

Team Members:

Nick Crowder, Szu-Kai “Andy” Hsu, Will Mecklenburg, Jeff Morris, David Nguyen

**Computational Linguistics
Senior Capstone CS 4984
Virginia Polytechnic Institute and State University
Blacksburg, VA
Team Hurricane
December 12 2014**

Table of Contents

I. Executive Summary	4
II. User Manual	5
Introduction	
Running Files	
Files for Users	
III. Developer Manual	7
Introduction	
Files for Developers	
IV. Introduction	9
Group Collaboration	
Tools Used	
V. Unit 1 - Summarize with Word Frequencies	11
Intro	
Data	
Conclusion	
VI. Unit 2 - Summarize with Corpus Characteristics	12
Intro	
List of Stopwords	
Results	
Conclusion	
First Cleaning Attempt	
VII. Unit 3 - Summarize with Nouns and POS Tagging	15
Intro	
Results	
Problems Encountered	
Conclusion	
VIII. Unit 4 - Classifying and Using N-Grams	17
Intro	
Results	
Conclusion	
IX. Unit 5 - Summarize with Name and Entities	19
Intro	
Results	
Problems Encountered	
Second Cleaning Attempt	
Conclusion	
X. Unit 6 - Summarize with Topics	21
Intro	

	Results	
	Conclusion	
XI.	Unit 7 - Summarize with Indicative Sentences	23
	Intro	
	Results	
	Conclusion	
XII.	Unit 8 - Summarize with Filled in Templates	25
	Intro	
	Results	
	Conclusion	
XIII.	Unit 9 - Summarize with an English report	27
	Intro	
	Results	
	Issues	
	Conclusion	
XIV.	Conclusion	29
	Analysis of Project Management Tools	
	Analysis of Summarization Techniques	
	Solution Analysis	
XV.	Lessons Learned	31
XVI.	Acknowledgments	32
XVII.	References	32

Executive Summary:

This report presents the findings and results of our semester-long project to automatically generate summaries of two different collections of documents about hurricanes. The collections summarized include a small corpus about Typhoon Haiyan, containing ~2,000 files and a large corpus about Hurricane Sandy, containing ~70,000 files. One of the topics that is continually mentioned throughout the paper is the cleaning of the corpora, as they initially contained a lot of duplicates and noise. Since many of the methods discussed in the paper will not run effectively on datasets with lots of noise, we spent a lot of time and effort on cleaning the corpora.

The paper will begin by talking about the simpler methods used to summarize the collections. These methods include finding the most frequent words, using a WordNet, and using a part of speech tagger. Next the paper discusses how we used a classifier to determine if documents were about hurricanes or not. We then talk about using latent Dirichlet allocation (LDA) with Mahout to extract topics from the two collections. Next we talk about summarizing using k-means clustering with Mahout on the sentences in the two corpora in order to find a centroid, or a representation of the average sentence in the corpus.

The last two techniques we talk about both involve extracting information from the corpora with regular expressions. We first used the extracted data to fill out a template summarizing the two corpora. Then we used the same data alongside a grammar to generate a summary of the each of the corpora. After each summarization technique is talked about in detail, the conclusion summarizes the strength and weaknesses of each technique and then address the strengths and weaknesses of our specific solution to summarize the two corpora. Lastly we talk about the different lessons that we learned throughout the semester.

User Manual:

Introduction:

To use our Python scripts, a user must have Python 2.7 installed on their machine. The user must have NLTK dependencies installed, and must have downloaded the nltk tools and corpora via `nltk.download()`. The user needs installed gensim to run certain files(specified by the file name). Lastly the user needs access to a Hadoop cluster to run some of the files(specified by the file name). To run all of our Python modules, please edit the paths within `properties.py` to the local corpora paths prior to running any scripts.

Running Files:

All modules can be run from the command line using the default Python syntax:

```
python filename.py
```

If an IDE is being used, just run the specific Python file.

To run the mappers and reducers on Hadoop, see this example command:

```
hadoop jar pathToHadoopJar.jar -mapper mapper.py -reducer reducer.py -input  
InputDirectory/* -output outputDirectory/ -file mapper.py -file reducer.py
```

Replace `pathToHadoopJar` with the path to the specific Hadoop jar for that cluster.

Replace the `mapper.py` and `reducer.py` with the correct mapper and reducer files.

Files for Users:

1. **Classifier.py:** creates feature set for input corpus and outputs the accuracy of the classifier
2. **LDAtutorialCode.py:** outputs topics and probability of each word to occur in the topics from the input corpus, must have gensim installed to run
3. **Mapper.py:** passes each noun to the reducer.py, must be run on Hadoop cluster with reducer.py
4. **Ngrammapper.py:** passes each ngram from the input corpus to the ngramreducer.py, must be run on Hadoop cluster with ngramreducer.py
5. **Ngramreducer.py:** counts occurrences of each ngram passed in by the ngrammapper.py, must be run on Hadoop cluster with ngrammapper.py
6. **Properties.py:** contains paths to corpora and tools such as Stanford NER, must change paths for personal use
7. **Reducer.py:** counts number of occurrences of different things depending on what mapper it is being used with, output number of occurrences of items passed to it
8. **Regex.py:** extracts information from input corpus using regexes and outputs a summary of the corpus using the grammar in the file
9. **RegexTemplate.py:** extracts information from input corpus using regexes and outputs the information on a pre built template

