CS4984/CS5984
Big Data Text Summarization

Generating an Abstractive Summary for the #NeverAgain Gun Control Movement

Authors

Anuj Arora
Chreston Miller
Jixiang Fan
Shuai Liu
Yi Han

Instructor: Dr. Edward Fox

Department of Computer Science
Virginia Tech
Blacksburg, VA 24061

December 11, 2018

Table of Contents

List of Figures

List of Tables

1 Abstract

When you are browsing social media websites such as Twitter and Facebook, have you ever seen hashtags like #NeverAgain and #EnoughIsEnough? Do you know what they mean? Never Again is an American student-led political movement for gun control to prevent gun violence. In the United States, gun control has long been debated. According to the data from the Gun Violence Archive (http://www.shootingtracker.com/), in 2017, the U.S. saw a total of 346 mass shootings. Supporters claims that the proliferation of firearms is the direct spark of a series of social unrest factors such as robbery, sexual crimes, and theft, while others believe the gun culture represents an integral part of their freedom. For the Never Again Gun Control Movement, we would like to generate a human readable summary based on deep learning methods so that one can study incidents of gun violence that shocked the world such as the 2017 Las Vegas shooting, in order to figure out the impact of gun proliferation.

Our project includes three steps: pre-processing, topic modeling, and abstractive summarization using deep learning. We began with a large collection of news articles associated with the #NeverAgain movement. The raw news articles needed to be pre-processed in multiple ways. An ArchiveSpark script was used to convert the WARC and CDX files to a readable and parsable JSON. However, we figured out that at least forty percent of the data was noise. A series of restrictive word filters were applied to remove noise. After noise removal, we identified the most frequent words to get a preliminary idea if we were filtering noise properly. We used the Natural Language Toolkit's (NLTK) Named Entity chunker to generate named entities, which are phrases that form important nouns (people, places, organizations, etc.) in a sentence.

For Topic Modeling, we classified sentences into different buckets or topics, which identified distinct themes in the collection. The Latent Dirichlet Algorithm does not directly take the documents as inputs. They have to be cleaned and tokenized (broken down into individual words). It had to be converted into a vector for each article in the collection. We chose to use the Bag of Words (BOW) approach. The Bag Of Words method is a simplifying representation used in natural language processing and information retrieval. In this model, text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity.

According to topic modeling, we needed to choose the number of topics, which means one must guess how many topics are present in a collection. There is no foolproof way of replacing human logic to weave keywords into topics with semantic meaning. To address this we tried the coherence score approach. Coherence score is an attempt to mimic the human readability of the topic, and the higher the coherence score, the more "coherent" the topics are considered. The last step for topic modeling is Latent Dirichlet Allocation (LDA). Latent Dirichlet allocation (LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. Compared with some other algorithms, LDA is a probabilistic one, which means that LDA is better at handling topic mixtures in different documents. In addition, LDA identifies topics coherently whereas the topics from other algorithms are more disjoint.

After we had our topics (three in total), we filtered the article collection based on these topics. What resulted was three distinct collections of articles on which we could apply an abstractive summarization algorithm to produce a coherent summary. We chose to use a Pointer-Generator Network (PGN), a deep learning approach designed to create abstractive summaries, to produce said summaries. We created a summary for each identified topic and performed post-processing to produce one summary that connected the three topics (which are related) into a summary that flowed. The result was a summary that reflected the main themes of the article collection and informed the reader of the contents of said collection in less than two pages.

Topic	Article Count
Gun Control Policy	1680
Movements stemming from shootings	1325
Santa Fe High School Shooting	664

Table 1: The number of articles per topic identified by LDA.

2 Introduction

This project was for Dr. Edward Fox's Big Data Text Summarization class (CS 4984/5984) taught in Fall 2018. We were tasked with summarizing a large collection of news articles on the NeverAgain movement. Overall, this movement is about gun violence, specifically in schools. The articles span three main topics. First is policy behind gun control. The second is movements that began because of the shootings. Lastly, is the specific Santa Fe high school shooting.

The task of automatically summarizing text is a challenging topic of research. With the amount of new data available every day, understanding what is going on in the world is a daunting task, and one can be overwhelmed. The ability to summarize large amounts of data alleviates this task and allows consumers the opportunity to be informed of day-to-day events.

This course introduced many technologies available to aid in creating our summary. These technologies range from usage of clusters (a Hadoop cluster managed by Dr. Fox's lab or Advanced Research Computing (ARC) [1] clusters), various programming languages meant to run on clusters, and techniques to approach computational linguistics.

We approached the problem from a deep learning perspective. The first step was to understand our collection and identify the main themes (topics). We first perfomed cleaning to remove irrelevant articles. We then applied topic modeling through Latent Dirichlet Allocation (LDA) to identify the three main topics. Then we took the collection for each topic and applied a Pointer-Generator Network (PGN). This is a deep learning technique that is able to select different sections of text and also generates out-of-vocabulary (oov) words to provide a summary. The result for each topic was a summary of the collection that required some post-processing to streamline the summary. Along with this task, we also were assigned another collection (Dakota Access Pipeline Protests) to make a human created summary called the Golden Standard for that collection.

Our article collection was initially around 12,000 articles. That was reduced to around 3600 after duplicates, empty articles, and irrelevant articles identified through LDA filtering, were removed. Table 1 gives a breakdown of the number of articles per topic. Our solution was a mixture of Python 2 github.com repositories we forked and customized scripts, also written in Python 2.

We forked the code by Abigail See [15] for our PGN and also forked her repository [14] that processed articles into the needed binary format for the PGN. We did not modify the PGN code but used it to better understand the hyperparameters and have our own workspace for the PGN. The article converter we modified heavily to address our needs and made it more extensible. This included removing dependencies on the collection the PGN was trained on. The modified version allows one to take a collection of articles and create the necessary binary format in a few simple steps. We shared this modified version on github.com [11] and multiple teams in the class used it.

