CS4984/CS5984: Big Data and Text Summarization

HURRICANE MATTHEW SUMMARY

December 15, 2018

Professor: Edward A. Fox
Brendan Gregos
Michael Goldsworthy
Thoang Tran
Areeb Asif
Virginia Tech, Blacksburg, VA 24061

Contents

List of Tables

List of Figures

Acknowledgements

We put a lot of effort into seeing this project to its completion; however, it would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them.

We would like to express our deepest gratitude and would like to thank our Instructor Dr. Fox for providing his timely suggestions and stimulating feedback. His encouragement helped us achieve valuable results. We also appreciate his efforts in providing us with extremely good study materials and reference guides which enhanced our learning.

We would like to thank the National Science Foundation for the funding (NSF: IIS-1619028).

We would like to extend thanks to all the guest speakers who provided us with needed equipment and several useful approaches to the project. We would also like to appreciate Liuqing Li, the TA for the course, for his frequent help to us in guiding us regarding problems incurred during our development process. We would also like to thank him for his special lectures which helped us learn new tools which were of immense help to the project.

Last but not least we would like to appreciate the generous support we have provided to each other. We would also like to congratulate each team member for their continuous effort and investments of hard work which helped us in achieving our goal.

Executive Summary

This is the final report for team 3 of CS4984/CS5984: Big Data and Text Summarization. Our group was tasked with attempting to generate an automated summary of two collections of webpages relating to the Hurricane Matthew event in 2016: a small (~420 articles) and a large (~11000 articles) data set.

Both of these collections contained a large number of noisy and irrelevant documents, so cleaning the data was an important and ongoing process throughout the project. We found that the best method for cleaning unwanted JSON noise in the articles was jusText, and we developed a method based on filtering documents by their LDA topics which proved to be a good way to remove irrelevant documents.

In generating our summary, we completed a number of initial and intermediate tasks along the way, such as counting the most common words, sorting the words into their parts of speech, and finding named entities. These were useful intermediary steps towards our final method of summary generation, as many of the tools we used to solve these problems, such as GenSim and NLTK, proved useful for generating a template summary and an extractive summary.

Ultimately, our extractive summary, which was made using the algorithm TextRank, proved to be better than our template summary, although we believe it is likely that the template summary could have been expanded, through the inclusion of more slots, which would boost its performance.

In addition to the conclusions drawn by our team regarding automated summary generation, we discovered which tools and project management methods were most useful.

Introduction

This report presents the methods, results, and conclusions from our team's attempts to summarize news articles related to Hurricane Matthew. Our team had two data sets to work with, one with ~420 articles, the small data set, and another with ~11000 documents, which were initially in the format of web archive (.warc) files.

Initially our group had to select tools to use and organize our team responsibilities. We chose to organize our code base using Git. We initially chose to use PySpark for the bulk of the code, and later changed to just Python. We used a number of Python libraries: NLTK, Gensim, spaCy, Sci-Kit Learn, and others. This will be explained in more detail in the User and Developer Manuals.

Another initial task involved cleaning the data and removing duplicate documents and irrelevant documents. This proved to be a difficult and important task; it was an ongoing process throughout the entire project. We found a lot of JavaScript and other HTML code, which we cleaned with jusText. We found that filtering the documents by LDA topic was a good method for removing duplicates.

Our group tried many different paths to summarization. This report follows the sequence of increasingly complex summarization attempts in each chapter. Initial tasks were as simple as counting the most common words, and the most common words by part of speech. Intermediate tasks included finding the LDA topics, and finally summarizing the corpus using regular expressions and TextRank.

Finally the report will present our key takeaways, an analysis of the tools and algorithms we used, our lessons learned as a team, and directions that we would have liked to explore if we had more time.

Chapter 1

Unit 1 - Most Frequent Important Words

1.1 Introduction

Our approach involved a simple cleaning step of the collections of websites about Hurricane Matthew. The collection was processed to remove the functions and HTML code that create the appearance of the webpages. After this cleaning process, we obtained a JSON file which contains all the text content of the webpages.

As is common in natural language processing, this initial text file still contained a lot of irrelevant words and punctuation. For example, articles, (many) verbs, and prepositions are not important to the meaning of the data set even though they are heavily used in natural language. We processed the new JSON file to filter out and remove the stop words from NLTK that are frequently in natural language but do not carry any important meaning about the data set. In addition to these stop words from NLTK, we also created a list of custom stop words that appeared regularly in the JSON file but do not have any relevance.

Once we filtered out stop words, custom stop words, and punctuation in the data set, we counted the words to find the term frequency of the words in the data set. The idea behind this is that the greater the frequency of the word appearing in the data set, the more its meaning is indicative of the content of the data. However, we need to account for situations where a word can appear a large number of times in a single document of the collection of webpages but nowhere else in the collection. Such situations would skew the results and make a word having greater significance to the collection of webpages than it actually is. In order to eliminate this situation, we adopted the term frequency-inverse document frequency (TF-IDF) statistics to add weight to each word. The term frequency of a word is offset by the number of documents in the collection that contain this word. Using the TF-IDF statistics, we assigned the weight to each word, which reflects its importance in indicating the meaning of the collection of the webpages.

1.2 RESULTS

Our results show the most frequent words using Term Frequency and the top of the most frequent important words using TF-IDF.

Words	Frequency	Important Words	TF-IDF
hurricane	18581	hurricane	11727.4
matthew	9754	storm	9277.9
storm	9253	matthew	9069.8
people	7242	october	8320
said	4925	people	8014
florida	4516	haiti	6973
haiti	4313	florida	6674
south	4294	destruction	6419
october	4058	south	6225
north	4019	winds	5705
one	4006	water	5155
winds	3748	carolina	5152
new	3672	beach	5143

Table 1.1: Word frequency and the most frequency important words

1.3 CONCLUSIONS

The TF-IDF results provided general ideas about our data set. By observing the word frequency, we learned that our data set is talking about hurricane, storm, destruction, etc. There are limitations to the TF-IDF results, however. For instance, the word frequency does not give us sufficient information for the whole context of our collection, or how words were being used in the data set. Therefore, in the next step, we will utilize different techniques to analyze the collection to get more complete information.

Chapter 2

Unit 2 - WordNet Synsets

2.1 Introduction

WordNet Sysnsets are elements of the WordNet lexical database of English words. There are sets of words grouped together if they are synonyms. Each word can belong to a number of different synsets, depending on how it can be used, for example ‘ripe’ may be in the synset of words related to fruit ripeness, but also in the synset relating to opportune moments (‘The time is ripe for change’).

These synsets are potentially useful for text summarization because finding the synsets for words in a document tells us much about the semantic meaning of the text, and finding the most common synsets may be a good way of getting a kind of ‘dot-point’ summary of the key words and themes in a text.

2.2 Method

Initially we tried a simple and naive method of using the NLTK wordnet interface [7] to identify synsets on the list of most common words. Of course this is a problematic method since it will give so many unrelated and useless synsets. Consider for example, ‘wind’: in a hurricane collection, we almost always want to talk about the noun, the movement of air, rather than ‘wind’ the verb, to move in a twisting or spiral course.

Thus, we identified the need to identify which synset each word belongs to by considering the context of the words around it. A very common method for this task is the Lesk algorithm [8]. This algorithm considered the synsets of the neighborhood of words around a given word, and finds the synsets that best overlap.

This algorithm has an implementation in NLTK [9]. As discussed in the Introduction chapter, we initially used PySpark for our tasks, but we ran into an error which prevented us from using the NLTK Lesk implementation on our data in combination with Spark. We set this task aside for a long time, and completed the other tasks. Due to time constraints we never got back to this task, and thus have no results.

Chapter 3

Unit 3 - Part of Speech (POS) Tagging

3.1 INTRODUCTION

Part of Speech (POS) refers to the grammatical type of a word in a sentence, such as a noun, verb, adjective, etc. There are a multitude of motivations for identifying part of speech, primarily to filter words by part of speech, or to extract information based on the relations between tagged words.