10. **U2YourSmallStats.py**: runs statistical analysis on input corpus and plot information such as the number of instances of different size of words, and percentage of words that are not stopwords
11. **Unit2.py**: outputs percentage of times each word in our relevant word list appears relative to the total words in input corpus
12. **Unit3.py**: outputs most frequent nouns in the input corpus that are not in stopword list
13. **Unit4.py**: takes a small training set of labeled negative and positive documents, trains classifier, and then runs classifier on rest of the collection, outputs the accuracy of the classifier
14. **Unit5.py**: uses chunking on input corpus to find different patterns and the number of their occurrences
15. **Unit5Mapper.py**: uses chunking on input corpus to find different patterns, passes each one to the reducer.py, must be run on Hadoop cluster with reducer.py

Developer Manual:

Introduction:

A developer must have Python 2.7 installed on their machine. The developer must have NLTK dependencies installed, and must have downloaded the nltk tools and corpora via `nltk.download()`. The developer should also have gensim on their machine to work with some files (specified next to filename). The developer also needs access to a Hadoop cluster to work with certain files (specified next to filename).

If a developer is interested in improving and making extensions to the current project they should begin by looking at `Regex.py`. `Regex.py` uses a grammar to summarize the corpora and regexes to pull out information to be used by the grammar.

Files for Developers:

1. **Baseline.py**: outputs percentage of times each word in our relevant word list appears relative to the total words in Brown, Reuters, and State of the Union corpora
2. **Classifier.py**: creates feature set for input corpus and outputs the accuracy of the classifier
3. **LDAtutorialCode.py**: outputs topics and probability of each word to occur in the topics from the input corpus, must have gensim installed to run
4. **Mapper.py**: passes each noun to the reducer.py, must be run on Hadoop cluster with reducer.py
5. **Ngramfinder**: finds and outputs n-grams (bigrams as is) not containing stopwords
6. **Ngrammapper.py**: passes each ngram from the input corpus to the ngramreducer.py, must be run on Hadoop cluster with ngramreducer.py
7. **Ngramreducer.py**: counts occurrences of each ngram passed in by the ngrammapper.py, must be run on Hadoop cluster with ngrammapper.py
8. **Properties.py**: contains paths to corpora and tools such as Stanford NER, must change paths for personal use
9. **Reducer.py**: counts number of occurrences of different things depending on what mapper it is being used with, output number of occurrences of items passed to it
10. **Regex.py**: extracts information from input corpus using regexes and outputs a summary of the corpus using the grammar in the file
11. **RegexTemplate.py**: extracts information from input corpus using regexes and outputs the information on a pre built template
12. **SentenceTokenization.py**: splits up the input corpus by sentences
13. **Setence.py**: makes a file for each sentence in the corpus, filters sentences by length, sentences can be put in directory and have k-means run on them

14. **Statistics.py**: returns the percentage of words in the input corpus not in the stopword list
15. **TextUtils.py**: contains functions to filter numbers, words with non alpha characters, and non alpha characters from the input lists
16. **TextUtilsTag.py**: functions for filtering empty lines and words with specified tags
17. **U2YourSmallStats.py**: runs statistical analysis on input corpus and plot information such as the number of instances of different size of words, and percentage of words that are not stopwords
18. **Unit2.py**: outputs percentage of times each word in our relevant word list appears relative to the total words in input corpus
19. **Unit3.py**: outputs most frequent nouns in the input corpus that are not in stopword list
20. **Unit4.py**: takes a small training set of labeled negative and positive documents, trains classifier, and then runs classifier on rest of the collection, outputs the accuracy of the classifier
21. **Unit5.py**: uses chunking on input corpus to find different patterns and the number of their occurrences
22. **Unit5Mapper.py**: uses chunking on input corpus to find different patterns, passes each one to the reducer.py, must be run on Hadoop cluster with reducer.py
23. **WordDB.py**: contains a list of stop words that we created, used for filtering out noise

Introduction:

This paper pertains to the Computational Linguistics course or CS4984 taught by Dr. Fox in Fall 2014. The course covers the use of various linguistics tools to analyze corpora, specifically file collections scraped from the Internet. The focus for our group was to analyze two different collections of documents focusing on Hurricanes that made landfall. We were initially given a ClassEvent corpus containing a collection of files about heavy rains in Islip, New York. The purpose of the ClassEvent was to rapidly test techniques on a small, clean set of data before attempting them on the small and large collections.

The YourSmall corpus, our smaller collection of files, contained two thousand files pertaining to Typhoon Haiyan, however after removing insignificant files and duplicates, only 112 remained. The other collection was named YourBig, and primarily focused on Hurricane Sandy. YourBig was initially ~70,000 files, but was reduced to 2,000 after deleting irrelevant files. We wrote all of our code in Python using Eclipse and gedit (on the cluster).

Our first attempt at summarizing the corpora used the Hadoop open-source file management and MapReduce framework to process the large sets of data for the purpose of finding patterns in the data, as well as discovering the higher level topics of the corpora. Following this, we began the use of NLP (natural language processing) specific tools such as K-Means and Mahout to analyze the corpora in more meaningful ways. These efforts culminated in a generated template describing the event, which was filled in using data extracted from the documents.