The rest of this report is organized as follows. We begin by describing our methodology for generating a golden standard for another team's collection. After this we provide a literature review and then dive into great detail on our design and the implementation of our system. Then we provide a user manual so others are able to use our system. We also include a developers manual to explain our system's architecture, how we processed our data, how to install any necessary

software and libraries, plus future work directions. Lessons learned is discussed next, where we detail the time line of our project, challenges faced, and approaches we tried. We then conclude the report and provide acknowledgements.

3 Golden Standard Methodology

Here we describe the methodology followed when creating the Golden Standard for the Dakota Access Pipeline protests (NoDAPL). We organize the golden standard summary based on three steps: background research, writing process, and final update.

To begin, we first read the Wikipedia page [2] to get to know the basic idea of what the Dakota Access Pipeline protests are, and how Wikipedia [3] organizes the summary. During week 7 of our class we had a guest speaker, Michael Horning, a professor from the Department of Communication at Virginia Tech, who discussed how to write a good summary. Given his advice, our summary should include:

- Who, what, where, when, why, and how
- A single sentence
- Generally no more than 35 words
- Place attribution at the end of the story
  - People who carry weight/prominence should be named

We then searched each Wikipedia subtitle using the Solr Index system to find related articles in the database and figure out the differences that need to be highlighted. Every team member also read several news articles about NoDAPL. We also reviewed the official website, and social tools such as Facebook, for Dakota Access Pipeline protests. After we finished all the preparation work, we shared our own views on what should definitely be included in our gold standard summary, what should not, what may need a general mention, which parts should be highlighted, and any discrepancies from different articles.

Based on the discussion, we started our write-up. The first thing we did was to design an outline. Here is the outline designed for our golden standard summary:

- Background
  - Location
  - Group of people/Organization
  - Purpose of the pipeline
  - Contention
- Timeline
  - When the Dakota Access Pipeline project started
  - When the sacred stone camp was established
  - The path of the event
  - Has the problem been solved or does the protest still exist
- Detail on Affected area

  - The environment of the affected area
  - The damage of the program to the local environment
- Protester
  - The concerns of the protester Indigenous youth groups
- Sacred Stone Camp
  - Who sponsored it
  - Who has responsibility for the camp

The outline helped us focus and organize our ideas. After that, based on our outline, we picked three to five reliable sources to read through. We also used the built-in summary tools in Mac OS Mojave, where you can select a block of text, then right click, and select service ¿ summarize. (You need to enable the summarize feature in the settings first) to aid our work. After reading our identified articles, we started to write the summary. Each team member wrote one or two sections separately, after which we compiled them, and went through the whole summary together. When the draft became too long, we realized our outline had become too specific. So we decided to cut some less important material and to make it more abstract rather than focusing on one or two specific events.

One of the challenges we encountered was while we were looking for sources about NoDAPL, there are so many articles with bias because it is a protest. A lot of people boycotted the government decisions, and some clashes turned violent. Thus, we needed to manually filter sources and rewrite some phrases to summarize the facts. In the end, we had a nicely sculpted summary that informed the reader of the timeline of the events and the objective facts.

4 Literature Review

Text summarization has been an active area of research, especially within the last few years. Summarization can be broken down into two categories: extractive and abstractive. Extractive is where an algorithm selects "important" or "relevant" sentences from the source text and arranges them verbatim into a sequence to make a summary. Abstractive algorithms are able to summarize text through created out-of-vocabulary (oov) words to generate a summary that is closer to how humans summarize. The summarization performed and presented in this work is abstractive summarization through the use of a Pointer-Generator Network (PGN) [16]. Ironically this was the first summarization technique suggested to us by Dr. Fox through the class's Canvas website. We searched on GitHub for the source code, which is available here [15] using Tensorflow. We also found a PyTorch version of the PGN [9], but we found the Tensorflow version to produce better summarization results. We then did a broader search on GitHub to see what else was available. Our interest was to not build something that has already been built and tested, hence, looking for open source options. We searched for "abstractive text summarization" giving us many options, one being a sequence to sequence model called RLSeq2Seq [7] based off of the paper [8]. We were unable to sucessully get this summarization technique working. In the end, we chose the PGN as it performed well.

5 Design and Implementation

Much effort was required in the creation and implementation of our approach and system. In this section we provide, in great detail, how we approached processing our data and implementation of what we created.

5.1 Raw Data Conversion and Indexing

The raw data that was provided to the teams was in WARC and CDX formats. These are typical formats for saving data from web crawlers. The website is crawled and the content is saved in a WARC file. The CDX file provides additional information about the time of crawled and the URL crawled, and together the WARC and CDX files provide a complete picture of the web crawling.

The Scala script *ArchiveSpark_sentence_extraction* (provided) was used to convert the WARC and CDX files to a readable and parsable JSON. The next step was to index this JSON on Apache Solr to maintain a central search engine for quick access and have the ability to search by keyword(s).

5.2 Noise Removal

This was one of the most challenging parts of the entire project. At least forty percent of the data was expected to be noise, and this was confirmed by our initial summaries. The summaries had massive topic variations, from gun control to anti-semitism. As we discovered later (after a search on Twitter), this was because the *NeverAgain* hashtag was used in various connotations throughout social media and not just in relation to high school shootings.

