Natural Language Processing: Generating a Summary of Flood Disasters

Acanfora, Joe
joe5158@vt.edu

Evangelista, Marc
marcato@vt.edu

Keimig, David
davidk47@vt.edu

Su, Myron
myronsu@vt.edu

December 11, 2014

Abstract

This document describes how our team learned about, and applied natural language processing tools to create a summary of flood events based on large collections of news articles. The body of the document covers topics including

- Solr
- Machine Learning
- Frequency Analysis
- Regular Expressions
- Geocoding
- Part of Speech Tagging
- Chunking / Chinking
- Named Entity Recongition
- Latent Dirichlet Allocation
- Natural Language Generation
- Clustering
- Hadoop

This document does not give an in-depth theoretically explanation of how these tools work but rather focuses on our application of the tools to our particular data sets and to the problem of generating natural language summaries. *Floods.*

Contents

1 Introduction

This document is a cumulative and final report of the efforts taken by the four authors while they were enrolled in the Virginia Tech Computer Science course 4984; a special study on computation linguistics of which we were enrolled in the fall semester of 2014. The intent of the report is two fold: first to demonstrate our results and our knowledge of the tools that helped us summarize large unstructured datasets into machine generated human readable summaries. And second, to help future readers understand, replicate and potentially adapt our work to other datasets and other problems. We begin by discussing the sources sizes and cleanliness of our data before moving into a practical analysis of the tools we explored and used to create our results. At the end of the document we conclude by discussing what we have learned over course of the semester and the strengths and weaknesses of our results.

2 Solr

Solr is an open source enterprise search platform from the Apache Lucene project. Its major features include full-text search, hit highlighting, faceted search, dynamic clustering, database integration, and rich document handling. Providing distributed search and index replication, Solr is highly scalable and is the most popular enterprise search engine with NoSQL features. For this project we had access to a Solr instance gave us searchable access to our copora.

2.1 Corpus: ClassEvent (Islip)

The ClassEvent collection is a set of 11 files that contain news articles relating to a flooding event that took place in New York State in 2011. The files in the collection all strongly related to the event, and were mostly devoid of erroneous text and external links.

2.2 Corpus: YourSmall (China Flood 2011)

YourSmall is a collection of 500 documents pertaining to the floods that occurred from June to September 2011 in the central and southern parts of the People's Republic of China. The documents are news article containing phrases pertaining to 'china' and 'flood' and are stored in a database accessible in a Solr instance. Many of files in this collection were dirty, meaning some of the files were partially, or entirely invalid to the topic at hand. Files that contained valid news articles often also included other irrelevant information.

2.3 Corpus: YourBig (Pakistan Flood 2010)

YourBig is selection of over 20,000 documents relating to a large flood that took place in Pakistan in 2010. These text files are held in the same database as YourSmall and can be queried similarly. Akin to the YourSmall collection, this set of documents also contains large amounts of text with erroneous information unrelated to the primary topic of the collection.

3 Cleaning Our Corpuses

3.1 WordNet

When cleaning up are data we decided to use the tool WordNet to help identify synonyms of our top 20 flood words. This allowed for our relatively small list of words to cover a much wider range of vocabulary. The way we went about implementing this was fairly straight forward; using the WordNet API we created a new list of the top X flood words. These words were then checked for in each paragraph of our collections and only paragraphs that contained more than two occurrences of the word were kept and marked as good data. These paragraphs would be used to represent our filtered out collection that removed a lot of the unnecessary text given in our collections.

3.2 Top 20 Words

To facilitate finding high quality files, paragraphs and sentences our group derived a set of twenty words that correlated strongly with the presence of information relevant to the topic of flooding. This list of 20 words is heretofore referred to as 'YourWords.'