Group Collaboration:

To work effectively as a group we utilized a variety of techniques and tools to coordinate our efforts. Initially, we set up a Google Drive account that managed all of our files, however we quickly realized this method was ineffective for code management. We then created a Git account to maintain the Python scripts. In addition, we used Trello, an online task board, for assigning and tracking jobs. Lastly, we used GroupMe, an online messaging client with SMS features, for centralized communication.

Tools Used:

Hadoop: Hadoop is an open-source framework that we used to manage the 11-node cluster.

Cloudera: A way to run Hadoop locally mainly for testing purposes.

LDA (latent Dirichlet allocation): LDA is used for obtaining topics from our corpora.

Mahout: Mahout is a tool used for clustering, classification, and filtering.

Python 2.7: A high level programming language that is compatible with all of the previously mentioned tools.

WordNet: WordNet groups words with other semantically equivalent words.

Unit 1

Intro:

This unit focused on the basics of linguistics and analyzing text. The first task was familiarizing ourselves with Python and the NLTK toolkit. The biggest issue we faced when running the various built in commands (such as creating a frequency distribution) was the number of stop words that were showing up in our result. Valuable words such as “rain” were getting pushed towards the bottom of the most frequent words list by words such as “and”. To remedy this issue, we utilized NLTK’s stopword list to filter these words out at runtime. Another thing we began to notice in this unit was the importance of bigrams. In our results, we found that the preceding word could easily determine the meaning.

Data:

We found that longer words tend to have more meaning to the summary of the corpus. Additionally, collocations were much more indicative of the corpus than single words.

Conclusion:

We discovered that frequency is a poor descriptor of what the dataset is about without filtering out stop words. Also, just looking at frequency gives no context to how words were being used. Collocations proved to be more relevant because the first word gives context to the second word and vice versa.

Unit 2

Intro:

Unit 2 focused on word frequency and relations. We started by manually creating a list of hurricane related words. We then constructed a WordNet based on our list to get similar words. Finally, we searched YourSmall and ClassEvent for the words in our list, and recorded the relative frequency (# word occurrences / # total words) of each word. We also searched the Brown corpus for words in our list to establish a baseline for the frequency of our words in an unrelated corpus.

Our list of stopwords:

In addition to the Stanford NER list of stopwords, due to our unique collections we had to add to that our own list of stopwords. Many of these are web specific, since our corpora documents were scraped from the Internet (Figure 1)

```
'rss', 'subscribe', 'comments', 'abc', 'copyright', 'desktop',  
'reddit', 'tweet', 'pinterest', 'careers', 'sitemap', 'email',  
'glossary', 'youtube', 'facebook', 'imagesfullscreen', 'afpgetty',  
'apfullscreen', 'news', 'people', 'said', 'epafullscreen', 'photos',  
'trending', 'celebrity', 'agencyfullscreen', 'stories', 'captcha',  
'audio', 'image', 'E-mail this to a friend', 'usa now', 'connect',  
'afp\getty'
```

Figure 1. Custom stop word list

We were still getting a large amount of garbage in our most frequent words list, which was a result of us not having cleaned our data effectively yet. This caused us to start expanding our list of stopwords.

Results:

Word	YourSmall	ClassEvent	Brown Baseline
hurricane	0.0061%	0.0246%	0.0003%
storm	0.0421%	0.2539%	0.0034%
rain	0.0123%	0.9628%	0.0738%
wind	0.0242%	0.0295%	0.0169%
eye	0.0013%	0.0059%	0.0273%
tropical	0.0068%	0.0354%	0.0022%
flood	0.0059%	0.6438%	0.0044%

haiti	0.0%	0.0%	0.0%
florida	0.0%	0.0%	0.0%
damage	0.0309%	0.1063%	0.0085%
injury	0.0%	0.0%	0.0011%
displaced	0.0026%	0.0%	0.0016%
water	0.0184%	0.3721%	0.0292%
category	0.0013%	0.0%	0.0016%
cyclone	0.0069%	0.0059%	0.0%
death	0.0113%	0.0236%	0.0103%
cloud	0.0014%	0.0532%	0.0038%
monsoon	0.0%	0.0059%	0.0%
blow	0.0018%	0.0	0.0049%
wave	0.0053%	0.0059%	0.0077%

Table 1. Percentage of our words in each collection

Conclusion:

We did not account for the difference in jargon when talking about storms from different locations. When the storm occurs in the Pacific, it is called a typhoon and when it is in the Atlantic, it is called a hurricane. This caused some of our expected words to have lower than expected relative frequencies.

We were able to determine words that are frequently associated with the word Hurricane. Those words tend to be related to some kind of destruction or loss along with any word pertaining to a large storm. In addition, we started to realize the problem that our word counts were not accounting for varied ways of writing the same word, such as “Nov” and “November”.

One thing that initially surprised us was the lack of frequency of the words that were in our WordNet. However, since the articles are pulled mainly from news sources, it is likely that they all tend to use the most straight-forward language in order to provide an easy reading experience for their audience. This would cause some words to be very prominent in the most frequent words list, while their synonyms would not occur.

First Cleaning Attempt:

We realized that we desperately needed to clean up YourBig in order to obtain accurate results when running the most frequent words script. Additionally, since we still had tens of thousands of files made everything we did on the Hadoop Cluster take a large amount of time to process. We initially determined that we would try to delete files based on file size. We arbitrarily chose 10kb as a starting point, and then looked at a large number of files around 10kb to see if they contained relevant material. We determined that a lot of files at 10kb contained useful articles, so we moved our cutoff line down to 8kb. Most files around that size did not contain pertinent articles, so we proceeded to delete all files 8kb and under. This left us with 30,000 files in YourBig and 1,500 in YourSmall. We also looked at the largest files and determined that they were all large sets of raw data, so we deleted the outliers. This only deleted a few files from YourBig and none from YourSmall.