There were three attempts to filter the noise and each one was an improvement over the previous one. These iterations (in order) are listed below:

- **Iteration 1**: A simple article filter based on the keywords *gun, shoot, shooting, school, high, violence* was used. This reduced the number of useful articles from approximately 12,000 to 7,000. However, the summary from the PGN was still mostly noise.
- **Iteration 2**: One of the teams posted a modified *ArchiveSpark_sentence_extraction* script which kept the HTML tags and used the jusText package to keep only the body of the webpages and not the irrelevant hyperlinks, redirects, and video/audio/image embeddings. This further reduced our useful article count to approximately 5,000. This made the summary significantly better, but we still had a lot of irrelevant topics like anti-semitism protests and gay marriage rulings. The function used is shown below:

```
def HtmlClean(rawtext):
    #Takes the input as a string with HTML tags included
    #Checks each paragraph for whether it's meaningful text or not
    #Outputs the text as a string with HTML tags and nav elements removed
    paragraphs = justext.justext(rawtext,justext.get_stoplist("English"))
    return ' '.join([p.text for p in paragraphs if not p.is_boilerplate])
```

Figure 1: HTML Clean Function

- **Iteration 3**: This attempt used both the previous attempts plus a more restrictive filter, i.e., only those articles with one of the keywords *gun, guns* were kept. This reduced our useful article count to approximately 4,000 and our summary started looking much better and relevant.

This marked the end of our noise removal attempts. We decided to stick with our last iteration, as it led to relevant and useful summaries. In addition, since we were allowed to post-process our summaries, building a classifier would not have yielded significantly better results. The final noise filter code is shown below:

```
dfpd=pd.read_json("C:/Users/anujr/docker/cs5984/share_dir/dfcleanedbig.json",
orient='records',lines=True) %Loading the HTML tagged (and then cleaned JSON)

lis=['gun','guns'] %words we want in our articles

sel=list(map(lambda x: dfpd.text.str.contains(x,flags=re.IGNORECASE),lis))
boo=pd.Series(np.repeat(False,len(dfpd)))

for i in range(len(sel)):
    boo=boo|sel[i]

dfhtmpd=dfpd[boo]
```

Figure 2: Noise Filter Code

5.3 Data Cleanup, Parts of Speech, and Tokenization

- **Data Cleanup**:
  - **Stopwords**: Stopwords are words that add no unique semantic value to a sentence, article, or document. They cannot be included in the keywords of a document, and a search engine usually ignores them. They serve no purpose for frequent words analysis, bi-grams, or topic identification. Hence, it was in our best interest to remove them from the entire collection. Some examples of stop words are: am, is, are, their, like, what, etc.
  - **Punctuation**: Punctuation symbols (! , . etc.) are not useful either, for the same reasons mentioned in the previous point. They were also removed from the collection as part of the pre-processing.
- **Parts of Speech Tagging**:
  - A part of speech tagger simply scans the corpus and assigns a part of speech to each word. Unfortunately, the output is not as simple as a Noun or a Conjunction; we next discuss what the part of speech tagger really outputs.
  - The Part of Speech tagger used outputs something called treebank tags, which are extremely detailed grammatical tags. An example is shown below (MD is Modal, PRP is Personal Pronoun, etc.):

    ```
    [('Can', 'MD'), ('you', 'PRP'), ('please', 'VB'), ('buy', 'VB'),
    ('me', 'PRP') , ('an', 'DT'), ('Arizona', 'NNP'), ('Ice', 'NNP'),
    ('Tea', 'NNP')]
    ```
- **Lemmatization**:
  - Lemmatization refers to a process where a word is converted to its base form from multiple inflectional and derivational forms.
  - It differs from stemming as the latter relies on a much more crude rule based process to chop off the ends of words down to an approximation of the word stem.
  - Lemmatization does so with the use of vocabulary, synonyms, and a morphological analysis of the words. Hence, it is a far more sophisticated method than Stemming.

– We wanted to incorporate Parts of Speech tagging into lemmatization from the outset of the project, as it gives far more accurate results. Unfortunately, NLTK's *WordNetLemmatizer* did not take Treebank tags as inputs. It took basic noun, verb, conjunction, etc. tags only and to convert to an NLTK compatible format, we used a simple cleaning function, which is shown below.

```
def get_wordnet_pos(treebank_tag):

    if treebank_tag.startswith('J'):
        return wordnet.ADJ
    elif treebank_tag.startswith('V'):
        return wordnet.VERB
    elif treebank_tag.startswith('N'):
        return wordnet.NOUN
    elif treebank_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
```

Figure 3: Treebank to WordNet POS tags

- **Cleaning Function and Tokenization**: All the aforementioned pre-processing activities were carried out with a concise cleaning function, shown below. Once the collection was cleaned, every article was tokenized (split into an array of words) so that it could be fed into the next part of the code. The *sentencesnorm* array (shown below the cleaning function) represents the clean and tokenized corpus, as a list of lists.
  Note: The *sentences* array is a list of all documents in the corpus. The stopwords list is the *stop* array and *punct* is the list of punctuation symbols.

```
def clean(doc):
  stopless=" ".join([word for word in doc.split() if word not in stop])
  punctless="".join([i for i in stopless if i not in punct])
  norm=
  " ".join(lemma.lemmatize(word,get_wordnet_pos(nltk.pos_tag([word])[0][1]))
  for word in punctless.split())
  return norm

  sentencesnorm=[clean(doc).split() for doc in sentences]
```

Figure 4: Cleaning Function

5.4 Frequent Words

One of the suggested tasks in the syllabus was to identify a set of the most frequent words that occurred in the corpus, to get a preliminary idea if we were filtering noise properly or not. There were two attempts at this, each one being better than the previous, which are detailed below.

- **Iteration 1**: This involved cleaning the corpus (as detailed in the previous section) and lemmatizing the tokens without Part of Speech tagging. The results for the top ten most common words, along with their frequency of occurrence is shown below in Figure 5:

Word	Occurrences
gun	24869
school	21708
shooting	13384
people	12853
student	12234
state	8765
high	7776
time	6993
year	6856
law	6561

Figure 5: Most Common Words

- **Iteration 2**: This was the same as the previous iteration, except that lemmatization was done with part of speech tagging. The results are shown in Figure 6. One can clearly see a minor rearrangement of frequencies. Also, all instances of the word shooting have been lemmatized to shoot.