The creation of the YourWords list was achieved by a combination of two techniques. First we took various articles online that speak about floods and combined them into a list of words. This approach was a very coarse and provided only a sampling of words from a few news sources. We removed stopwords from the MIT stopword dataset and also ignored words less than four characters. We then ran a frequency distribution on the resulting word set to pick the To hone in on our final list of YourWords we aggregated Twitter data that contained the word flood and did a frequency analysis on this data. Since we did not have access to the whole Twitter firehose, we used a subset of the data, being able to get 100 Tweets a request. We combined these two words list, taking similar words from both list and turn that into our final result.

YourWords	
flood	rain
overflow	dam
storm	severe
water	damage
submerge	washed
collapsed	river
discharge	downpour
downstream	flash
levee	sweep/swept
torrential	runoff

3.3 Frequency Analysis by Paragraph

We took our YourWords list and ran WordNet on them to generate a larger list of indicative words. We then tokenized both the YourSmall and YourBig collections by paragraph and checked each paragraph to see if it contained 2 or more words generated from WordNet, if the paragraph passed that check it was appended to a single file, all paragraphs not passing that check were deleted.

3.4 Decision Tree Machine Learning Algorithm

Besides generating a summary one of the key issues we were tasked with was, "how to deal with garbage text?". To solve this problem we used a classifier to classify good text and bad text in each of our collections and remove the files that contained irrelevant data. To do this we looked into three main classifiers: Decision Tree, Naive Bayes, and MaxEnt. We ran each of these classifiers and tested them on a tagged feature set to see which gave us the best results. After the process was complete we determined that the best candidate was the Decision Tree which gave us nearly 90% accuracy on our test feature set.

Colletion	Accurate	Inaccurate	Percentage
YourSmall	90	10	90%
YourBig	83	17	83%

3.5 Size and Duplicate Reduction

To facilitate the process of the reduction of size we incorporated the methods of WordNet, our Top 20 Words, frequency analysis, and classification. The pipeline we developed for doing this involved using WordNet to extend our 20 Words and then running through the files that the classifier tagged as good and grabbing only paragraphs that contained 2 or more of our Top 20 words and/or synonyms. After doing this process we were able to help dramatically cut down the size of our collections so that we could focus summarizing just the most important data.

3.6 Statistics on Cleaned Data

Collection	Pre-clean size	Post-clean size	bytes reduced
YourSmall	2.0 MiB	288 KiB	86%
YourBig	136.7 MiB	3.7 MiB	98%

Collection	Pre-clean # docs	Post-clean # docs
YourSmall	537	1
YourBig	20,416	1

4 Results

After all the analysis and development, we ran the same process on each of the corpora. The results are what follows in this section.

4.1 YourSmall (China Flood 2011)

> In June 2011 a flood spanning 9.94 miles caused by heavy rain affected the yangtze river in China. The total rainfall was 170.0 millimeters and the total cost of damages was 760 million dollars. The flood killed 255 people, left 87 injured, and approximately 4 million people were affected. In addition 168 people are still missing. The cities of Wuhan Beijing and Lancing were affected most by flooding, in the provinces of Zhejiang Hubei and Hunan. Finally nearly all of the flood damage occurred in the state of China.

4.2 YourBig (Pakistan Flood 2010)

> In August 2010 a flood spanning 600 miles caused by heavy monsoon affected the indus river in Pakistan. The total rainfall was 200.0 millimeters and the total cost of damages was 250 million dollars. The flood killed 3000 people, left 809 injured, and approximately 15 million people were affected. In addition 1300 people are still missing. The cities of Nasirabad Badheen and Irvine were affected most by flooding, in the provinces of Sindh Mandalay and Punjab. Finally nearly all of the flood damage occurred in the state of Pakistan.