The way POS tagging is done can vary, and there are a number of methods. Typically it is not good enough to simply look at words individually, as we found out, since many words, for example 'fish', may have different syntactic functions depending on the context ('to fish' vs. 'a fish'). An approach that works much better is looking at words in the context of the sentence they are in and determining their POS by the words around them. We used the NLTK POS tagger, which uses an averaged perceptron. An even more powerful tool is spaCy, which we found worked extremely well, despite taking a large amount of time.

3.2 METHOD

With the goal of finding the most common words grouped by POS, initially we naively used the NLTK POS tagger on our list of most common words from unit 1. This, of course, is not useful, and gives essentially no extra information that is useful for creating a summary. On top of not being useful, it is also incorrect. Our method from unit 1 leaves a simple list of words, lacking all context, making the POS tagger mark all instances of each word with the same POS.

After realizing our naivety, we used the NLTK tagger on the text of the webpages in our collection. The NLTK POS tagger uses an averaged perceptron tagger, described in [2]. This gave much better results. We also used a similar, but slightly better tool, the spaCy POS tagger, which uses a similar underlying algorithm. The results were still quite bad as a summary of Hurricane Matthew, since we had not yet removed documents relating to the other hurricanes, such as Irma and Maria, and a lot of the most common nouns were clearly names and places that were important to those other topics.

Finally, after cleaning the data using filtering based on LDA topics later in the semester, we redid the POS tagging again on the big data set that had been filtered to remove as many irrelevant documents as possible, and the resulting lists of nouns and verbs were much better.

3.3 Results

As mentioned in Method, we initially ran the NLTK POS tagger on the words from Unit 1 in the order of most frequent to least frequent. This approach is incorrect, none the less, here are the top 10 most frequent nouns and verbs from our small data set using this method:

Noun (frequency)	Verb
‘hurricane’, 4595	‘wind’, 1119
‘storm’, 1967	‘said’, 705
‘matthew’, 1539	‘north’, 544
‘people’, 1131	‘hit’, 449
‘home’, 877	‘expected’, 418
‘today’, 849	‘carolina’, 371
‘category’, 787	‘beach’, 360
‘island’, 776	‘landfall’, 323
‘resident’, 707	‘flooded’, 285
‘water’, 610	‘get’, 270

Table 3.1: Results of naive POS tagging

The next results come from tagging the words in their context, then counting them; these results coming from the big data set after being filtered by LDA topics:

Noun (frequency)	Verb
‘matthew’, 22260	‘said’, 7889
‘hurricane’ (proper noun), 20227	‘expected’, 3206
‘storm’, 13294	‘flooding’, 1596
‘haiti’, 10445	‘say’, 1504
‘people’, 9029	‘according’, 1466
‘florida’, 8771	‘including’, 1419
‘hurricane’ (noun), 7105	‘damaged’, 1382
‘carolina’, 6963	‘help’, 1304
‘wind’, 5024	‘evacuate’, 1295
‘october’, 4569	‘get’, 1290

Table 3.2: Results of better POS tagging

3.4 CONCLUSIONS

The first set of results comes after cleaning out most of the unwanted JavaScript and HTML noise, but many documents were still unrelated to Hurricane Matthew; instead they were about other hurricanes or completely unrelated topics. This is not a failure of the method of POS tagging in this way, but instead due to poor cleaning. Despite this, the noun list is surprisingly good. The verb list however, lists ‘wind’ as a verb, as in ‘to wind a string’, whereas it is clearly going to be more commonly used as a noun in the context of articles about hurricanes. ‘carolina’ is another strange word to be classified as a verb. This shows the limit of POS tagging words that are not in their proper context.

The results from the big data set are much more encouraging. In fact the list of nouns serves as a great list of keywords, including the name of the storm, the places that were hit, and the month of the incident. The list of verbs contains some useless verbs such as ‘said’ and ‘including’ which do not add much information. Possibly the lists generated here could be filtered by an importance ranking, perhaps using tf-idf, to get a list of important verbs. In spite of this, some verbs such as ‘evacuate’ and ‘damaged’ are very relevant in the context of Hurricane Matthew.

It is clear from these results that POS tagging words and sorting by their frequency is a good way of getting a list of important nouns, and with some filtering a good set of verbs as well.

Chapter 4

Unit 4 - A Set of Words/Word Stems

4.1 Introduction

English has rather complicated grammatical rules dictating how a word, especially a verb, is used. For instance, the verb "to be" can have many different forms such as am, are, is, was, and were. If we count the frequency of the words in a document, the different forms of a word must be counted as one in order to have an accurate counting. The methods for reduction of different forms of a word into a single common base form are called stemming and lemmatization.

Stemming is a simple process of truncating the ends of words to arrive at the common base form. For example, the stemming of the words organize, organizes, and organizing would reduce them to organize. The process is rather crude and thus do not take into account other variations of the words, such as the verb "to be" above.

Lemmatization is an advanced operation that take into account the morphological analysis of words. It aims to remove inflectional endings only and to return the base or dictionary form of a word. For example, the word "saw" would be returned either as see or saw.

In our analysis, we performed lemmatization using "Words Lemmatization" in NLTK to convert the different forms of words into common base of words. Once we reduced the words into their common bases, we counted the words to compute the term frequency discussed in Unit 1.

4.2 RESULTS

The result shows the comparison between the word frequency with and without lemmatization.

Words	Without Lemmatization	With Lemmatization	Difference
hurricane	18581	19439	858
storm	9253	9873	620
matthew	9754	9793	39
people	7242	7259	14
said	4925	4925	0
florida	4516	4519	3
haiti	4313	4314	1
south	4294	4294	0
october	4058	4058	0
north	4019	4019	0
one	4006	4146	140
winds	3748	5295	1547
new	3672	3672	0

Table 4.1: The word frequency with and without lemmatization

4.3 CONCLUSION

As we can see from the above results, without lemmatization we missed many variations of the same words. This led to under-counting these words. Moreover, in some cases where there is no variation, such as 'October' or 'south', performing lemmatization does not hurt our accuracy. Thus, using the lemmatization gives us more accurate word frequency.

Chapter 5

Unit 5 - Named Entity Extraction

5.1 Introduction

The goal of named entity recognition is to a) find proper nouns in text and b) classify their type. These types can include organizations, places, and amounts such as money, cardinal numbers, and so on. We thought that this type of classification could prove incredibly powerful, because we could then take data ("12 houses were lost to flooding") and turn it into a relationship, linking "12 houses" with "flooding".

5.2 Results

The initial implementation of this was not difficult. We leveraged spaCy, a powerful NLP library, which has built-in named entity recognition [6]. This approach presented a few problems, however. First of all, it was slow. A lot of this has to do with the internals of how spaCy works: it is a pre-trained neural network that looks at a piece of text once and determines several important qualities of the text; it does chunking, sentence recognition, part of speech tagging, named entity recognition, and more, all at once. This simplifies the code, as the user just needs to call

```
nlp("sentence")
```

to run spaCy on a string. The speed downsides inherent in doing a lot of extraneous work are considerable, however: the large data set took around three hours to run through spaCy, and spaCy does not have any good ways to serialize the data to save it for later[11]. Therefore, we took a two-pronged approach to make runtime acceptable.

The first thing we did to increase speed was migrate our code from Spark to using regular Python. This had several advantages. The main reason we moved was for speed. Using Python's standard multiprocessing library [10], same-machine performance increased by about 50%. We weren't able to get multi-node workloads running smoothly with Spark, so moving to a single-machine paradigm made sense for us. The other big advantage we gained by doing this is that the multiprocessing library is much better-documented and easier to use than Spark, and our development pace from this point forward was much further accelerated.

