839 Project Stage 1: Institution and Organization Extraction from Fake News

Daniel K. Griffin
Department of Computer Science
University of Wisconsin Madison
dgriffin5@wisc.edu

Yudhister Satija
Department of Computer Science
University of Wisconsin Madison
ysatija@wisc.edu

Mitali Rawat
Department of Computer Science
University of Wisconsin Madison
mitali.rawat@cs.wisc.edu

1 Introduction

We have chosen to mark institutions, organization names etc as our entities. After splitting a line on white spaces and some special characters for easy processing, we mark them using tags <[> (opening tag) and <]> (closing tag) in the "Markedup data" directory containing 300+ documents. This puts each word is on a different line in the file. The folder "Markedup data" is split into 200 and 100 files randomly and we name them "training set" and "test set" respectively. We then create a set of examples by considering sequences of 8 or less valid English words and applying a few preprocessing and pruning rules. For each of the examples, we then create a feature vector using a defined set of features. We start our process by performing cross-validation on the training set. We use the classifiers mentioned and additionally an AdaBoost classifier. We pick the best among these (Random Forest), train it on our entire training set I and test it on the test set J.

2 The Quick List

This section provides a quick reference sheet of our accomplishments, matching the list of what the .pdf file should contain on the course project web page. This section is meant simply to provide a quick synopsis of the required information to make evaluation easier.

1. **The names of all team members**: Daniel Griffin, Yudhister Satija, Mitali Rawat
2. **Entity type that we have decided to extract**: We have decided to try to extract institutions or organizations. This includes political parties, universities, companies, news organizations, websites, and other directly named social organizations. Examples include things like "Wikileaks", "Hillary Clinton Campaign", "FBI", "New York Times News", "Facebook", and "Twitter.com". We have not included things like places, instruments, or diseases. Non-marked examples include "ADHD", "USA", "California", "Epi Pen".
3. **Total Number Of Marked Entities**: We counted 2320 entities. To do this count, we used the command "cat * | grep -c "<\[>""
4. **Number of documents in set I, the number of mentions in set I**: We counted 1624 entities in the training set, with 200 files in the training set.
5. **Number of documents in set J, the number of mentions in set J**: We counted 696 entities in the test set, with 100 files in the test set.
6. **Type of the classifier that you selected after performing cross validation on set I *the first time*, and the precision, recall, F1 of this classifier (on set I).**: We decided to use

the RandomForrest model initially, and got an average precision of 0.92131 with std. dev. 0.0161, and an average recall of 0.6105) with std. dev. 0.0133. Our model had an average F1 score of F1 0.6813 with std. dev. 0.0188.

7. **Type of the classifier that you have finally settled on *before* the rule-based post-processing step, and the precision, recall, F1 of this classifier (on set J).**: We decided to use the RandomForrest model initially, and got a precision of **0.8405**, a recall of **0.6067**, and an F1 score of **0.7047**.

8. **Post processing rules**: If a predicted instance is a single word and exists in the dictionary, then we classify it as negative. This helps to reduce the false positives due to words that have all of their characters capitalized.

9. **Final precision recall and F1 of classifier**: After post processing, we get precision of **0.8753**, and recall of **0.6040**. Which gives F1 score of **0.7147**

10. **If you have not reached precision of at least 90% and recall of at least 60%, provide a discussion on why, and what else can you possibly do to improve the accuracy**: So we are just shy of our desired precision metric. There may be many reasons for this. One reason could be that the test set false positives should have actually been marked as true positives. There were many instances such as this in the training set that we had to fix. Other methods for us to try to improve the precision would be to try to remove the false positives from the training set related to three letter acronyms. This kind of false positive seems to be the most prevalent, and doesn't.
    Majority of false positives were due to sub-sequence of actual positive being marked by classifier as positive. A partial match detector could be used in post processing to clean out these false positives. But due to high computation requirement (match each false positive against all positives labels) this was not included in our scripts.

3 The Pipeline

3.1 3 Stage Process

1. Split text files to words on separate lines
    - We split our entire marked up data on white spaces and punctuations (except apostrophe). We did this to be able to easily set the start word index, which would just be the line number of the word in the document. We can also thus easily lookup the word from the document id and the start index.
2. Create examples (tuples)
    - Next we created tuples for the examples we would be passing to our learning algorithms.
    - We first pick continuous sequence of 1 to 20 lines from a marked up data. We label this tuple +ve if it begins and ends with our tags but doesn't contain any tag in between.
    - We also save extra information with this tuple using preprocessing rules, like:
        - the document name,
        - the start index of the tuple in file
        - the end index of the tuple in the file
        - the stripped version of the raw string after removing ALL punctuation
        - the number of words in the stripped string
    - With this on set I, we were getting close to 420k examples. While the count of +ve examples was just 1.6k, this was a huge skew.
    - Hence, we included some pruning rules. We remove a tuple if
        - the stripped string is empty or longer than 8 words;
        - the raw string contains bad punctuation marks like "%", "[" or ">";
        - the raw string begins with a period;
        - there is only a single character left in the stripped string;
        - the string contains the end of a sentence, but not examples like ".com";
        - the string contains any of the bad keywords which should not be in a institution name.