5 Tools - Used

Like with any set of data, many different kinds of analysis can be performed on a natural language document in order to its different characteristics. In order to generate a summary for flood events, we first needed to filter out irrelevant documents using a classifier. Then by looking at only the relevant documents, we could start picking at the text to extract event information. To find the places where disaster struck, we could use

5.1 Regular Expressions

In most cases, text in natural language will express the same idea in different formats. A phone number can be written as 555-123-4567 or (555)123.4567, and dates could be delimited by slashes or dashes. Words can be spelled differently: the word "color" could also be spelled "colour". This poses a problem when trying to find a certain pattern in text, as searching for a literal string match will yield limited results. Regular expressions solve this problem, as they describe a language used to match variable string patterns.

A regular expression (regex) is an expression that describes a pattern of characters. By using a special language, regular expressions can be used to match string patterns that are variable but still conform to a specific format. They are commonly used to match things that like dates, email addresses, and phone numbers, but can also match more complex patterns like phrases or entire sentences.

In the analysis of the corpora, we noticed that most of the documents were from news reports, and were about the same type of event: floods. Because of this, information about the events and statistics would follow similar patterns between documents. Documents would report the cause of floods as "a result of ____", or damage costs as "$____ in damages". Since we already knew what kind of information we were looking for (rainfall, cost in damages, number of deaths, etc.), we could use regular expressions to effectively scrape data that were reported in a certain pattern.

In addition to the nature of the documents, we could only find regular expressions useful because the corpora were cleaned of irrelevant text. This is because more primitive regexes were used to find statistics like rainfall, compared to the more complex ones used to find flood causes. These primitive regexes navely matched all measurements like inches of rainfall or dollars in damages. If the data were not cleaned, we would include measurements from irrelevant events in the same article, which could skew the reported statistics.

Regular expressions turned out to be the primary tool for finding initial data from the document, but was mostly due to the nature of the corpora, and its cleanliness.

5.2 Contextualizing Location data

Geocoding is the process of taking a human readable location or address, and finding the corresponding latitudinal and longitudinal coordinates. Reverse Geocoding takes coordinates and turns them into a human readable address. For the purposes of this project we used the google geocoding api to contextualize our data. The python package pygeocoder was used to interface with the api from our existing python scripts. We used pygeocoder because it was easy to add to our existing codebase. We were able to enter all locations found by the regular expression into the geocoder which output corresponding contextual location information, which included: the state the location is in, the province the location is in, and the city the location is in depending on the type of location entered; for example, if a state is entered just the state will be in the output. If a province is entered the province and the state will be the output, and so on. This gave us two pieces of information: the geopolitical administrative level of the given location, and the corresponding sub and super levels. This allowed us to know the most affected countries, provinces and cities without that information being directly specified in the corpora.

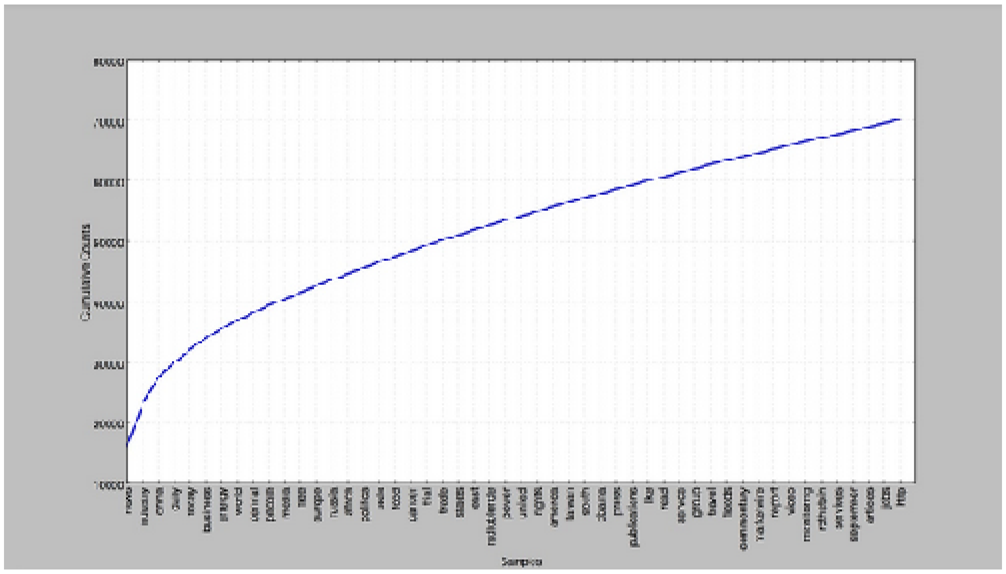
5.3 Decision Tree

One of the tools that was critical in size reduction of our collections was the decision tree. The basic way this form of classification works is through a series of observations and conclusions that form a tree-like structure.