We decided to step around the serialization problem by outputting to a text file. This file would contain the named entity results for proper nouns and the part of speech for all other words. We could then further process this file to extract meaningful data. The results this method generated were pretty good, and a sample of the output file is copied below:

```
Matthew PERSON
is VBZ
expected VBN
to TO
pick VB
up RP
speed NN
tonight TIME
as IN
the DT
center NN
approaches VBZ
southwestern JJ
Haiti GPE
and CC
nears VBZ
eastern JJ
Cuba GPE
late Tuesday DATE
. .
```

Chapter 6

Unit 6 - LDA

6.1 Introduction

Latent Direchlet Allocation (LDA) is a way to model the topics of a document. Each document is assumed to have a number of topics, with each words having a probability to appear in these topics. Topics are identified automatically by looking at the frequencies of words appearing together. For example, in our raw data, topics that are likely to be detected are related to Hurricane Matthew, Hurricane Irma, or hurricanes generally.

The exact mechanism of LDA is outside the scope of this report, but it is worth explaining as a black-box method, which is ultimately how it was used in this project. The primary inputs are any number of documents represented as a bag of words, as well as an integer number of topics to be found. The bags of words are frequently converted to a numerical representation using a method called id2word. The output will be a set of sets of words and probabilities, representing the likelihood of these words being found in each topic. A score, the coherence score, can also be computed, which is the average of the pairwise word-similarity scores of the words in the topic, which can be seen as the strength of these topics.

These topics can then be used in a number of ways to contribute to the building of a summary. In our case, we used LDA topics to filter out any document that was not related to the topic of Hurricane Matthew. We then ran LDA again on the remaining documents, and extracted sentences based on their strength for the topics, to hopefully cover the widest amount of information in the fewest number of sentences.

The tool used was the Gensim library [3], which has an easy to use LDA modeller.

6.2 METHOD

Although there are a number of tutorials for using the Gensim LDA topic modeller and getting good results, we primarily followed the guide in [5]. The basic algorithm is as follows:

Algorithm 1 LDA filtering

```
 1: procedure PRE-PROCESS DATA
 2:     Remove duplicate documents
 3:     Remove stopwords and lemmatize
 4:     Create id2word dictionary
 5:     Convert documents to bag-of-words representation using dictionary
 6: procedure FIND OPTIMAL TOPICS
 7:     loop from i=2 to 30:
 8:     Perform LDA with $i$ topics
 9:     Compute coherence score
10:     Save best topic number
11: Manually identify which topics correspond to Hurricane Matthew
12: procedure FILTER DATA
13:     Select documents which have a strength score higher than 0.3 in one of the topics
    related to Hurricane Matthew
```

Figure 6.1: LDA Filtering Algorithm

After doing this, there is a new collection. LDA can be run again on this corpus to find the topics of the documents that are related to Matthew, which are the topics that are much more interesting for the purpose of generating a summary. Ideally, it was hoped that the topics will correspond to different aspects of the hurricane; perhaps one of the LDA topics will be about the wind speed, one may be about damage, one about evacuation areas, etc. After finding these topics, the corpus can be split into sentences, and sentences that are closely related to these topics can be found, which results in an LDA based extractive summary.

6.3 RESULTS

The number of topics that were found to have the highest coherence score for our big data set was 8. These are the sets of words and their percentages:
$[(0, u'0.008 * "one" + 0.007 * "would" + 0.007 * "time" + 0.006 * "like" + 0.006 * "new" + 0.006 * "get" + 0.006 * "day" + 0.005 * "right" + 0.005 * "many" + 0.005 * "year"'),$

(1, u'0.014 * "trump" + 0.011 * "election" + 0.010 * "state" + 0.010 * "clinton" + 0.008 * "voter" + 0.008 * "court" + 0.007 * "vote" + 0.006 * "president" + 0.006 * "political" + 0.006 * "white"'),
(2, u'0.010 * "may" + 0.007 * "information" + 0.007 * "service" + 0.006 * "emergency" + 0.006 * "disaster" + 0.006 * "email" + 0.005 * "please" + 0.005 * "policy" + 0.005 * "use" + 0.005 * "plan"'),
(3, u'0.014 * "dog" + 0.013 * "game" + 0.010 * "van" + 0.010 * "cocoa" + 0.009 * "september" + 0.008 * "player" + 0.008 * "bay" + 0.007 * "tampa" + 0.007 * "retrieved" + 0.007 * "que"'),
(4, u'0.033 * "hurricane" + 0.027 * "matthew" + 0.019 * "carolina" + 0.019 * "county" + 0.014 * "north" + 0.013 * "home" + 0.012 * "photo" + 0.010 * "power" + 0.010 * "flooding" + 0.010 * "october"'),
(5, u'0.034 * "haiti" + 0.020 * "people" + 0.016 * "hurricane" + 0.012 * "matthew" + 0.011 * "storm" + 0.010 * "country" + 0.009 * "said" + 0.008 * "water" + 0.007 * "home" + 0.007 * "relief"'),
(6, u'0.033 * "hurricane" + 0.028 * "storm" + 0.021 * "matthew" + 0.019 * "florida" + 0.015 * "wind" + 0.011 * "coast" + 0.008 * "beach" + 0.008 * "south" + 0.007 * "friday" + 0.007 * "area"'),
(7, u'0.039 * "news" + 0.033 * "health" + 0.014 * "disease" + 0.011 * "blog" + 0.010 * "google" + 0.010 * "report" + 0.009 * "zika" + 0.008 * "french" + 0.008 * "update" + 0.007 * "site"')]

Inspecting these manually, it is obvious that topics 4, 5 and 6 are strongly related to Hurricane Matthew, and thus selecting the documents that have text that is strongly related to these topics will likely produce a well cleaned corpus.

After selecting only these documents that strongly relate to these topics, we ran LDA again to produce a new list of topics, within the documents that are about Matthew:
[(0, u'0.026 * "river" + 0.019 * "flood" + 0.018 * "county" + 0.012 * "share" + 0.011 * "warning" + 0.011 * "tornado" + 0.010 * "area" + 0.010 * "look" + 0.010 * "storm" + 0.009 * "likely"'),
(1, u'0.048 * "hurricane" + 0.036 * "storm" + 0.034 * "matthew" + 0.017 * "haiti" + 0.015 * "wind" + 0.014 * "florida" + 0.014 * "category" + 0.013 * "coast" + 0.011 * "bahamas" + 0.011 * "cuba"'),
(2, u'0.040 * "haiti" + 0.032 * "people" + 0.016 * "cholera" + 0.012 * "country" + 0.011 * "haitian" + 0.011 * "earthquake" + 0.008 * "aid" + 0.007 * "world" + 0.007 * "water" + 0.007 * "many"'),

