words in general and verbs that are deemed hateful, we need to ensure that they are indeed presented with nouns that are relevant to the topic of discussion - such as swear words and theme-dependant nouns (eg. politically significant words) - to ensure that they are used in relevant messages that spew hatred. For this, we first segregate the corpus into noun phrases and note the most frequent NPs that are used in tweets marked as hateful. We add a few dozen such recurrent nouns into a text file for later use while checking for hate speech, for which we add them to a list.

Final testing

We now have all the lexicons we need, all having used rule-based approaches, to finally use on our corpus and check for hate speech. We wish to gauge the role played by each lexicon in the final result to see how much accuracy is added to the hate speech detection by each part of the process. Thus, instead of running the complete algorithm all at once, we split it into parts to see which lexicons make a significant difference to the end result. This is achieved simply by commenting out the use of certain lexicons while running the hate speech detector. We then run this algorithm over the code to get a list of the number of strongly and weakly negative words, the number of items corresponding to the hate lexicon and the number of items corresponding to the lexicon of theme-based nouns. Whenever we wish to negate the role played by one of the lexicons, we comment out the lines of code that correspond to the others. We use these counters to judge the hatefulness of the sentences as judged by various combinations of metrics.

We use a well defined criterion to segregate the tweets by the level of hatefulness:

- If the tweet has two or more words qualified as strongly negative, it is marked as strongly hateful. If it has one strongly negative word with non-zero hate-verbs or non-zero theme-based nouns, it is marked as strongly hateful. If it has at least one word each from our hate-verbs lexicon and themed nouns, it is likewise marked as strongly hateful.
- If it has one word that is strongly negative, but no other word from our other lexicons, it is weakly hateful. If it contains one weakly hateful word with a theme based noun, it is weakly hateful. If it contains neither of the above but contains a hate-verb, it is also weakly hateful.
- If the tweet satisfies neither of these criteria, it is judged as non-hateful.

Final results

Now that we have defined the criteria for which tweets are judged hateful or otherwise, we can finally test the algorithm. To do this, we first need to manually define what the objective results for hatefulness must look like so that we may then compare them to the results obtained from our various algorithms. We do this by using the tags that are a part of the corpus, which has already been judged for hate speech. Instead of using the criteria the corpus uses (non-hostile, defamation, fake, hate, offensive), we have used a simpler criterion of no hate, weakly hateful and strongly hateful in our project. Hence, we need a connection between the two. For this, we consider that:

- All the non-hostile tweets correspond to “not hate”.
- All the tweets marked as fake or defamatory correspond to “weakly hateful”.
- All the tweets marked as hate, offensive or a combination of one or more tags (which, as mentioned before, cannot contain non-hostile) are deemed to correspond to “strongly hateful”.

Thus, when we finally judge the precision (ratio of true positives to that of the total positives, i.e. true+false positives), recall (ratio of true positives to the sum of true positives and false negatives) and F-score (harmonic mean of precision and recall) of our algorithm, we shall consider the aforementioned points as the objectively correct qualification that we shall match against. Now, let us move onto the final results we arrived at. First, we take a look at the results that we obtained without the subjectivity analysis and then, at those that we obtained with the subjectivity being taken into account. Also, as we had mentioned previously, we wished to check for the effect had on the final results by the various parameters used to quantify the hate speech, so each result table will have different rows that signify the parameters used while getting the result.

Here are the results obtained without using subjectivity clues:

Feature sets	Precision (%)	Recall (%)	F-score (%)
Semantic (weakly/ strongly -ve)	39.33	77.99	52.29
Semantic + hate-verbs	39.35	78.03	52.32
Semantic + hate-verbs + nouns	42.54	84.35	56.55

As you can see, the hate-verbs alone do not make too much of a difference to the accuracy of our model, but adding the themed-noun lexicon improves it relatively significantly.

Here are the results obtained with subjectivity clues used:

Feature sets	Precision (%)	Recall (%)	F-score (%)
Semantic (weakly/ strongly -ve)	45.37	84.98	59.16
Semantic + hate-verbs	45.49	85.02	59.27
Semantic + hate-verbs + nouns	48.58	91.84	63.54

Clearly, using subjectivity clues makes a very significant improvement to the accuracy of our hate-speech detector (nearly 7% increase in the F-score), which shows the importance of checking for contextual nouns while searching for and detecting hate speech.

Another noteworthy factor is how the Recall is much higher than the precision; much higher, in fact, than the Recall value of the original project we referred to even though the precision we get is relatively lower. We argue that this is due to our algorithm being very efficient with actually making sure that the hate-speeches are tagged correctly, as are the innocuous tweets. Thus, it leaves few false negatives (i.e. misses) behind. There are however, relatively more false positives which takes away from the precision. We argue that this is caused by our objective criterion (as described above) not being particularly objective at separating weak hate from

strong hate because of the misalignment between criteria used by the corpus to tag hate speeches, and the criteria we ourselves use to segregate them. However, our algorithm compares favourably to the original in terms of segregating non-hate tweets from hateful tweets (about 66% precision in segregating non-hate) because of the criteria being more objective.

Conclusion

The detection of hate speech in languages such as Hindi with wide userbases is getting increasingly relevant in this digital age. Our model uses a rule-based approach to look for lexical patterns that allow us to detect and quantify hate speech to reasonable precision. An important function here was played by the analysis of subjectivity in our algorithm, accounting for which made a significant improvement in the accuracy of hate-speech detection.