Classification Method	**Percent Accuracy**
Decision Tree	90%
Naive Bayes	80%
MaxEnt	73%
Sklearn	92%

5.4 Frequency Analysis

We used frequency analysis to help sort our results generated from the regular expressions and to help generate our top twenty words. The following figure shows how this tool can be used to sort words or bigrams by how often they occur in the corpus.



5.5 Parts of Speech Tagging

We used the POS backoff tagger for our regular expression cause string. By Checking to see if the cause string returned by the regular expression contained a subject we were able to generate a sentence that was gramatically correct. ”In June 2011 a flood spanning 9.94 miles caused by heavy rain affected the yangtze river in China.” Without the POS tagging our sentence was: ”In June 2011 a flood spanning 9.94 miles caused by heavy affected the yangtze river in China.”

6 Tools - Explored

6.1 Mahout Kmeans Clustering vs. NLTK Clustering

6.1.1 NLTK Cluster Results - ClassEvent (Islip)

> Astounding, record-smashing rainfall swamps Long Island; 11 inches in 3 hours Astounding, record-smashing rainfall swamps Long Island; 11 inches in 3 hours Real Estate Rentals Cars Today’s Paper area forecast: Some humidity, clouds, shower chances through the work weekPM Update: Showers possible overnight, and again on TuesdayView all Archives More Astounding, record-smashing rainfall swamps Long Island; 11 inches in 3 hours By Jason Samenow August 13 at 1:38 pm More Comments Doppler estimated rainfall shows extreme totals over Long Island (National Weather Service) In just a few hours this morning, Long Island witnessed a jaw-dropping downpour unprecedented in New York state history.

6.1.2 Kmeans Results - ClassEvent (Islip)

Note the sentence has been filtered (stopwords, nonalpha, etc)

> ago islip today newsletters local paper aire ny recent register nb view clouds accident current inches pine subscribe archives cleared log account forecast scattered section south traffic long radar rain brentwood falls full historic newsday special conditions gallery sagtikos weekly map tv humidity mph flooding island manage pkwy photos record weather torrential rainbows capital clouds pileus cop gang rain year read state highway moses robert parkway parts due sunrise closed flooding ocean wednesday meadowbrook night

6.1.3 Kmeans Results - YourSmall (China Flood 2011)

Note the sentence has been filtered (stopwords, nonalpha, etc)

china monsoon people season heavy rains triggered year kill millions evacuate force floods hundreds

6.1.4 Kmeans Results - YourBig (Pakistan Flood 2010)

Note the sentence has been filtered (stopwords, nonalpha, etc)

home badin dec washington death million official basic flood sep activities pakistan view date items president title alerts destroyed gear vu yar incident khan toll report copyright rains accounts contact reaches dera post google pa relief victims blog homes sindh worsen news pur rahim shelter source flooding floods allah balochistan food

6.1.5 Kmeans Run Times

Corpus	**Run Time (Seconds)**
ClassEvent (Islip)	663s, ∼10 minutes
YourSmall (China Flood 2011)	1318s, ∼20 minutes
YourBig (Pakistan Flood 2010)	5737s, ∼1.5 hours

6.1.6 Analysis and Conclusion

Because the result accuracy from running the mahout kmeans was low we decided to also trying filtering out any sentences that did not contain a word from yourWords. We found that the results were slightly better but still not much of an improvement. In the end, we decided against using clustering methods with our corpus, instead turning to focus on cleaning our dataset more.