$(3, u'0.016 * "disaster" + 0.013 * "need" + 0.011 * "help" + 0.011 * "relief" + 0.011 * "community" + 0.010 * "response" + 0.009 * "support" + 0.009 * "red" + 0.008 * "cross" + 0.008 * "emergency"')$,
$(4, u'0.021 * "friday" + 0.019 * "thursday" + 0.017 * "closed" + 0.015 * "flight" + 0.014 * "open" + 0.011 * "game" + 0.011 * "miami" + 0.011 * "scheduled" + 0.011 * "october" + 0.010 * "park"')$,
$(5, u'0.076 * "photo" + 0.068 * "hurricane" + 0.068 * "matthew" + 0.041 * "october" + 0.040 * "caption" + 0.036 * "destruction" + 0.036 * "hide" + 0.035 * "path" + 0.013 * "carolina" + 0.012 * "florida"')$,
$(6, u'0.043 * "hurricane" + 0.031 * "matthew" + 0.024 * "carolina" + 0.022 * "north" + 0.015 * "lumberton" + 0.014 * "pollen" + 0.013 * "october" + 0.012 * "people" + 0.012 * "flooding" + 0.010 * "still"')$,
$(7, u'0.048 * "hurricane" + 0.047 * "matthew" + 0.041 * "fullscreen" + 0.017 * "today" + 0.015 * "beach" + 0.010 * "florida" + 0.010 * "buy" + 0.010 * "usa" + 0.009 * "via" + 0.008 * "network"')$,
$(8, u'0.031 * "hurricane" + 0.017 * "matthew" + 0.013 * "news" + 0.011 * "right" + 0.011 * "get" + 0.010 * "email" + 0.009 * "new" + 0.009 * "time" + 0.008 * "story" + 0.007 * "live"')$,
$(9, u'0.020 * "county" + 0.020 * "carolina" + 0.018 * "north" + 0.015 * "flooding" + 0.014 * "florence" + 0.014 * "fire" + 0.013 * "sunday" + 0.012 * "news" + 0.012 * "saturday" + 0.011 * "road"')$,
$(10, u'0.019 * "storm" + 0.014 * "florida" + 0.012 * "people" + 0.012 * "say" + 0.012 * "coast" + 0.011 * "wind" + 0.009 * "hurricane" + 0.008 * "mph" + 0.008 * "could" + 0.008 * "georgia"')$,
$(11, u'0.021 * "beach" + 0.017 * "new" + 0.012 * "police" + 0.010 * "spot" + 0.009 * "florida" + 0.008 * "pier" + 0.008 * "neighborhood" + 0.008 * "nfl" + 0.008 * "tuesday" + 0.007 * "opened"')$,
$(12, u'0.034 * "wind" + 0.023 * "storm" + 0.016 * "tropical" + 0.015 * "forecast" + 0.015 * "high" + 0.014 * "weather" + 0.013 * "mph" + 0.010 * "surge" + 0.009 * "area" + 0.009 * "may"')$,
$(13, u'0.023 * "florida" + 0.020 * "said" + 0.019 * "county" + 0.019 * "carolina" + 0.018 * "state" + 0.016 * "storm" + 0.015 * "south" + 0.014 * "hurricane" + 0.014 * "thursday" + 0.013 * "official"')$,
$(14, u'0.064 * "shelter" + 0.044 * "school" + 0.039 * "county" + 0.031 * "pet" + 0.017 * "emergency" + 0.016 * "animal" + 0.015 * "open" + 0.014 * "high" + 0.008 * "resident" + 0.008 * "need"')$,
$(15, u'0.044 * "haiti" + 0.029 * "matthew" + 0.023 * "hurricane" + 0.021 * "home" +$

$0.014*"jeremie"+0.013*"people"+0.013*"storm"+0.011*"country"+0.011*$ $"across"+0.011*"full"'),$
$(16,u'0.024*"home"+0.021*"water"+0.021*"damage"+0.017*"said"+0.015*"tree"+$ $0.013*"power"+0.011*"street"+0.010*"road"+0.009*"many"+0.009*"one"')]$

Selecting sentences from the corpus that have the highest strength from each of these topics gives this result:
"Share Share Content brought to you by Mike Moss Sept. 17 Models yesterday did a fairly good job showing that heavy showers and storms would concentrate into a band or two over central NC during the night - here's a look at the radar a little after 3 AM. After passing Jamaica and Haiti, Matthew's centre was expected to pass about 50 miles (80 kilometres) east of the US Navy base at Guantanamo Bay, Cuba, where authorities evacuated about 700 spouses and children of service members on military transport planes to Florida. Aid groups especially fear cholera, a waterborne infection that has stubbornly plagued Haiti since shortly after the earthquake, when it was believed to have been inadvertently introduced by United Nations peacekeepers assigned to the country. As the ongoing international relief mission progresses and more experienced experts arrive to aid longer-term recovery and reconstruction, we anticipate U.S. military capabilities will no longer be needed, and any remaining tasks performed by the task force will be assumed by other, more experienced relief organizations. - Walt Disney World Resort Of course with Mickey's Halloween Party being canceled tonight and tomorrow, Disney is offering guests with tickets to these parties 1-day tickets to the park or tickets to Mickey's Very Merry Christmas Party. Hide Caption 23 of 80 Photos: Hurricane Matthew's path of destruction Adam and Alec Selent watch waves crash over a retainer wall at the Ocean Club condominiums in Isle of Palms, South Carolina, on October 7. Pat McCrory says more damage is still to come for many people in the eastern part of North Carolina as the state faces its ninth day of Hurricane Matthew's aftermath. Malcolm Denemark, Florida Today, via USA TODAY Network Fullscreen None Workers started before dawn removing umbrellas and the colorful rocking chairs that line the Cocoa Beach, Fla. pier in preparation for Hurricane Matthew. Free Newsletters Your daily news briefing from the editors of CT. More Newsletters Reply on Twitter Join the conversation on Facebook Christianity Today Direct (Daily) Get the most recent headlines and stories from Christianity Today delivered to your inbox daily. 11 Man killed by police in shootout; two injured September 09, 2018 Greenville Police Department officers shot and killed a man after they, along with other officers, observed him shooting into a crowd in an alleyway early Sunday, the department reported. As evening fell, the winds picked up along Vero Beach, midway between West Palm Beach and Cape Canaveral, stripping away palm fronds, ripping

awnings and blowing sand that stung the face. Beachgoers Try To Steal Pot After Marijuana Bricks Wash Ashore Florida Beach Flagler County authorities are working with federal officials after bundles of marijuana washed ashore and multiple beachgoers tried taking the packages. We can expect snow totals to be minimum maybe accumulating an inch of snow, but even that will be unlikely due to a misplaced jet stream, gusty surface winds, and extremely warm surface temperatures right before the snow is expected to fall. As of 2 p.m., the only county that had declared a mandatory evacuation was Brevard County – where a mandatory evacuation for the county's barrier islands takes effect at 3 p.m. Scott tweeted that tolls on State Road 528 in Brevard County had been suspended, but his office hasn't mentioned any other affected roadways. Ulster County Animal Response Team The Ulster County Animal Response Team is a mobile emergency animal shelter in Ulster County New York that will be set up next to the county emergency shelter to care for pets. Rescue workers in Haiti struggled to reach cutoff towns and learn the full extent of the death and destruction caused by Hurricane Matthew as the storm began battering the Bahamas on Wednesday and triggered large-scale evacuations along the U.S. East Coast. Water flooded roads even blocks away from the ocean In town, a tree collapsed across a Springfield road, but mere feet away, a gas station and convenience store remained open, and a customer stepped inside."

6.4 CONCLUSIONS

The method of using LDA topics to filter documents by topics that are of interest seems to be a very well motivated and successful idea. The documents selected by this process were also used to improve the other methods in other units. The flaw, of course, is the need for a human in the loop to select which topics seem to be good and seem to be bad. This introduces an undesirable amount of subjectivity, which is not ideal for the creation of an automated summary.

Finding the LDA topics for the filtered data seems like it may be useful in summarizing the data, although the somewhat naive approach of selecting sentences with high strength in each topic proved to be unsuccessful due to other noise in the data. Some of the topics do appear to be related to different aspects of the hurricane, some topics including words like 'damage' and others including words that are proper noun locations. This implies that perhaps the topics chosen by LDA could be used to give a metric for the coverage quality of a summary generated by other means. For example an abstractive summary could be analyzed for how strongly it hits on each topic, and this metric would indicate its quality as a summary.

Unfortunately the sentences extracted by the topic strength method are contaminated with a lot of noise. The opening three words ‘Share Share Content’ is an indication that a lot of website hyperlink text was included where it should not have been. Additionally there are ‘sentences’ that are really multiple sentences that lack a period, so the sentence tokenizer considered them as one. These problems have plagued our project since the beginning, and no satisfactory method has been found to extract only grammatically correct sentences. Instead the approach has been to throw away sentences that are beyond a certain length or that include certain words, and this clearly has not been enough to get good results in this LDA based sentence extraction method.

Chapter 7

Unit 7 - Extractive Summary with TextRank Summariser

7.1 Introduction

In this unit, we planned on using the summarization function in the Gensim package in order to obtain an extractive summary of the collection of webpages. In order to make this summarization function effective, we must first clean the collection. The cleaning steps performed in this unit involved removing whole sentences. We first use sent_tokenize in NLTK to split sentences in a document and then to filter sentences that contained irrelevant information. We also placed several requirements on the sentences in order to make sure the sentences we processed at the end contain meaningful and relevant information:

1. Sentences must have more than four words. This requirement is set because a sentence with four words cannot contain any relevant information.