- After this pruning we reduced the no. of -ve examples by almost half. We were left with some 230k examples of which 1.6k were +ve

3. We then pass this set of tuples to a featurization script. The features are described below. We thus get 2 csv files (pandas dataFrame). One that contains our tuples including raw string and extra preprocessed information. Another contains the features and the labels for each of the tuples. There is one to one mapping between the two by the index of data. The features file is the one that is passed to the learining algorithms.
4. Finally, we learn a model on the featurized training instances. Our training set script has 3 capabilities. The first learns a suite of classifiers on the training set to determine the best model performance. The second runs a 5-fold cross validation on the training set for a decision tree and random forrest classifier, calculates the mean and standard deviation for both precision and recall, and saves the results to a set of text files. The third reads the precision and recall results from the previous text files, plots a histogram of precision, and outputs false positive instances.

4 The Features and Classifiers

4.1 Features

We use 20 features described as follows:

F0. "[The]" occurs 1 or two lines before string.
F1. Number of capitol Letters.
F2. Verb occurs 1 or two lines after the string.
F3. Total character length
F4. Total number of words
F5. Number of capitol letters before the string.
F6. Number of capitol letters in line after this string.
F7. "on" comes before
F8. "called" comes before
F9. "they" comes after
F10. .?! comes in the middle of and entry
F11. Number of "."s
F12. "," is in the raw string
F13. "," is in the first or last raw string position
F14. "." is in the first or last raw string position.
F15. "as", "a", "an" is in the raw string.
F16. The faction of the number of words where only the first character is capitalized to all words.
F17. The rawString has a Single capitalized word after it.
F18. Contains a keyword.
F19. fraction of capital letters to wordCount

4.2 Classifiers

We compared a number of classifiers including decision trees, random forrests, support vector machines, linear regression, logistic regression, and AdaBoost. For the most part, decision trees and random forrests worked the best for our dataset. AdaBoost also sometimes worked well, but its performance during cross validation was spurious. This intuitively makes sense because most of the features we used for our models are very analogous to rules. Since decision trees and random forrests naturally encode their state space with chains of rules, it makes sense that they would perform the best with features that are essentially rules. We further tuned our model performance by using the scikit-learn random forrest model's "predict_proba()" function. This function returns a confidence score for belonging to a class, rather than a class label. Using this, we tuned our model to reduce our recall, and improve our precision. This probability threshold also seems to have worked well for the test set.

5 Ways to Improve

There are many ways the team discussed trying to improve performance. The first was to try to change our definition of a successful extraction. Currently, we say that only entities that are fully extracted are correctly classified, and all partial extractions are negative. However, we found that many of our false positives are actually extractions that are contained within large entities that have been tagged in the training set (we think there is a skewed bias towards smaller entities). A partial classification would remove many of the false positives of this form that we are seeing.
We also have a ton of different features that we haven't yet tried to include in our model such as:

1. "and" comes before or after
2. "from" comes before
3. Contains "of", or "of" as second word.
4. Contains "for"
5. "in" comes before, as a feature for indicating "not an institution", as it is likely a place.
6. "near", "at" comes before or after, as a feature for indicating "not an institution", as it is likely a place.
7. "is" comes after.
8. "according to" comes before or after
9. "officials" comes after (Immediately or two to three places after)
10. "members of" comes before
11. "member", "director", "chief", "sources", "agent", "chairman" or other similar person designations comes directly after entity.
12. "reported", "notes", "reviewed" and similar words comes after (But not "said", as these are people)
13. "by" comes before (This may cause some issues though, as it could be cause names of people to occur)
14. "named", "entity"
15. "group", "organization", "society", "founded", "formed", "foundation", "community", "agency", "center" "movement", "newspaper" in or nearby
16. ".org", ".com", ".co", "ltd", "partners", "company", "project", "corporation", "website", "
17. nonprofit", "center" before or after.
18. Not a city, state, or country (Use a corpus for this)

We could further try to improve our model's performance by using out-of-bag feature importance from our random forrest to gauge which features are the most important, and which features are redundant, thus helping to reduce our model's dimensionality. Following along the line of iterative improvement, we could try to run outlier detection models, or explicitly create bagged models on disparate input spaces.