Word	Occurrences
gun	25009
school	21726
people	12853
student	12234
shoot	10838
state	9142
high	8418
time	8418
year	6856
law	6561

Figure 6: Most Common Words

5.5 Named Entities

Named Entities are words or phrases that form the important nouns (people, places, organizations, etc.) in a sentence. They can serve as important, discriminating features between collections of texts and can give us important insights about the texts(s). NLTK's NE chunker was used to generate these entities. The code that was used to accomplish it (with their frequencies), is shown below.

```
#Most freq Named Entities
NElist=[]
for i in range(len(sent1)):
    for sent in nltk.sent_tokenize(sent1[i]):
        for chunk in nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(sent))):
            if hasattr(chunk, 'label'):
                NElist.append((chunk.label(), ' '.join(c[0] for c in chunk))
```

Figure 7: Named Entity Code

The top ten results from the code are shown below, for GPE: Geo-Political Entity.

```
[(('ORGANIZATION', 'NRA'), 4510),
 (('GPE', 'Parkland'), 4265),
 (('GPE', 'Florida'), 2470),
 (('PERSON', 'Trump'), 2234),
 (('GPE', 'U.S.'), 2233),
 (('GPE', 'American'), 2028),
 (('GPE', 'Texas'), 1807),
 (('GPE', 'America'), 1621),
 (('GPE', 'United States'), 1579),
 (('GPE', 'Washington'), 1447)]
```

5.6 Topic Modelling

Topic Modeling[3] is a technique for classifying sentences, passages, or documents into different buckets or topics, which identify different themes in the collection. An excellent example for its utility would be an e-commerce based company using customer feedback to identify areas of improvement in the organization's services. Before we can actually run the topic modelling algorithm, there are some pre-processing steps that need to be accomplished. They are described below.

- **Data Cleanup and Tokenization**: The previous section explains how this was done.
- **Dictionary Creation and Document Vectorization**: The algorithm does not take the normalized and tokenized word corpus directly. It has to be converted into a vector for each document in the corpus. This can be achieved in multiple ways (*doc2vec* and *word2vec*) but after a good amount of research of available tutorials and literature, we decided to go with the *Bag of Words (BOW)* approach. The Bag of Words approach is quite intuitive and simple to understand. The best way is to use an example.

  For arguments sake, let us consider that our entire vocabulary (dictionary) consists of the following five words:
  `[‘‘baby’’,‘‘born’’,‘‘tomorrow’’,‘‘yesterday’’,‘‘today’’]` and we have indexed each unique
  word as {`‘‘baby’’:0,‘‘born’’:1,‘‘tomorrow’’:2,‘‘yesterday’’:3,‘‘today’’:4`}. These will serve as the keys for each unique word in the vocabulary henceforth.

  Now, consider the sentence: `[‘‘A baby was born yesterday’’]`. After the cleanup and tokenization phase, this sentence would reduce to `[‘‘baby’’,‘‘born’’,‘‘yesterday’’]`. On using the Bag of Words approach, this sentence would be represented as `[0,1,3]`. This

particular vector would now be fed into the topic modelling algorithm as a representation of a unique document. Of course, our vocabulary will be a far larger list in the actual study, but this idea illustrates the concept well.

- **Choosing Number of Topics**: This is probably the most ambiguous part of topic modelling, which is still an important area of research today. There is no foolproof way of replacing human logic to weave keywords into topics with semantic meaning, but that's not to say it hasn't been attempted. We used one of these attempts, called the coherence score. Coherence score is an attempt to mimic the human readability of the topic, where the higher the coherence score, the more "coherent" the topics are considered. The coherence scores for three to five topics were plotted, as shown below. Even though four topics have the highest coherence score, further analysis revealed that they did not make sense and that three topics were the ideal choice. This is a testament to the fact that nothing can substitute for human interpretation.

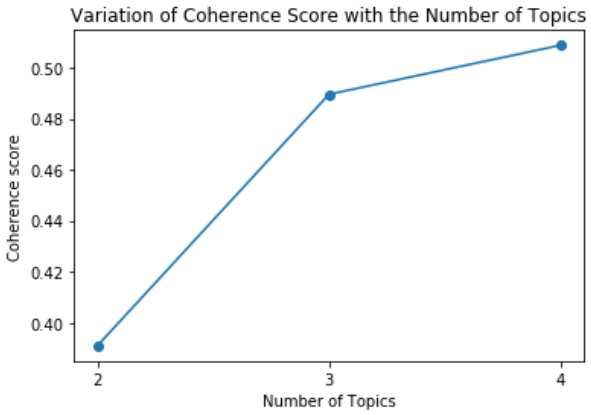

Figure 8: Coherence Scores

- **Latent Dirichlet Allocation (LDA)**: We used the Latent Dirichlet Allocation (LDA) algorithm[3] for topic modeling. The only other major algorithm that is used often is Non-negative Matrix Factorization (NMF). However, NMF is a deterministic algorithm, while LDA is a probabilistic one, which means that LDA is better for handling topic mixtures in different documents. In addition, LDA is believed to identify topics coherently whereas the topics from NMF are more disjoint. Hence, we used LDA for our analysis. The results from LDA (with three topics) with the top ten keywords for each topic are shown below.

  Note: The topic indices start with 0 and end at 2. The topic number is the first item when the parentheses begin.