2. The sentences must be shorter than twenty five words. Longer sentences are most likely run-on sentences that would only cause problems in the clustering.

3. The sentences cannot contain any of our own custom words such as FaceBook or share, etc.

In the next cleaning step, we removed any documents that contain only two sentences since these only produce noise into our summarization of the collection. Lastly, we removed any document that does not contain the word hurricane since these documents are irrelevant to the topic.

Once the collection of webpages is cleaned using the above procedure, we used summarization in the Gensim package to summarize the collection. Initially we performed summarization on one document but it failed to yield any result. So we decided to perform the summarization on groups of documents. We divided the collection into groups of 300 documents, and summarized each of these groups individually. We then combined the results into a text file. Next, we summarized the text file to obtain an extractive summary that summarizes the content of the collection of webpages.

7.2 RESULTS

Here is our extractive summary result for the big data set:

Residents across four states on the east coast are bracing for impact as Hurricane Matthew continues to make its way toward the U.S. The storm headed toward Florida on Wednesday after causing severe devastation across Haiti and bringing heavy rainfall and strong winds to the entire Caribbean region.
Hurricane Matthew pummeled the Florida coast this morning with powerful winds, potentially devastating storm surges and torrential rain.
United States: Downgraded to a Category 3 storm, Hurricane Matthew made landfall as evacuation orders were issued in Florida, Georgia and South Carolina.
More than two million people have been evacuated from Florida, Georgia, North and South Carolina as the United States prepares for Hurricane Matthew to hit.
At 5 p.m. EDT Friday, the National Hurricane Center said Matthew had sustained winds of 110 mph, making it a very powerful Category 2 storm.
Piers in Florida and South Carolina were damaged by storm surge and heavy surf from Hurricane Matthew. Hurricane Matthew has moved on to South Carolina where it made landfall as a Category 1 storm with sustained winds of 75 mph.
Update: Hurricane Matthew has weakened slightly to a Category 2 storm with 110 mph winds off Florida and Georgia coasts.

7.3 CONCLUSIONS

The result of this unit is quite good. This gave us sufficient information for our extractive summary. Our extractive summary could be further improved with a more careful cleaning process.

Chapter 8

Unit 8 - Generating Values for Template Slots

8.1 INTRODUCTION

In order to create a template summary, we needed to have interesting and important bits of data to fill out the template. We decided to take our Named Entity results and further process them to create singular words that we could insert into the template. Our first attempt did the following:

- Get named-entity/subject pairs (via spaCy) that contain all of a list of input strings.
- Convert the obtained named-entity/subject pairs to strings, removing either the named entity or the subject based on the desired output.
- Sort by smallest length strings. Smaller strings in our experience were more on-topic.
- Use regex searching[12] to search through the ordered list for the first string that will fit in the template, and insert it.

This approach did not produce ideal results, and often no results were found or the inserted string made no sense. The main reason suspected for this is because we sorted effectively twice by strings/regex and then sorted by smallest instead of by most common. We then redesigned the algorithm to do the following instead:

- Find sentences that contain both a regex string of interest and a named entity type of interest. Save the named entity word.
- Sort the resulting list of named entity words by most common.
- Return this sorted list. You can get the strongest hit by getting the first item in the list, or use the top several items.

8.2 Results

The second algorithm consistently returns good results assuming the type of named entity is correct and the regex search string is fairly general. A sample of the results is below:

```
ne: QUANTITY  regex: wind.*mph
        results: ['140mph', '145mph', '120mph', '155mph']

ne: CARDINAL  regex: \d+.*deaths
        results: ['11', 'three', '38', '26', 'thousands']

ne: GPE       regex: damage
        results: ['Haiti', 'Florida', 'Fla.', 'Cuba', 'Roseboro']
```

8.3 Conclusions

We were pleased with the results of this template value generator. The generator produces generally good results as long as the user keeps inputs fairly general. One thing we noticed is that the outputs are very prone to manipulation via repetitive articles, a good filtering step such as our LDA filter is necessary to get good output, and the more varied the articles, the better the end result.

Chapter 9

Unit 9 - Template Summary

9.1 Introduction

In the previous unit, we generated values to insert into the template summary. For this unit, all we had to do was write the template to insert those values into. Our final result is below.

9.2 Results

Hurricane Matthew, a category 4 hurricane, made landfall on Tuesday in Haiti. The wind speeds were measured at 140mph, and the hurricane caused 11 deaths as it traveled onto land. The largest amount of damage took place in Haiti, Florida, Fla., Cuba, Roseboro, Baracoa, and Nassau. millions homes lost power. AP, UN, Reuters, Facebook, FEMA, U.N., and NOAA were working to document the damage and assist the victims. There were also reports of looting in Haiti. 175,000 people were evacuated from their homes by authorities in anticipation of the storm.
We are pleased with these results, and if we were going to invest more time in a certain area, this type of summary would be chosen. The main issues we saw here were grammatical issues related to substitution and repeat entities due to abbreviations. The algorithm we devised is very flexible and the few issues with the summary above should be solvable.

Chapter 10

Generated Summaries vs. Golden Standard

10.1 Introduction

This section will show the Golden Standard summary that another team wrote for us to try and match in quality, and we will compare our results to Gold Standard.

10.2 Gold Standard

What follows is the Gold Standard written by Team 1 to try and provide a goal for us to target.

On September 22, 2016 a mass of thunderstorms formed off Africa. On September 28, moving away from the Antilles (where it caused extensive damage to landmasses), with sustained winds of 60 mph, the storm gained tropical cyclone status when southeast of St. Lucia. It became Hurricane Matthew on September 29, when north of Venezuela and Columbia. On September 30, Hurricane Matthew became a significant hurricane, reaching Category 5, with wind speeds of 165 miles per hour, before moving to the Caribbean. It was the first Category 5 Atlantic hurricane since Felix in 2007, the deadliest since Stan in 2005, and the costliest since Sandy in 2012. Preparations, cancellations, warnings, and watches proceeded in the Windward Islands, Columbia (where one person drowned in a swollen river), and Jamaica. On October 4, Hurricane Matthew made landfall in Haiti's Tiburon Peninsula with peak winds up to 145 mph as a Category 4 hurricane, and rainfall reaching over 30 inches. Hurricane Matthew caused a humanitarian crisis in Haiti, affecting over 2 million people, many needing clean water, food, shelter, sanitation, household goods, and medical supplies; more than 800 people died as a result of the storm. Dozens of cholera treatment centers were destroyed which led to a fear that the cholera epidemic would worsen. The World Health Organization began a vaccination campaign against cholera to avoid further spreading of the disease following Hurricane Matthew's destruction. The disturbance caused by Hurricane Matthew caused the Haitian presidential election to be postponed. There were 546 fatalities in Haiti and damage costing $1.9 billion; this was Haiti's worst disaster since the 2010 earthquake. Experts say that it will take at least 5 years to restore plantations of export crops like coffee, cocoa, and fruits. Four were killed in the Dominican Republic, which had heavy rainfall. On October 5, Hurricane Matthew hit eastern Cuba, resulting in 4 fatalities and $2.58 billion in losses, especially in the GuantÃąnamo Province. The city of Baracoa suffered the most, with 25-foot waves crashing over the coast and many buildings collapsing. The Red Cross was ordered to rescue those trapped in collapsed buildings. On October 5 and 6, Matthew then moved through the northern islands of the Bahamas, directly hitting Grand Bahama. Peak-winds were 145 miles per hour, and peak rainfall was 20 inches. As Hurricane Matthew traveled through the Bahamas, the hurricane was classified as Category 3. There were no fatalities in the Bahamas. In response to the destruction Hurricane Matthew had caused, Florida Governor Rick Scott ordered the evacuation of 1.5 million people living in potential danger zones. President Obama also declared a state of emergency in both Florida and South Carolina, calling in the Department of Homeland Security to prepare relief measures. On October 6 and 7, Matthew moved along the entire length of the Florida Atlantic coastline, first as Category 3. It was downgraded to Category 2 late on October 7, then to Category 1 on October 8. The storm killed 12 people in Florida and 3 in Georgia. Over 1 million lost power in Florida, and more than half a million in Georgia and South Carolina. It made landfall on October 8 at Cape Romain National Wildlife Refuge near McClellanville, South Carolina, with 85 mph winds. 30 people were killed in North and South Carolina and 2 in Virginia; these states suffered flooding due to torrential rainfall, with portions of Interstate 95 shut down. Robeson County schools did not reopen till 31 October. The peak rainfall for Hurricane Matthew in the United States was about 20 inches in North Carolina. On October 9, turning away from Cape Hatteras, North Carolina, Matthew reentered the Atlantic as a post-tropical storm and moved along the eastern coastline towards Canada. The total damage to the United States is estimated at more than $10 billion, with more than 40 people dead. On October 10, Hurricane Matthew's remnants hit Canada, specifically Newfoundland and Cape Breton, causing flooding, strong winds (over 56 mph), and power outages. There were no fatalities, but the total damage to Canada is estimated at $7.6 million. Most of this damage affected roads in Newfoundland. In Canada, Hurricane Matthew was absorbed by a cold front, and completed its 3,000-mile-long journey of destruction. Recovery and reconstruction continued into 2017 and 2018.