6.2 Chunking

Another tool that we explored was word chunking. Using the built in nltk chunker we could tag our sentences for locations, dates, and other meaningful phrases. While we found the tool to be fairly effective at its job, we choose to use regular expressions to find most of these keywords so the chunking was obsolete for our purposes. Of course, if time permitted we could have perhaps incorporated chunking along with our regular expressions.

6.3 NER Tagger

One useful tool in language processing was the Named Entity Recognition (NER) tagger. Our project utilized the Standford NER tagger which was able to identify things such as locations, persons, companies etc. The tagger was similar to the chunking method discussed before, and while both methods were accurate in results; it was ultimently unnecessary to include these tools in our processing.

6.4 Hadoop

When a dataset get too large to process serially, it becomes necessary to parallelize the job to decrease runtime. A common technique to run a task in parallel is MapReduce. Without going into specific detail, MapReduce is a technique in which the input is organized, or ”mapped”, into key-value pairs, and then processed by parallel workers (”reducers”) according to key.

Apache provides a software library called Hadoop that enables easier parallelization. Within Hadoop is Hadoop Streaming, an interface that allows for MapReduce using only the standard input and output of programs. Hadoop Streaming allowed us to easily incorporate MapReduce into our Python code, parallelizing any long tasks.

Using Hadoop for small sets of data had virtually no speedup compared to a serial run in all cases. However, running it with significantly larger sets of data showed a significant speedup in most cases. In cases where Hadoop did not seem to impact runtimes is most likely linked to the limitations of Hadoop Streaming using standard input/output, and its compatibility with the implementations our analyses.

6.5 Bigrams

To find the most common high quality collocations in a set of text files we first found all the bigrams in the corpora. We then compared the generated bigrams to the collocations in the provided Bigrams10k.txt. We used the collections.Count() function to find the top 50 collocations, which can be seen below. The top 25 results are relatively high quality compared with the bottom half of the list, and create a mostly clear picture that the text files are news stories about floods. We tried filtering stopwords and non alpha numeric content but the results were much worse and included noise from social networks and other news stories links. One of the problems with our approach is that the code takes a long time to run, even on a small collection. The results would improve dramatically if this method was used after the collection was cleaned using the previously mentioned methodology.

Top Bigrams

'breaking'	'news'	369
'press'	'releases'	294
'press'	'release'	203
'million'	'people'	132
'heavy'	'rains'	57
'registered'	'trademark'	53
'people'	'dead'	51
'state'	'media'	50
'death'	'toll'	49
'news',	'agency'	48
'special'	'election'	47
'people'	'like'	46
'might'	'like'	43
'news'	'pages'	43
'media'	'reported'	39
'climate'	'change'	39
'last'	'year'	30
'heavy'	'rain'	28
'first'	'time'	27
'death'	'penalty'	26
'email'	'address'	26
'hong'	'kong'	25
'stem'	'cell'	23
'flooded'	'street'	21
'flood'	'waters'	21
'third'	'parties'	17
'news'	'service'	16
'power'	'plant'	16
'political'	'news'	16
'agency'	'said'	15
'still'	'missing'	14
'many'	'people'	14
'third'	'party'	14
'fighter'	'jets'	14
'prime'	'minister'	13
'online'	'news'	13
'local'	'government'	13
'last'	'week'	12
'every'	'year'	12
'many'	'areas'	11
'drinking'	'water'	11
'news'	'reports'	11
'global'	'warming'	11
'breast'	'cancer'	11
'prostate'	'cancer'	11
'amid'	'fears'	10
'presidential'	'debate'	10
'cell'	'phone'	10
'people'	'affected	10
'three'	'decades'	10

7 Take Aways

7.1 Learning Goals

Throughout the project our team was very passionate about learning about how computers interpret and use natural language. A lot of us were immediately interested in the MapReduce portions of the class, while others were interested in the machine learning based classifications. We were tasked with the job of summarizing corpuses with the topic of Flooding. Success in the task was our driving motivation to achieve what we did in the end of the project.