  - **Topic 0 Keywords**:

    ```
    (0, ‘0.024*‘‘school’’ + 0.021*‘‘police’’ + 0.019*‘‘shoot’’ + 0.012*‘‘people’’ +
    0.012*‘‘kill’’ + 0.011*‘‘2018’’ + 0.011*‘‘high’’ + 0.010*‘‘santa’’
    + 0.010*‘‘fe’’ + 0.010*‘‘shot’’’)
    ```

  - **Topic 1 Keywords**:

    ```
    (1, ‘0.030*‘‘school’’ + 0.021*‘‘student’’ + 0.014*‘‘gun’’ + 0.011*‘‘people’’
    + 0.010*‘‘shoot’’ + 0.010*‘‘high’’ + 0.009*‘‘parkland’’ + 0.008*‘‘life’’
    + 0.008*‘‘like’’ + 0.008*‘‘violence’’’)
    ```

  - **Topic 2 Keywords**:

    ```
    (2, ‘0.025*‘‘gun’’ + 0.012*‘‘state’’ + 0.008*‘‘nra’’ + 0.008*‘‘law’’
    + 0.007*‘‘trump’’ + 0.006*‘‘people’’ + 0.006*‘‘year’’ + 0.005*‘‘use’’
    + 0.005*‘‘right’’ + 0.005*‘‘firearm’’’)
    ```

  These results are relatively simple to interpret. As an example, let us interpret the results for the first topic (Topic 0). The words are the differentiating words that define the topic, and have to be semantically woven together and interpreted. The coefficients attached to each word can be interpreted as a probability of occurrence or weight of the particular word. In the first topic (Topic 0), *school* has the highest chance of occurring, followed by *police*, *shoot* and so on. This reasoning can be extended to all three topics, and after a careful analysis of the keywords and associated documents (LDA labels each document with the dominant topic too), we came up with the following interpretation of the topics:

  - **Topic 0 Interpretation**: The general theme deals with the Santa Fe shooting.
  - **Topic 1 Interpretation**: The general theme deals with shooting incidents in general, with a slight emphasis on the Parkland shooting.
  - **Topic 2 Interpretation**: The general theme deals with the policy and legal aftermath of various shootings.

  Now that we have labelled documents, for each topic, we created three separate JSONs to be fed into the deep learning model. This is discussed in depth in the next section.

5.7 Abstractive Summarization Using Deep Learning

Here we describe the initial deep learning approaches tried and the final approach used to achieve our abstractive summary. Through our professor, Dr. Edward Fox, we were made aware of an approach called a Pointer-Generator Network (PGN) by Abigail See et al. [16]. This approach was

a hybrid between extractive and abstractive summarization. A PGN copies some words from the source text (pointing) along with using a generator to create novel words. The PGN also performed something called coverage where the algorithm keeps track of what has been "pointed" to and penalizes the algorithm if it points at the same material more than once. We also investigated the RLSeq2Seq approach by Yaser Keneshloo et al. [8]. However, the documentation was not as easy to follow by a novice and so we were not able to get a working version. Hence, we decided to stay with using a PGN approach as we were able to get some good working results.

We initially tried Pointer Summarizer, a PyTorch implementation of the PGN. We were able to get this to work, but the default parameters created repetitions in the summaries, which was not desired. We then tried the original github code for the PGN, a TensorFlow implementation. We had good success using just the default values. After this, we decided to continue to use the TensorFlow version of PGN.

Given our chosen approach, we performed some pre-processing on our collection to remove any empty and duplicate articles (described in Section 5.6). After that we performed topic modeling using Latent Dirichlet Allocation (LDA) to identify the major topics in our collection. For each topic identified, we filtered the collection to only contain topic terms in order to identify relevant articles. We then applied the PGN to create a summary. This was done for each topic identified, three in total for our collection, as described in Section 5.6.

For post-processing, a summary for each topic was created using the PGN. We then put the topics in an order that made sense to tell the story contained in our article collection. We first removed all repetitive information from each summary. Then we arranged some sentences to make more logical flow and in this process combined some sentences. Lastly, we added a little transitional material between each topic. This resulted in a clean abstractive summary.

CNN/Daily Mail Dataset

We used a pre-trained model made from the CNN/Daily Mail dataset [6]. This is a collection of 300k plus news articles from the news sources CNN and Daily Mail. This collection was ideal for us as we were summarizing news articles.

Single Document Summarization

The PGN used was originally tested on single documents and resulted in a short (a few sentences) summary. The default hyperparameters were set to accommodate a single document of a certain length. We learned to adjust the hyperparameters to result in a summary of varying length and detail. See Section 7 for more details.

Multi-Document Summarization

For our abstractive summarization we needed to summarize an article collection and not just one article. As discussed in Section 9.2, we decided to concatenate all the articles into one large article as suggested by Xuan Zhang, a recent Ph.D. graduate who worked with Dr. Fox. We tried this on our small collection (article count in Table 2) without using LDA to see what would result. Performing summarization on a merged small article collection resulted in the summary in Figure 9. This small collection was created by randomly selecting articles from the large collection. The summary produced appeared to have more relevant articles in it as the summary reflected at least one theme of the NeverAgain movement.

We then moved to summarizing the concatenated articles from the entire collection, size can be seen in Table 2. Figure 10 shows the resulting summary. The main themes of the NeverAgain

```
1 nra lapdogs in the white house and congress are doing everything they can to block policies that keep weapons of war off the streets .
2 if the senate confirms pro-gun extremist howard nielson to the federal bench , it would make the country 's mass shooting crisis even worse
  by giving the nra more power to undermine gun restrictions and expand access to firearms .
3 howard nielson has called state laws that prevent people younger than 21 from purchasing handguns and publicly carrying firearms
  unconstitutional .
4 he also tried to overturn a ban on assault weapons in illinois by arguing that high-powered rifles were not as dangerous as pistols and
  revolvers .3 in addition to advancing the nra 's hateful policies , undo decades of civil rights progress and empower the nra .
5 we can not allow that to happen , while holding a key position in george w. bush 's department of justice , nielson wrote a memo that
  justified the torture of civilians captured by u.s. forces in afghanistan .5 in 2016 .
6 he co-authored an amicus brief in the supreme court case of whole woman 's health v. hellerstedt , arguing in favor of imposing medically
  unnecessary standards on health facilities .
7 he says the judge hearing the case should recuse himself for being gay .4 in 2005 , while holding a key position in george w. bush 's
  department of justice , the life of known security , the life of known security , the life of boko haram , the life of boko haram , the
  life of chibos girls , the life of chibos in nigeria and the life of chibos girls , the life of corruption , the life of known security ,
  the life of boko haram , the life of electricity , the life of many other issues .
8 nielson proposition 8 a policy that banned same-sex marriage and argued that the judge will legitimize their hateful policies , undo
  decades of civil rights progress and empower the nra .
9 we can not allow that to fill the federal courts with right-wing judges who will legitimize their doors .7 trump and his extremist
  republican party are trying to fill the far right of many other issues .
10 if the senate confirms us.thirty-four mass shootings have occurred since the parkland , florida massacre and federal lawmakers have yet to
   respond with any real gun reform .
```