10.3 ROUGE Scores

We compared the Rouge scores for both our template summary and our extractive summary to the Gold Standard above to quantify how good the various types of summaries were. According to the ROUGE scores, our extractive summary was superior. We think that an expanded template summary (with more slots) should be sufficient to beat the extractive summary, however.

	Template Summary	Extractive Summary
ROUGE-1	0.16129	0.16129
ROUGE-2	0.03333	0.03333
ROUGE-L	.06452	.12903
ROUGE-SU4	.03529	.03529
MAX ROUGE-1 among sentences	.50000	.60000
MAX ROUGE-2 among sentences	.21053	.25000
Entity Coverage	6.25%	11.46%

Chapter 11

Conclusion

11.1 Analysis of Tools

We found that the most useful tools were spaCy, Gensim, and NLTK. Initially when we started our project we aimed to use the Zeppelin tool contained in a Docker image, which had PySpark installed. After making code here, we planned to upload it to a Hadoop cluster to run our code quickly on our large data set. We found that Spark was unintuitive and trying to make it work was more trouble than it was worth. For this reason we began to transition away from Spark and rely more simply on Python code executed on our local environments. After cleaning and filtering our documents, we found that even the big data set did not pose too big a challenge for any of the tasks we attempted.

Gensim proved to be essential for our cleaning of the data, spaCy was essential for generating our template summary, and NLTK was useful for the simpler tasks and aiding in our understanding.

11.2 Analysis of Methods

Since we tried many methods, we found that some contained useful information for generating a summary and others were less effective. Ultimately the best method we found for generating a summary was to gather information from regular expressions used on POS tagged words and named entities. We found that a wide variety of specific information can be found with this method, and it can be inserted into a well crafted template.

We found that even some of the simpler tasks, such as counting POS tagged nouns and verbs, created a good list of keywords, although there were some issues with generic words. Filtering the words by an importance metric such as tf-idf, proved to be useful.

For removing documents that were irrelevant to Hurricane Matthew, we found that filtering documents by LDA topic models, was a well performing method. It seems likely that this may not be the best method; a trained classifier may perform as well or better, but for our purposes filtering by LDA topic was sufficient.

11.3 Lessons Learned

The biggest lesson learned by our team was the magnitude of the difficulty and complexity of natural language processing. It was interesting to know there are so many unsolved problems, and despite the things we did learn, we found that there is so much more left to learn.

Another important takeaway was that we should have abandoned methods and tools that were not working out sooner. A particular example of this is the attempt of our team to use Spark. None of the group members were proficient in it, so we all attempted to learn how to use it at the start of the semester. Upon finding that Spark is complicated and not always user friendly, we pressed on and spent much time debugging. We should have played to our strengths and used tools we were familiar with instead of forcing ourselves to use a tool that was only giving us headaches.

In the more practical sphere of things we learned, we learned how important clean data is, and that data sets must be very thoroughly pre-processed before any kind of natural language processing will give remotely correct results.

Chapter 12

User Manual

12.1 Introduction

The user manual explains the various files we created, along with their dependenicies, inputs, and outputs. For those interested in learning more about how to modify the included files, we recommend reading both the user manual and the developer manual.

12.2 Dependencies

All files require Python 3 to run. To set up dependencies, we recommend installing PySpark, spaCy, Gensim, and NLTK from pip. You should also run the following commands at a terminal:

```
python -m spacy download en_core_web_sm
```

12.3 Files

LDAFiltering.py: This file includes the method for obtaining the LDA topics from the collection of documents. To filter the documents, the good topics need to be identified manually, and the code run again with the parts for saving a new JSON file uncommented.

LDASentenceExtraction.py: This file will automatically create an LDA model from the document and output a list of sentences that correspond to those topics.

POSTag.py: This file will output a list of the top 10 most common nouns and verbs in all of the documents.

wordFrequency.py: This file includes cleaning data steps and computing TF and TF-IDF. The cleaning data steps include removing empty documents, duplicate documents, stop words, and custom stop words that we defined.

summaryTextRank.py: This file works by cleaning data, dividing the collection into groups of 300 documents and summarizing these groups, and finally outputting these summaries into a text file. The cleaning step involves removing empty documents, removing duplicate documents, removing documents that have less than 3 sentences, and removing sentences with less than 3 words or greater than 26 words (run-on sentences). We used the Gensim summarization package to summarize the documents in each group. We appended the summaries of the groups into a new document for the final summarization.

allsummary.py: This script produces the extractive summary for the new document outputted from summaryTextRank.py. We use the Gensim summarization package to extract the summary.

listify.py: The various filtering tools we wrote all rely on Spark, which does not output valid JSON files. This script takes invalid Spark JSON and turns it into JSON that can be read by Python's JSON library. This script reads on standard input and writes to standard out. Run it as follows:

```
python listify.py <infile >outfile
```

namedEntity.py: This script takes a cleaned JSON file (sent through listify.py) and outputs a .txt file (loosely based on the CSV format) that contains the results of a spaCy named entity analysis of each article. You need to change hardcoded input file strings, and output is written to standard out by default, and should be redirected to a file. Run it as follows:

```
python namedEntity.py >outfile
```

templateSummary.py: This script takes the output from the namedEntity.py script and creates a template summary. The template and input file are hard coded, and output is written to standard out by default. Run it as follows:

```
python templateSummary.py >outfile
-or to print to console-
python templateSummary.py
```

12.4 FLOWS

Word Frequency:

1. Run wordFrequency.py. The output will be the results of TF (Term Frequency) and TF-IDF (Term Frequency-Inverse Document Frequency).

Generate Extractive Summary:

1. Run summaryTextRank.py to obtain the summary for each 300 documents.

2. Run allsummary.py to obtain the extractive summary from the new document.

Generate Template Summary:

1. Obtain the WARC and IDX files for your data set.
2. Run the Spark WARC to JSON converter [1]
3. Run the LDA cleaner
4. Run listify.py on the output from the LDA cleaner
5. Run namedEntity.py on the JSON output from listify.py
6. Run templateSummary.py on the txt file from namedEntity.py.

Chapter 13

Developer Manual

13.1 Introduction

This manual is a recommended read for those interested in further modifying or expanding upon our work. You should be familiar with the ideas outlined in the User Manual before proceeding, as this manual will build from it.