Unit 3

Intro:

In Unit 3 we focused on tagging the words in our corpora by part of speech. We then used our part of speech (POS) tagging method to summarize the corpora by the most frequent nouns, as nouns are typically the most indicative of the content of a corpus.

We trained our POS tagger on a subset of 100 files from YourSmall, utilizing a trigram tagger, which backed off to a bigram tagger, then a unigram tagger, and finally defaulted to tagging as a noun. After training the tagger, we ran it on ClassEvent and YourSmall, and were able to determine that nouns made up 50.4% of words in the ClassEvent file and 55.6% of words in the YourSmall file. We also noticed that nouns were more indicative of the topics of the corpora than verbs or adjectives or any other parts of speech, confirming the hypothesis for the unit. We also implemented a MapReduce solution to find the most frequent nouns from YourBig.

Results:

Our most frequent nouns for both YourSmall (Table 2) and YourBig (Table 3) are listed in the table below.

YourSmall	
typhoon	29454
haiyan	16416
philippines	11712
tacloban	9944
people	8536
survivors	5411
november	5076
city	4795
u.s.	4737
storm	4578

Table 2. Most frequent nouns (YourSmall)

YourBig	
news	302880
sandy	218638
hurricane	207371
york	190120
post	173401
us	166648
obama	146601
october	142310
comment	128164
home	121675

Table 3. Most frequent nouns (YourBig)

Problems encountered:

We had issues running our POS tagger on the Hadoop cluster due to the cluster not having Numpy, a numerical python extension. This forced us to use a suboptimal POS tagger that would work on the cluster.

Conclusion:

We figured out that nouns can be very effective in determining what sort of information is contained in the collections. We also saw that a lot of insignificant web-related words still remained in our big, and were not being filtered. We determined that it would be a poor choice to just add them to the stop words list, as more Internet words would just float to the top to replace them. Instead, we started seeking ways to delete unrelated content from our files.

Unit 4

Intro: This unit focused on machine learning through the use of classifiers. We created two subsets of documents, one being negative files (files that were not about hurricanes) and the other being positive files (files about hurricanes). Both collections had approximately two hundred hand-picked files. The negative collection was made of flood related news articles. In addition to classifiers we focused on using most frequent n-grams, specifically bigrams, to summarize the corpus.

Results: A small sample of our most frequent bigrams on our big are represented in the table below (Table 4)

YourSmall	YourBig
('high', 'winds')	('north', 'east')
('localised', 'flooding')	('east', 'coast')
('roads', 'could')	('coast', 'places')
('could', 'make')	('night', 'time')
('daytime', 'temperatures')	('temperatures', 'falling')

Table 4. Most frequent bigrams

These bigrams help assign meaning to our common nouns. “Winds” was a common noun in both corpora, but after seeing the bigram “high winds”, we are able to see that the reason “winds” occurs frequently in both corpora is that hurricanes have high winds. The bigrams gave us a lot of insight into patterns in the documents that would be very useful in future units. An example of this is the “temperatures falling” bigram, which is a specific part of the phrase “temperatures falling below X”. The bigram indicated that this phrase is common in hurricane articles, so if we were to need to find the lowest temperature, we would have a clear way to do that.

Our classifiers did not perform up to our expectations. They both had a close to 50% accuracy, determined using cross validation, and therefore we did not opt to use either one for any further summarization efforts. To cross-validate the classifiers, we divided the original corpus into N subsets (folds), and we then train a model using all of the data except that fold. We then test the model on the excluded fold and determine whether or not it was correct. The low accuracy can be attributed to both the cleanness of the data sets and the fact that we used flood files as our negative. Since flood articles have a lot

in common with hurricane articles, the classifier did not effectively learn how to identify hurricane files.

The accuracy of each excluded fold for both of the classifiers are posted in the following table, as well as the average of all folds (Table 5).

Type	1st-fold	2nd-fold	3rd-fold	4th-fold	5th-fold	AVG
Naive	0.506667	0.640000	0.493333	0.573333	0.560000	0.5623
Decision Tree	0.506667	0.626667	0.573333	0.600000	0.573333	0.5759

Table 5. Accuracy of the classifier

Conclusion:

Our implementation of the classifiers was not effective due to our choice of training sets as well as the cleanliness of our data. Bigrams were very useful for giving context to our most frequent words, as well as showing patterns that might be used to find specific pieces of data.

Unit 5

Intro:

In Unit 5, we used named entity recognition (NER) to summarize our corpora using the most frequent named entities. We explored various methods such as NLTK's name entity chunking, Stanford NER, and utilized the Hadoop cluster in order to parse through our corpus and sort through different entities. After testing both tools and looking at some sample results, we determined that Stanford NER was more effective at detecting the named entities in the sample material, so we chose to use it.