Figure 9: Example abstractive summary from PGN on a small subset of the collection.

```
1 jax is located at clayton county ac , adoption center .
2 he needs funds to help with vetting , boarding and possible transport to an out of state rescue .
3 all donations are tax deductible and any funds received for a dog who is adopted , reclaimed or euthanized goes toward helping other dogs .
4 if the campaign remains available within a few days , it 's likely that we determined it not to be in violation of our policies .
5 if the campaign remains available within a few days , it 's likely that we determined it not to be in violation of our policies .
6 record one right now.resistance at tule lake about this episode the dominant narrative of the world war ii incarceration of japanese-
  americans has been that they behaved as a model minority - cooperating without protest and proving their patriotism by enlisting in the
  army .
7 resistance at tule lake tells the long-suppressed story of 12,000 japanese americans who dared to resist the united states governments
  program of mass incarceration during the war .
8 branded as disloyals and re-imprisoned at tule lake segregation center , they continued to take a minute or two to record one now .
9 record a short video message of support .
10 or upload one from your device .
11 you can preview or redo your video before you post it .
12 but that we may contact you for more information , but that we wo n't notify you personally of our decision .
13 if the campaign remains available within a few days , it 's likely that we determined it not to be in violation of our policies .
14 thank you .
15 we do our best to find placement .
16 record one right now.resistance at tule lake about this episode the dominant narrative of the world war ii incarceration of japanese-
   americans has been that they behaved as a model minority - cooperating without protest and proving their patriotism by enlisting in the
   army .
17 resistance at tule lake tells the long-suppressed story of 12,000 japanese americans who dared to resist the united states governments
   program of mass incarceration during the war .
18 record a video upload a video nothing grabs attention for your cause like a personal video .
19 upload a short video message of support .
20 upload a short video message of support .
21 upload a short video message of support .
```

Figure 10: Example abstractive summary from PGN on the collection.

Collection Type	**Article Count**
Small Collection	481
Large Collection	12111

Table 2: The number of articles per collection.

ROUGE-1	0.11538
ROUGE-2	0.04000
ROUGE-L	0.11538
ROUGE-SU4	0.02143

Table 3: The ROUGE scores for our generated abstractive summary as compared to the Golden Standard.

collection did not show up in this summarization, which we found surprising. Performing topic modeling aided by revealing the actual relevant articles that contained the main themes. There is also repeated material and "junk", e.g., "upload a short video message of support". As described in Section 7, we tweaked the hyperparameters of the PGN to minimize repeated material and junk.

Run-Time

The run-time of our PGN over our pruned collection was surprisingly fast. As discussed in Section 2, our large collection was quickly pruned to around 3600. This number was further divided by topic as seen in Table 1. Given any permutation of the hyperparameters described in Section 8, the max run-time was between 2 and 3 minutes on our local Ubuntu server. This was possible as our Ubuntu server (described in more detail in Section 8) has a high-powered graphics card that could perform general purpose computing on graphics processing units (GPGPU). The version of the PGN we used was written in TensorFlow which takes advantage of graphics cards with this compute capability. Hence, our run time was quite fast which allowed fast turn-around time when tuning hyperparameters.

6 Evaluation

We performed different levels of ROUGE scoring for our generated abstractive summary. This was compared to the Golden Standard summary prepared by Team 6. The scores can be seen in Table 3. The scores were not that great, compared to the state-of-the-art, such as those reviewed by [4], plus our entity coverage was about 7.25%. Entity coverage is the number of entities, e.g., people's names, places, dates, identified in the generated summary as compared to a Golden Standard summary created by humans. However, this was a great learning experience and the scores are not too bad when considering the simplistic approach we took. Once relevant articles were identified, we simply concatenated them by topic and ran them through the PGN. We did not do any sophisticated multi-level summarizing such as summarize X documents into Y summaries, then summarize those summaries. Our approach shows the results of simple concatenation. This gives us a good baseline for comparing other techniques in the future.

An observation about the entity coverage is: we wonder if different pre-processing techniques or hyperparameter tuning could increase the coverage. Now that we have a baseline, we could tell which techniques work better compared to others. Further evaluations and tuning of this kind would be very interesting future work.

7 User Manual

In this section we will describe the pieces of our system and how to use them. As described earlier, the deep learning approach we decided to use is the Pointer-Generator Network (PGN) [16]. For

topic modeling, we chose LDA. We will first go through how to use LDA for identifying topics, then how to use the PGN for summarization of those topics.

The PGN used from github came with a pre-trained model using TensorFlow 1.2.1. We tried to train our own model, but using TensorFlow 1.5. The training did not work with 1.5 as the code was written for 1.2.1. So we decided to use the pre-trained model. We installed TensorFlow-GPU 1.2.1 to be able to run the code faster. This worked quite well.

7.1 Pre-Processing and Topic Modeling

This section will explain each part of the code and how to use it to clean the corpus and generate useful, relevant, and labeled articles (by topic) for summary generation using the PGN. The process starts with loading the data frame (after noise removal and preliminary cleaning). Then the following steps are followed (in the given order).

1. Load the custom stop word list, which was pickle dumped into a .txt file, and load the punctuation list from the string package.

2. The `Text` column in the data frame has to be converted into a list of all articles. The `sentences` (all lower case) and `sent1` arrays show how this is done.