13.2 Tools Used

- NLTK - a Python library designed to collect several different resources related to natural language processing. It's organized as a collection of modules that do not rely on one another. This means that the authors can pack a lot of functionality in, but we found it to not always have the most coherent API.
- spaCy - A Python library that mirrors a lot of the functionality of NLTK, but takes a different approach. NLTK provides several algorithms for each natural language operation, while spaCy provides only the best one available. This leads to simpler development, better results, and fewer intermediary data formats.
- Gensim - A Python library for topic modeling, and similarity-related retrieval with large corpora. Gensim includes implementations of TF-IDF, word2vec, doc2vec, latent Dirichlet allocation (LDA), and summarizer.
- Spark - Created by the Apache Foundation, Spark is a framework for writing parallelized big data programs. The main version of Spark requires Scala, but we mostly made use of its PySpark derivative, which is a Python port.

- Docker - A containerization tool. Docker allows developers to build applications with easily maintainable and swappable layers, while getting many of the isolation benefits of virtual machines. We used Docker to host a local Zeppelin instance for development.
- Zeppelin - An online code notebook for experimenting with code. We used it to work with Spark, but dropped it when moving away from Spark code due to Zeppelin not working well with Git due to how zeppelin stores its notebooks.
- Git - Version control system. Used for synchronizing code between members and sharing code with other teams.

13.3 Optimization

The portions of code we have written that take a non-trivial amount of time to run are optimized to take advantage of multiple-processor systems. One file in particular that should be discussed is namedEntity.py. This script can take a long time to run due to running spaCy's main nlp() function on each article. The code is set up as a process pool, with the default number of workers spawned being 40. This default should take advantage of nearly any system, and limits memory consumption to about 4GB. For systems with fewer processor threads, this number can be reduced to cut down on memory usage, and if RAM allows, you should increase this number to the allowable number of processor threads if that is more than 40.

Appendix A

Gold Standard

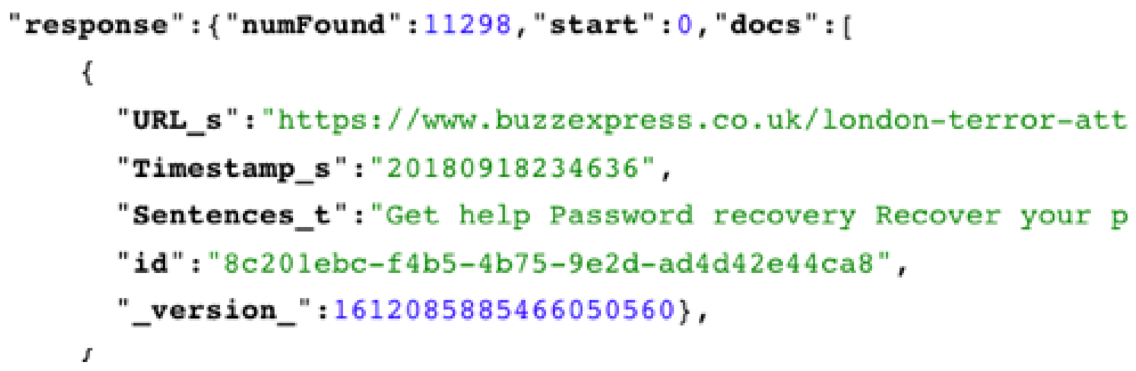
A.1 Approach

We were assigned to draft a Golden Standard Summary for Team 4 which was working on ’The Westminster attack of 2017’. The basic idea was to retrieve optimum and relevant information regarding the proposed topic. The approach we followed to retrieve information was firstly to fulfill the main ingredients of a good summary, which could be broken down into following subtopics:[13]

1. WHAT: what happened
2. WHEN: date, time, other temporal placement markers
3. WHERE: physical location of the attack
4. ATTACKER: individuals or groups responsible for the attack
5. WHY: reasons for the attack
6. CASUALTIES: deaths, injuries caused by the attack
7. DAMAGES: All sort of damages caused by the attack
8. AFTERMATHS: countermeasures, rescue efforts, prevention efforts, other reactions to the attack (e.g., police investigations)

Secondly, A rough outline was made, keeping in mind the structure of the summary [14]. To be considered as a good summary, it should encompass each aspect of information available. Thus it was important for us to thoroughly understand the topic.Thus, in order to get the gist of topic with neutrality, the Wikipedia page of "2017 Westminster attack" was taken into the consideration [15]. After designing the structural template, the next step was to get the content for each sub-heading.

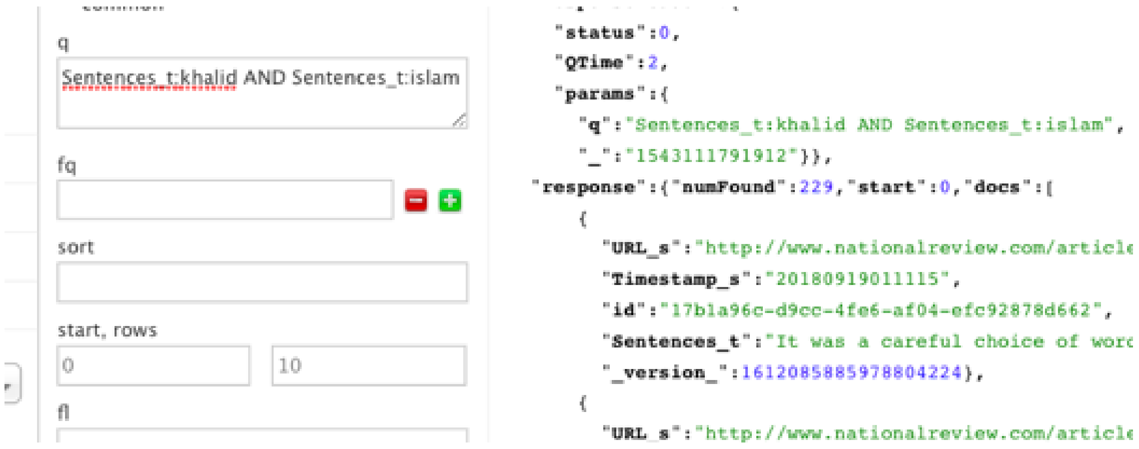
Queries made on Solr pertain to the subtopics listed above [16]. Records contain column names among which most important were of Sentence and Timestamp.



```
"response":{"numFound":11298,"start":0,"docs":[
    {
      "URL_s":"https://www.buzzexpress.co.uk/london-terror-att
      "Timestamp_s":"20180918234636",
      "Sentences_t":"Get help Password recovery Recover your p
      "id":"8c201ebc-f4b5-4b75-9e2d-ad4d42e44ca8",
      "_version_":1612085885466050560},
    ,
```

Figure A.1: Solr Records for Team4

There were numerous articles and it was difficult to extract information which was absolutely flawless. For example, we encountered articles which were from the next few days after the attack and may contain inadequate and unreliable information, and so were discarded. Queries made on Solr consist of Boolean function to extract all available information in best form [17].



q: Sentences_t:khalid AND Sentences_t:islam

fq:

sort:

start, rows: 0 10

fl:

```
  "status":0,
  "QTime":2,
  "params":{
    "q":"Sentences_t:khalid AND Sentences_t:islam",
    "_":"1543111791912"}},
"response":{"numFound":229,"start":0,"docs":[
    {
      "URL_s":"http://www.nationalreview.com/article
      "Timestamp_s":"20180919011115",
      "id":"17b1a96c-d9cc-4fe6-af04-efc92878d662",
      "Sentences_t":"It was a careful choice of worc
      "_version_":1612085885978804224},
    {
      "URL_s":"http://www.nationalreview.com/article
```

Figure A.2: Solr boolean query

We took help from Wikipedia to help us in determining the truthfulness of the articles and determine the best structure for the summary. Eventually we found enough information from Solr articles to compile them to a best effort Golden Standard summary consisting of all the valuable information regarding the topic.

Lastly some trivial information was edited while maintaining the original structure of the summary at the same time, to give a best possible Gold Standard in two pages as per requirements.