7.2 Challenges

The initial challenges our team faced involved getting a grasp of the tasks ahead. There was a good deal of technologies we needed to learn to make our project a success. Topics in natural language processing, such as named entity recognition and clustering, were very new ideas to us. Distributed technologies and design paradigms like Hadoop and Solr, for large data sets, was an ease to understand at a use-able level, but a challenge to wield with full ability.

On a more logistics note, group organization and management was one of the tougher tasks. Balancing group schedules and meeting times were hard but once we established and set aside time, it was fine! Some work assignments in lessons that might not have been conducive to load balancing work was a huge challenge to overcome. Other logisitic operations, like group submissions were an ease once a schedule and rotation was in place.

7.3 Mistakes

One of the biggest mistakes we made was relying on our initial corpus data. Like many other groups, we had a big challenge midway through the lessons because of poorly cleaned data. Our analysis in the "Corpus Clean Up" section really shows how we reduced the given corpus down from a huge number to a very small number of documents.

7.4 Intuitiveness of Topic

Near the end of our project, we began grabbing specific information to build a template. Utilizing RegEx and grammar patterns to get specific data like deaths and damage amounts felt very straight forward. Our team had very little issues looking for patterns and building RegEx queries to grab the data we needed from our English documents. Luckily our filtered corpus only contained English documents.

7.5 Further Improvements

With our corpus and event data we obtained our estimates were a slight under final counts. This may have been due to the fact our articles and documents were written very closely after or during the event. If the documents were too close to the event, the underestimates would be likely due to news sources not fully understanding the severity of the flood. With additional time, applying mathematical modeling to a severity count would have been an interesting improvement to try. Estimate the final damage with when the event took place versus after all the counts on in would have gave us a better estimate at the damages and loss of life.

8 Conclusions

Our team explored many tools that are publicly available for natural language processing. The biggest take away we have is that making computers interpret English text is not as daunting a task as the average computer programmer would initially expect. Between python's natural language toolkit, the available machine learning algorithms and the high quality documentation available, natural language processing is more approachable than ever. We were able to generate high quality human readable summaries of large datasets within just 15 weeks using open source programs. We learned a lot during the course and would encourage other computer science students to take the class as well.

9 Appendix

9.1 MIT Stop Words

http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

9.2 Regular Expressions

```
Location:
"((in|at)\s([A-Z][a-zA-Z]{4,}|[A-Z][a-zA-Z]{2,}
\s[A-Z][a-zA-Z]{3,}))|\s+
[A-Z][a-zA-Z]{3,},\s[A-Z][a-zA-Z]{2,}\s[A-Z][a-zA-Z]{3,}"

Girth: "(\d+.\d+\skilometers|\d+.\d+\smiles)

Monetary: "\d+.\d+\s(million|billion|trillion| thousand)
\s(dollars|dollar) |
US\s\d+\s(million|billion|trillion|thousand)"

Lives:
"(\d,?)+)\s+(a-zA-z]+\s+){0,5}(missing|injured|otherkeywords)"

Causes:
"(due\sto(\s[A-Za-z]{3,}){1,3}|result\sof(\s[A-Za-z]{3,}){1,3}|
caused\sby(\s[A-Za-z]{3,}){1,3}|by\s([A-Za-z]{4,}){1,2})|heavy
\s([A-Za-z]{3,})"

Rainfall:
"((\d+.\d+\smillimeters)|(\d+.\d+\smm))|(\d+.\d+\s(inches|inch))"
```

9.3 Geocoder

https://pypi.python.org/pypi/geocoder/0.9.1

9.4 NLTK Text Book

http://www.nltk.org/book_1ed

9.5 Source Code

`https://github.com/JoeAcanfora/cs4984_unit1`