3. The `clean` function removes stop words and punctuation, and lemmatizes an article with Part of Speech tagging. The cleaning function is applied to each article in the `sentences` list. Each article is then tokenized, to return a list of lists.

4. Another cycle of stop word removal is applied on the list of lists, and it is then flattened into a single list of tokens from all articles.

5. The `zipped` list is then created, which returns the unique words and their frequencies sorted in descending order.

6. A list of frequent named entities was then created using NLTK (ne_chunk), and sorted by frequency in descending order.

7. Using the tokenized list of lists (`sentencesnorm`), we create our vocabulary (dictionary using the Bag of Words approach) and document term matrix. Then, we use the `coherencevals` function to compute coherence scores for a range of topics.

8. We finally build the LDA model (with three topics). We finally labeled each article with its dominant topic using the `format_topics_sentences` function, and saved it in a data frame.

7.2 Working with the PGN

The PGN works well for short articles with no adjustments to the default hyperparameters. However, when we concatenated our small collection of articles into one large article, we needed to adjust the hyperparameters to produce longer summaries. The hyperparameters that are of interest are:

- **max_enc_steps**, default is 400 - The max time steps of the encoder (max source text tokens).

- **max_dec_steps**, default is 100 - The max time steps of the decoder (max summary tokens). If made too large, the summary starts to repeat itself. We believe this is with respect to the source article. The longer the article, the less repetition, as the value of max_dec_steps increases. It appears to be a ratio.

- **min_dec_steps**, default is 35 - The minimum sequence length of the generated summary.
- **beam_size** , default is 4 - The beam size for beam search decoding. The larger the beam size, the more detailed information is summarized, but processing takes longer.

The hyperparameters are defined in the *config.py* file.

Our initial tests were on single articles; that is how we found out the details described above for max_dec_steps. We then turned to ideas for multi-document summarization. We took our small collection (several hundred articles), concatenated them into one large article, then tried with default hyperparameters. The generated summary was too short. Then we tried:

- **max_enc_steps** = 400
- **max_dec_steps** = 2000
- **min_dec_steps** = 3500 (with value 1000, we had better results)

This resulted in a very long summary with the last half being very repetitive. This leads us to believe that with the right combination of parameters and ratio, we can get a decent summary. We were not sure how to test this without trial-and-error.

We then performed exploratory analysis on our large collection to discover how the hyperparameters affected the summary. Through trial-and-error, we identified a set of values that was adequate for our needs. Our trial-and-error consisted of systematically adjusting different hyperparameters to understand how they affected the different characteristics of the generated summary, such as length and the amount of details included in the summary. The values identified are as follows:

- **max_enc_steps**=400, 1000, 2000 - depending on the topic → Santa Fe, movements stemming from shootings, and policy, respectively
  - This appears to be related to the size of the .bin, i.e., the number of articles in the collection. The smaller the collection the lower this parameter needs to be.
- **max_dec_steps** = 2000
- **min_dec_steps** = 850
- **beam_size** = 8

This appears to provide a decent amount of detail for a summary, but there is still some repetition within the summary. Either way, merging the articles gave some hope. It appears that max_dec_steps should be larger than min_dec_steps, which makes sense, as one is a min and the other a max. It appears these hyperparameters dictate how many tokens are considered when decoding, hence, controlling the length of the generated summary.

The main run command calls the PGN to run on decode mode with test data:

- time python run_summarization.py –mode=decode –data_path=./finished_files/test.bin –vocab_path=./finished_files/vocab –log_root=./pretrained_model –max_enc_steps=400 –max_dec_steps=400 –min_dec_steps=100 –coverage=1 –single_pass=1 –beam_size=4

This calls the Python script "run_summarization.py" in decode mode, specifies where the data is (test data in our case), the vocab file to use, where to write results to (log root), the maximum encoding steps, the maximum decoding steps, the minimum decoding steps, we are performing coverage, performing a single pass, and beam size for beam search. In coverage mode, the algorithm

is penalized for covering the same words more than once. This aids in minimizing repetition and making sure more of the article is considered. In single pass mode, the PGN decodes all the articles one by one and writes them to *log_root* → *decode_test_XXX* → *decoded*:

- The *XXX* is part of the file name that has the parameter settings in the file name.
- E.g.: decode_test_400maxenc_4beam_100mindec_200maxdec_ckpt-238410
  - This means it is decode mode, with test data, max_enc_steps is 400, beam_size is 4, min_dec_steps is 100, and max_dec_steps is 200.

We used the Linux "time" command with the Python script to keep track of how long our script runs. This was useful when determining how long the processing of our large collection took on our local server, which, in turn caused us to realize we could perform all our processing on our server.

A special note must be made about which vocab file to use. We are using the pre-trained model for the CNN/Daily Mail dataset. One is also able to get the processed files for this dataset from [11], Option 1. Simply download the finished_files zip file and use the vocab file provided.

7.3 Visualizing the Result

The PGN code comes with the ability to visualize the results. After running the PGN, a file called *attn_vis_data.json* will be generated at the directory *pointer-generator/pretrained_model_tf1.2.1/decode/*. In order to prepare the JSON file for visualization, one must clone this [13] GitHub repository. Then overwrite the *attn_vis_data.json* with the file generated by the PGN.

In the directory with the visualization program, if you are using Python 2, run

```
python -m SimpleHTTPServer
```

If you use Python 3 instead, run

```
python -m http.server
```

After running the command, open *index.html* in the repository or go to *http://localhost:8000*. The default port is 8000; if you encounter an error, you could try another port, e.g., port 8080, by running the command

```
python -m SimpleHTTPServer 8080
```

for Python 2, or

```
python -m http.server 8080
```

for Python 3 . You will see a web page with the original article and the summary of the article. The summary will have text highlighted in green where the intensity of green representing the probabilities. You can hover the mouse over a word to see the probabilities. Figure 12 illustrates some visualization results from our collection.