A.2 The Summary

On Wednesday March 22nd 2017, an attack took place at the Houses of Parliament in London. A car was driven at high speed across Westminster Bridge - hitting victims as it went. The horror began at the south London end of Westminster Bridge when a driver in a Hyundai Tucson jumped onto the pavement at around 2.30pm, mowing down pedestrians. He continued along the pavement, plowing through at least a dozen victims in total, before coming to a stop at the Houses of Parliament where he crashed his vehicle into railings. The driver then, armed with a knife, tried to make his way into the Parliamentary estate. The assailant killed an unarmed police officer before being shot dead by firearms officers just inside the gates. The attacker was later identified as 52 year old Briton Khalid Masood. It is strongly believed that Masood was inspired by Islamic extremism. [23]

Five died in the rampage, including a police officer, and more than 50 people were injured. The five victims who lost their lives were PC Palmer, mother-of-two Aysha Frade, US tourist Kurt Cochran, Romanian visitor Andreea Cristea, and pensioner Leslie Rhodes from south London. Aysha Frade, who worked as a Spanish teacher, and had only just left her place of work at the nearby DLD College London, was making her way across Westminster Bridge en route to collect her two children from school when she was mowed down by the terrorist. Kurt Cochran was an American on European tour with his wife. In his 50s, he was killed while walking on Westminster Bridge. Cochran was visiting London from Utah to celebrate his 25th anniversary with his wife Melissa, who was among the injured. Romanian Andreea Cristea, 31, who was visiting London with her boyfriend Andrei Burnaz, was knocked into the river Thames after being struck by the car driven by Khalid Masood as she strolled along Westminster Bridge with her boyfriend. She later died from her injuries. Leslie Rhodes, 75, a retired window cleaner, was on the way to a hospital appointment at the time. The police officer killed was PC Keith Palmer, 48, an unarmed police officer who was on duty with the Parliamentary and Diplomatic command. The 52-year-old attacker was shot by armed police and was the first casualty to arrive at St Mary's Hospital in Paddington, where he was pronounced dead. Among those injured were people from Ireland, America, South Korea, Romania, France, Poland and Germany. They were taken to St. Thomas' Hospital, which faces the Palace of Westminster across the Thames and also to King's College Hospital and the Royal London Hospital [20].

Born in Dartford ,Kent on Christmas Days As Adrian Russell Elms, Masood was brought up in Rye, East Sussex and later attended secondary school in Tunbridge Wells in Kent. Most recently he was living in the West Midlands. His father played no part in the newborn's life and when his mother Janet married two years later, he was given his step-father's surname, Ajao. When he was 16, he dropped out of school and by 18 he was described as a heavy drug user. In 2000, he was sentenced to two years in prison for stabbing and slashing the face of a cafe owner at Northiam in Sussex. In 2003, he was sentenced to a further six months in prison for possession of an offensive weapon following another knife attack in Eastborne in Sussex. As well as these two prison terms, Masood had convictions for public order offenses going back to 1983. Masood was not known to be a religious person and frequently went to pubs [19].

Masood reportedly converted to Islam in 2005. He was described by police as a criminal with a twenty year record of offending. He changed his name to Khalid Masood after he converted to Islam. The 52-year-old had recently split from his partner, Jane Harvey, following bitter rows after she refused to allow her daughter to move from Kent. The couple met at a Tunbridge Wells pub in 1991 before separating nine years later when he was jailed. The couple had two daughters - aged 19 and 24. The younger of the two still lives with her mother, who is a director of a chemical company in Kent. The friend said the tumultuous relationship was punctuated by Masood's violence towards others, who was then known by his birth name. Masood masked his life of crime with a CV which claimed he was an experienced English teacher who had worked across the world. The document lists Masood as a university-educated English teacher with experience working in places such as Saudi Arabia and Luton. The Saudi Embassy in the United Kingdom confirmed Masood had visited Saudi Arabia three times, including two stints teaching English there [18].

The Metropolitan Police initially believed the attack was inspired by international terrorism. On March 23, The Amaq News Agency, affiliated with ISIS, took credit for the killings and announced that the attacker was "a soldier of the Islamic State, executing the operation in response to calls to target citizens of coalition nations", but has been unable to prove officials were in any contact with Masood. Describing him as a terrorist, the Metropolitan Police confirmed that he acted alone. Later, on March 27, Neil Basu, Deputy Assistant Commissioner of the Metropolitan Police and Senior National Coordinator for UK Counter-Terrorism Policing, announced that Masood clearly had an interest in Jihad, but they found no evidence he was linked with any terrorist Organization. Six property raids and eleven arrests were made, but no charges were filed and all suspects were released without charges.

Bibliography

[1] GitHub VT_Fall18_CS4984-CS5984,
https://github.com/xw0078/VT_Fall18_CS4984-CS5984
Date Accessed: Nov 2018

[2] Matthew Honnibal: A Good Part-of-Speech Tagger in about 200 Lines of Python,
https://explosion.ai/blog/part-of-speech-pos-tagger-in-python
Date Accessed: Nov 2018

[3] Gensim: Topic modelling for humans, *https://radimrehurek.com/gensim/*
Date Accessed: Nov 2018

[4] Natural Language Processing with Python, *http://www.nltk.org/book/*
Date Accessed: Nov 2018

[5] Machine Learning Plus: Topic Modelling with Gensim,
https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/
Date Accessed: Nov 2018

[6] spaCy: Industrial Strength Natural Language Processing in Python, *https://spacy.io/*
Date Accessed: Nov 2018

[7] WordNet Interface, *http://www.nltk.org/howto/wordnet.html*
Date Accessed: Nov 2018

[8] Michael Lesk: *Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone*, SIGDOC '86

[9] Word Sense Disambiguation, *http://www.nltk.org/howto/wsd.html*
Date Accessed: Nov 2018

[10] Python Multiprocessing Library, *https://docs.python.org/3.6/library/multiprocessing.html*
Date Accessed: Nov 2018

[11] Python JSON Encoder and Decoder, *https://docs.python.org/3/library/json.html*
Date Accessed: Nov 2018

[12] Python Regex HowTo,*https://docs.python.org/3/howto/regex.html*
Date Accessed: Nov 2018

[13] TAC 2011 guided summarization Task guideline,
https://tac.nist.gov//2011/Summarization/Guided-Summ.2011.guidelines.html
Date Accessed: Nov 2018

[14] How to write a summary, *https://depts.washington.edu/owrc/Handouts*
Date Accessed: Nov 2018

[15] 2017 Westminster attack, *https://en.wikipedia.org/wiki*
Date Accessed: Nov 2018

[16] Solr Tutorial, *http://www.solrtutorial.com/solr-query-syntax.html*
Date Accessed: Nov 2018

[17] 10 tips for better search queries in Apache Solr,
http://www.solrtutorial.com/solr-query-syntax.html
Date Accessed: Nov 2018

[18] Why did the Westminster terror attack happen?,
https://www.presstv.com/Detail/2017/04/09/517339/Westminster-terror-attack-Why did the Westminster terror attack happen?
Date Accessed: Nov 2018

[19] Westminster attack: Could Khalid Masood have been stopped?,
https://www.bbc.com/news/uk-45733486
Date Accessed: Nov 2018

[20] Westminster attack: Everything we know so far about the events in London,
https://www.telegraph.co.uk/news/2017/03/22/westminster-terror-attack-everything-know-far/
Date Accessed: Nov 2018

[21] UK Police Officially Release Image Of TerroristKhalid Masood,
http://worldtodaypress.com/2017/03/westminster-terror-attackuk-police-officially-release-WestminsterTerrorAttack

Date Accessed: Nov 2018

[22] The terrorist, the bloody aftermath and his victims - in pictures,
http://www.telegraph.co.uk/news/2017/03/22/westminster-attack-pics/Londonattack
Date Accessed: Nov 2018

[23] How-London-terror-unfolded?
https://www.dailymail.co.uk/news/article-4339724/How-London-terror-unfolded.html
Date Accessed: Nov 2018