We used our own grammars for the chunking aspect of Unit 5, instead of NLTK's built in method. The grammars that we used for chunking are listed below (Figure 2)

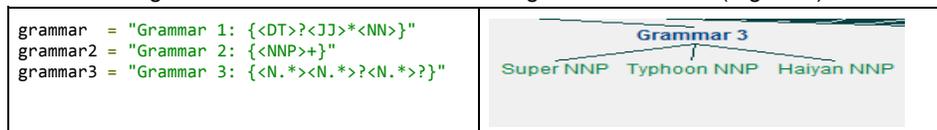


Figure 2. Three defined grammars & graphical representation of grammar 2

Results:

These are our results from running our NER on a sample set of sentences from YourSmall. The categories are the different types of named entities that Stanford NER is able to detect, as well as the number of named entities for each category from the sample set (Figure 3)

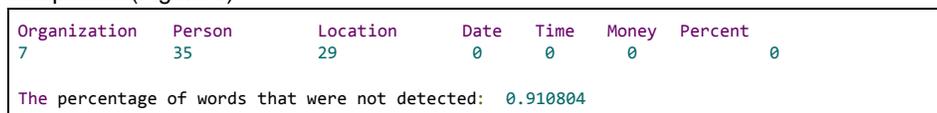


Figure 3. Name recognition entity results

Problems encountered:

We were unable to get the Python module "numpy" to work on the Hadoop cluster, preventing us from using Stanford NER on YourBig. Instead, we had to use NLTK's NER, which was suboptimal.

Second Cleaning Attempt:

At this point we determined that our data was still not as clean as we needed it to be. We wrote another Python script that would determine the relevance of each file based on the number of occurrences of the word hurricane, or typhoon in the case of YourSmall, and deleted the irrelevant files. This drastically reduced the size of YourBig to two thousand files, and our results after running this script were remarkably better than our previous units. YourSmall was fairly unchanged by this deletion. We also deleted duplicates, which reduced YourSmall to 112 files, and did not change YourBig.

This greatly increased the quality of results, as well as reduced the runtime of all of our scripts.

Conclusion:

This information gathered through chunking and named entity recognition was very helpful in giving context to some of the frequent words in our corpora. We were able to see, for example, that Tacloban showed up frequently because it was the name of the region where Typhoon Haiyan struck. With respect to the final goal of the project, NER was helpful in picking out certain important data points from the files. One of the key items that we would want to pull from an article on hurricanes would be the location where the storm made landfall. NER helps us find all the locations in the article so that we can specifically look at the most frequently occurring location, which is likely where the storm struck.

Unit 6

Intro:

In this unit we focused on using LDA with Mahout to extract topics from our data. After we ran LDA and collected our topics, we input our results into SOLR in order to determine whether or not a search on the topics would find articles about hurricanes.

Results:

The following are the results from running LDA on YourSmall (Figure 4) and YourBig (Figure 5). The results consist of a topic and a relative percentage for how prevalent the topic is in the corpus:

```
{central:0.10743399571253942, evacuation:0.05757070158514486, toll:0.03324732859300505, grave:0.025632665481392227, friends:0.01968800387189368, philippines:0.01932789237321569, forecast:0.01908562501464736, power:0.017559536160682666, nation's:0.016622171715427236, south:0.016484560103466026}

{us:0.402259764768619, read:0.05637992879323334, food:0.0455073680380059, last:0.037117477517028576, yolanda:0.03374614702502664, well:0.030168902393758783, typhoon:0.02399356074977917, samar:0.020829116931257285, dr:0.017321794538330455, officials:0.013537282255318816}

{marquez:0.05342234809134897, supplies:0.04691145235511276, killed:0.034779763774699796, chris:0.03374285621330252, island:0.031171877031696694, nicolas:0.026544337939198196, center:0.02652718439941285, share:0.024646858897682585, social:0.020938375108737004, hours:0.018782163837010285}

{philippine:0.12627241438682466, province:0.08163252519642183, link:0.03388990020524407, body:0.027508613106430838, address:0.02704201213933025, david:0.025071355460073937, using:0.021034136371276914, log:0.019949654051491537, car:0.019272470873047603, two:0.019198793928987358}
```

Figure 4. YourSmall LDA output results

```
{uscontact:0.0080607037758998, stayed:0.006751698739980142, hurricane:0.005636385154824114, flood:0.004842238390599852, family's:0.004798862647649561, stefani:0.0046864549038245, blogsblogs:0.00436321418632316, presscalderone:0.00431818091107269, ditches:0.004109495843486744, halted:0.004059492993521953}

{provide:0.026085693219503978, city's:0.019439330634066204, isn't:0.017053999802817212, tries:0.008068169088029306, toward:0.007604244968060675, sandyaftermath:0.004895610562430885, intimate:0.004880483533366992, yorklos:0.004613748866580673, carepollingwhite:0.004184492102296282, accountnotificationsprofilesettings:0.004141932729616573}

{toward:0.024381878323896072, provide:0.01710097615183011, city's:0.015593533790991867, isn't:0.015527878816323384, approximately:0.008553996671558674, yorklos:0.006950182812238167, sandyaftermath:0.005792757121692778, 2008author:0.005660229249715018, trademark:0.005046942592866921, ecclestone:0.004739355800350455}

{provide:0.028999476854741354, city's:0.019114794217594867, toward:0.008668452645960988, prediction:0.00776217438953668, damage:0.006224904119525, isn't:0.005445862506285112, yorklos:0.005383061906509139, immediate:0.005334314125357254, businessmoneyyou:0.004971774128457031, carries:0.004821870987847387}

{provide:0.03177325343821615, city's:0.013217421551902815, toward:0.00818799088572215, trademark:0.007879576494807599, storm:0.007756689695799677, approximately:0.007656991368379927, immedi
```

```
ate:0.005995331270130342, isn't:0.00554968745274174, burchshared:0.005502658986940906, faquser:0.00518198197171098}
```

Figure 5. YourBig LDA output results

The results show the most frequent topics and their frequency within the corpora.

An example of our SOLR queries is:

Query: event: Hurricane AND \"provide\" AND \"city\" AND \"isn't\"
Number of articles found: 848

Commented [1]: Box this?

The topics defining YourSmall had a higher relative percentage than the topics defining YourBig. This can be attributed again to the cleanliness of the files. At this point, we had not only deleted irrelevant files in YourSmall, but had also deleted the irrelevant parts of each file that remained. In YourBig, we had only deleted the completely useless files, but files with hurricane related content surrounded by unrelated content remained. Therefore, the total word count is brought up by the noise, bringing the relative percentages down.

Conclusion: The corpora needs to be cleaned heavily before running LDA becomes very effective. Our results for YourSmall were very good as it was almost entirely clean, however the results for YourBig were still slightly suboptimal due to noise.

Unit 7

Intro:

In Unit 7 we used k-means clustering on sentences. We wanted to see how well the centroid could summarize YourBig. We applied some restrictions to our sentences to make sure they were valid information. The restrictions are as follows:

1. Sentence length must be at least 3: We did this because any sentence under the length of 4 would be irrelevant. So we decided to skip them.
2. The sentence could not contain words from our custom stopwords list: The reasoning was so that any sentence that contained a word from our custom stoplist such as twitter, facebook, & share, would be ignored.
3. The sentence could not contain a colon, because this would mess with the parse clustering output script.
4. The sentence could not be longer than 21 words, as that would mean it was a run-on sentence, which would cause errors in the clustering.

Results:

Listed below are the closest words to the centroid as well as the squared errors (variance from the centroid), calculated using the squared euclidian distance method for YourSmall (Figure 6) and YourBig (Figure 7):

```
centroid: ['according:5.060', ' super:3.746', ' than:4.334', ' national:5.362', '
atmospheric:6.691', ' typhoon:2.405', ' u.s:4.605', ' winds:3.868', '$
Sentence: (672.82090899999992, ['according:5.060', ' super:3.746', ' than:4.334', '
national:5.362', ' atmospheric:6.691', ' typhoon:2.405', ' u.s:4$
(629.041967, ['according:5.060', ' super:3.746', ' than:4.334', ' national:5.362', '
atmospheric:6.691', ' typhoon:2.405', ' u.s:4.605', ' winds:3.86$
===

centroid: ['pang:1.634', ' through:1.107', ' just:1.330', ' buildings:4.899', '
slammed:3.837', ' friday:4.714', ' towns:1.438', ' typhoon:2.405', ' $
Sentence: (367.09674600000005, ['pang:6.537', ' just:5.319', ' buildings:4.899', '
slammed:5.115', ' friday:4.714', ' typhoon:2.405', ' estimate:6.0$
(328.87658600000009, ['buildings:4.899', ' slammed:5.115', ' friday:4.714', ' towns:5.753',
' typhoon:2.405', ' philippine:3.905', ' central:3.218', '$
(311.70878400000004, ['buildings:4.899', ' slammed:5.115', ' friday:4.714', '
typhoon:2.405', ' philippine:3.905', ' central:3.218', ' islands:4.986'$
===
```

Figure 6. Results of k-means & parse output script for YourSmall

```
centroid: ['academic:0.931', ' accepted:1.212', ' achieve:0.660', ' against:1.472', '
ago:1.812', ' alberta:1.452', ' algeria:0.931', ' amazon.com:1.026', ' angry:0.795', '
announcement:0.795'...]

centroid: ['academic:0.430', ' accepted:0.713', ' accordi:0.464', ' achieve:0.457', '
admiring:0.474', ' advisories:0.430', ' aesthetics:0.335', ' against:1.475', ' ago:2.263', '
agrees:0.518'...]
```

Figure 7. Results produced k-means & parse output script for YourBig

Conclusion:

K-means clustering on sentences was not an effective method for summarizing our corpora. The centroid of YourSmall had some descriptive words, such as typhoon, but many more unhelpful words, such as according. The centroid for YourBig was entirely unrelated words, largely because of the noise in the corpus.

Unit 8

Intro:

In Unit 8 we built and then filled in a template to summarize our two corpora. We looked at the results from previous units to determine what hurricane facts we would be able to pull out. We then wrote the template with blank spaces for each one of the data points we chose to include. Then we pulled the data out of the corpora using regular expressions and filled in the template. When attempting to pull the number of people who died, were missing, and were evacuated, we utilized cascading regular expressions. Specifically, we searched for keywords relating to death, missing, and evacuation, and then we ran an algorithm based on comparing the location of those words to the location of numbers in the sentence, in order to associate those numbers with the death event or the missing event or the evacuation event. The purpose of this was to implement a general solution for finding these facts, as opposed to a very narrow regular expression solution.

A sample regular expression from our template is below (Figure 8)

```
maxWindspeedString = "[0-9]{1,3}\s*(mph|MPH|kph|KPH|km/h)"
Output:
Winds gusted to as high as 235 mph, the Navy said.
```

Figure 8. A sample regular expression

Results:

Here are the results that we received from filling in our template for YourSmall (Figure 9) and YourBig (Figure 10):

```
The cyclone, Typhoon Haiyan hit in Tacloban in November 2013. Typhoon Haiyan, which formed in the Pacific, was a Category 5 storm, with wind speeds of up to 195 mph. The size of the storm was 600 kilometres wide, and it caused 15 inches of rain. In preparation for Typhoon Haiyan, 200,000 people were evacuated from the predicted path of the storm. However, the storm still killed 5,100 people, with another 2,000 still unaccounted for.
```

Figure 9. Template for YourSmall

```
The cyclone, Hurricane Sandy hit in New York in October 2012. Hurricane Sandy, which formed in the Atlantic, was a Category 1 storm, with wind speeds of up to 75 mph. The size of the storm was 1000 miles wide, and it caused 10 inches of rain. In preparation for Hurricane Sandy, 11 people were evacuated from the predicted path of the storm. However, the storm still killed 16 people, with another 5 still unaccounted for.
```

Figure 10. Template for YourBig

Many of the results matched up with the facts of the storms as provided by Wikipedia. The only area where there were issues was with the cascading regular expressions. Since it was a general solution, it was less accurate than the narrow regular expression

solutions, but if the algorithm were to be developed further, it could become as accurate as the narrow solutions while still remaining generic.

Conclusion:

The main issue is figuring out what should be included in the template. A template should be a good representation of what kind of relevant information can be pulled from a corpus. Templates should be detailed, but also have accurate information. With more detail, pulling accurate information becomes more difficult. So striking a good balance on the template can be difficult. We approached the issue of choosing facts to include by referencing all of our previous units. As stated above, each unit provided some insight into certain facts that could be included, such as work with bigrams informing us of the naming system for hurricanes (Hurricane/Typhoon followed by a name).

Unit 9

Intro:

The goal of unit nine was to use a grammar to generate a summary for our corpora building upon what we learned and built in the previous unit. There was no need for maintenance on our corpora at this point. We used regular expression to extract information from the corpora and a scoring system to decide what details to include in the summary. The scoring system was based off of the accuracy of our regular expressions and the percent that the top result was of the total number returned from the regular expression. The formula for the scoring system is as followed:

$(\text{Frequency of the top result} / \text{sum of the frequencies of the top ten results}) * \text{reliability}$

The reliability was manually determined based on the strictness of the regex.

Our grammar was as follows:

```
summary -> intro details close
intro -> [intro boilerplate using name, location, date]
details -> detail nextdetail
detail -> start [detailsentence]
nextdetail -> trans detail nextdetail
start -> [variations of the storm name]
trans -> ["additionally, also, furthermore, no transition"]
close -> [boilerplate about seeking more information]
```

Results:

Below are the results from the generated template for YourSmall (Figure 11) and YourBig (Figure 12):

The storm, Typhoon Haiyan, hit in Tacloban on November 2013. Haiyan was a Category 5. Additionally, the typhoon formed in the Pacific. Additionally, Haiyan caused 2,000 to be missing. Typhoon Haiyan had a size of 600 kilometres wide. Typhoon Haiyan caused 15 inches of rain. For more information, search for Typhoon Haiyan.

Figure 11. Summary for YourSmall

The storm, Hurricane Sandy, hit in New York on October 2012. The hurricane was a Category 1. Furthermore, the hurricane had a windspeed of 75 mph. Hurricane Sandy formed in the Atlantic. Also, Hurricane Sandy had a size of 1000 miles wide. Hurricane Sandy caused 10 inches of rain. For more information, search for Hurricane Sandy.

Figure 12. Summary for YourBig

The first sentence consists of some mandatory information that we will always try to include, regardless of the score. The details are included or not based on their score, and have some random elements to keep them from being too repetitive. The last sentence serves to tell the reader where they might find more information. Without the last sentence, the paragraph seemed unnatural, as it was just throwing facts at the user.

The last sentence acknowledges that the summary does not necessarily include all the facts by directing the reader to more information. Another potential option for the last sentence would be to generate a link to a search on the topic and provide that.

Issues:

An issue we ran into was that, since hurricanes often change in intensity, the articles would report different facts based on when they were written. For example, our regex picked up that Sandy was a category one, when it was a category three at its peak. This problem occurred due to the fact that most of our articles were written in the United States, and therefore were reporting the intensity of the storm when it made landfall in the States, instead of the highest intensity. This demonstrates the effect that the source of the articles has on the content, and by extension, our summary.

Another issue was that there was a large number of results for “Southern Ocean” due to the a large number of news sources reporting the location as the “Southern Pacific Ocean”, which both “Southern” and “Pacific” in our regular expression. This is indicative of a problem inherent to regular expressions, as there are often unintended matches that can affect the results.

Conclusion:

We used regular expressions to pull out all of the specific details that we wanted from our collections. The grammar produced a naturally flowing paragraph, and did a good job at only including facts that were likely to be correct. Many of the facts that were discluded were poor results from the cascading regular expressions. There are also a number of issues that are unavoidable due to the solution that we chose. By using regular expressions, it is possible that we will have unintended matches skewing the results. Additionally, to get the best summary, we would need to take into account the origin of the documents.

Conclusion

Analysis of Project Management Tools:

Initially we had issues coordinating our work and managing our Python scripts. We would frequently redo tasks that others in the group have already performed. We fixed this by initializing a standard version control repository (GitHub) and frequently updated our tasks and workload every week. Later in the unit we all agreed to meet up at a common time which improved our work efficiency.

Analysis of Summarization Techniques:

The most basic summarization technique that we used was finding the most frequent words. Word frequency with filtering of stopwords gives us a good basic overview of the corpora, but it does not provide any context for the words. The next technique used was using nouns and POS tagging to summarize the corpora. We tagged the words and then picked out the most frequent nouns. These provided a more accurate summarization than just most frequent words alone, as adjectives are often not a good representation of topics in the corpora. However, tagging still does not provide a sufficient summary for the collection. The words coming out still have no context.

N-grams were the next technique we looked at. Since N-grams associated multiple words with each other, there was more context provided. The most frequent N-grams often included a word from the most frequent words list, but associated that word with another noun or adjective, which often informed us of why that word was common, such as the common word being winds and the bigram being high winds. We learn that winds is common because hurricanes often have high winds.

Next we looked summarizing by named entities. Named entities in isolation are not a strong technique because they give things like organization names with no context. Named entities were useful when combined with other techniques such as n-grams. Named entities were good, however, at pulling out locations. The most frequent location is very likely to be the location that the hurricane made landfall.

After named entities we tried using LDA to pull out topics from our collections. LDA was an effective method for seeing some of the high level topics of the collections, along with metrics for why they were selected. LDA, like most frequent words, was unable to provide context. Additionally, LDA was not effective when the corpora were not cleaned.

We used K-means clustering by sentence to determine the centroid for our collections. K-means with our sentences was less effective than LDA. It gave fewer relevant words, especially for YourBig. This technique, like LDA, gave us a general idea of what information was available for extraction, but failed to provide specific details or context.

Regular expressions were the final technique that we used to try and summarize the collection. This was the most effective method for us. Regular expressions allowed us to be more precise and pull out specific information such as the location and size of the hurricane. We were able to write effective regular expressions because the previous units had given us a good idea of what was available to extract, as what the data would look like in the article. The primary issue with regular expressions was under or over-matching. Additionally, regular expressions would under-count certain data points if they were written in different ways, for example “Category 5” vs. “Category five”.

Solution Analysis:

The biggest strength of our solution is how much information the regular expressions can extract. They can extract informative details as well. From experimenting with the regular expressions we were able to make general expressions that seem to extract great information from our corpora. News articles sometimes tend to have information in sentences laid out in certain manners. They also tend to use certain words to describe the hurricanes that they are reporting on. These two facts helped us in writing effective regular expressions.

We think that the main weakness in our solution is the fact that regular expressions are somewhat limited as to how much information they can extract. We wrote effective regular expressions for our corpora, but they often undercounted since sentences in English can be written in so many different ways. Our regular expressions cannot detect every data point.

If we had more time, then we would look at some smarter solutions. The regular expressions work well but do have weaknesses as mentioned previously. We would want to add a method for combining all of the regular expression matches that mean the same thing but are written differently. If we had 12 matches for Category 5 and 6 for Category five, we would want the end result to be 18 for Category 5.

We also would try to experiment more with combining the different tools that we used in new ways. For example, we could try to take the output from K-means clustering and try to feed that into regular expressions. The different clusters could be pushed to different regular expressions depending on what data was contained inside of them. Additionally, we would want to make a more advanced grammar and text generation method, so that it can seamlessly create the detail sentences. One potential way to improve the text generation would be to take each fact and search the corpora for the surrounding words, and then use an english grammar to construct a sentence based on the words that frequently surround that fact.

Lessons Learned:

The lessons we learned provided a foundation that will help us grow and mature into professionals. Many of us were not familiar with Python, but thanks to this course we all now believe we have a greater understanding of the syntax. We learned about some of the many NLP tools and techniques with the assistance of the NLTK library and resources provided to us. The importance of using the Hadoop cluster was emphasized by having to use it constantly on the large data set of YourBig. We also learned a lot about linguistics in general, and the difficulties faced by researchers in the field.

Another important lessons learned was the improvement of time management and separation of tasks. We utilized Trello, Git, and groupme for communication reasons. We constantly had to refactor our code during the beginning of the semester. To remedy this we created a properties files (Python), which helped all of the members maintain their local paths. Looking back, we should have done this earlier.

Something that we all learned to appreciate is how difficult automating summaries of data can be. We ended up creating a good summary, but it took a semester and a team of five to accomplish this task.

Acknowledgements:

Special thanks to:

Dr. Fox
Xuan Zhang
Tarek Kanan
Mohamed Magdy Gharib Farag
National Science Foundation

References:

Bird, Steven, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. Beijing: O'Reilly, 2009. http://www.nltk.org/book_1ed/. O'Reilly Media, June 2009. Web. 5 Dec. 2014.

"Welcome to Apache Hadoop!" Web. 12 Dec. 2014. <http://hadoop.apache.org/>

"What Is Apache Mahout?." Web. 12 Dec. 2014. <<http://mahout.apache.org/>>.

"Welcome to Python.org." Web. 12 Dec. 2014. <<https://www.python.org/>>.

"K-means." *K-means*. N.p., n.d. Web. 12 Dec. 2014. <<http://nlp.stanford.edu/IR-book/html/htmledition/k-means-1.html>>.