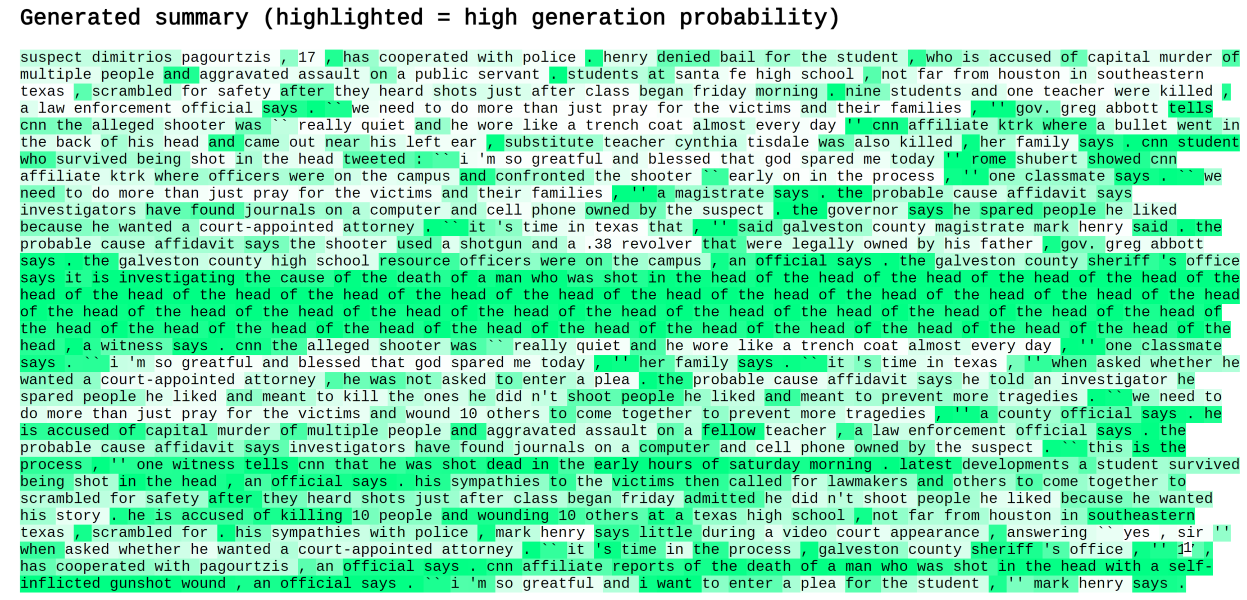

Figure 11: Visualization example.

7.4 Use Cases

Here we describe use cases for our system. Our system is meant to take a news article and provide a reasonable abstractive summary. One would use this in a number of cases:

- **Simple summary**: If there is not enough time to read an article, our system could provide a few sentences as summary.
- **Multi-document summary**: If one has a large collection of articles to summarize, they can be concatenated together and then our system can summarize them into one or two paragraphs.
- **Multi-topic collection**: If one suspects more than one topic in a collection, we are able to run topic modeling, filter the collection based on those topics, and run PGN on each filtered topic to provide a summary for each topic.

The summary produced will not be perfect, but be a guide for what the collection contains. For more articulated summaries, it is recommended to perform post-processing by a human to smooth things out and remove any redundancy.

8 Developer's Manual

Here we describe all that is needed to understand the architecture of our system, how it was developed, and how to maintain and/or extend it. We begin with an overview of the architecture. Then we describe what data pre- and post-processing was performed. The software and libraries needed are described next. We close out this section with future work and an inventory of the relevant files for the project.

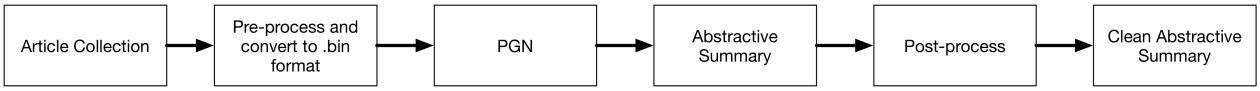

Figure 12: Architecture of overall system.

8.1 Project Architecture

There were a number of technologies used to run and maintain our system. We will first provide an overview of the architecture of the system. Then we will detail some software and hardware used in the development of the system. Lastly, we will detail the collection used as a basis for training.

Architecture

The architecture of our system is illustrated in Figure 12. We first start with an article collection. This collection is pre-processed to remove irrelevant articles (e.g., empty, or duplicates) and perform topic modeling to identify relevant topics in the collection. The pre-processed articles are then converted into a binary format to be fed into a Pointer-Generator Network (PGN). The output is a set of abstractive summaries (one per topic). These are post-processed to result in a clean abstractive summary. Post-processing entails removing repetitive information, deleting irrelevant information, re-arranging some sentences or phrases, and smoothing out the grammar and/or transitions.

We should note that the original implementation of PGN used ROUGE to test the results. This was feasible as the CNN/Daily Mail dataset used was labeled and a ROUGE score could be calculated. For us, our data is not labeled, so we disabled running ROUGE scoring. Besides, the library for ROUGE gave us an error:

- IOError: [Errno 2] No such file or directory: u'/home/chmille3/.pyrouge/settings.ini'

Since running ROUGE was not used when generating the summaries, we just disabled it so we would not generate an error every time. Later on in the project we run ROUGE to compare against the Golden Standard created by another team. Those results can be found in Section 6.

Version Control

For version control, we used git for our code repositories. This allowed us to keep a central location as development occurred across several computers. Once we had a deep learning environment established, our local Ubuntu server (described next) was the central location for code development as it had a complete and working environment.

Hardware

For this project, we built an Ubuntu Linux server to perform our deep learning. Our server is a Mac Pro version 5.1, hex-core Xeon processor at 3.33 GHz, 32GB of RAM, and a Nvidia GeForce 980 ti (6GB of RAM) graphics card. Our graphics card supports GPU acceleration with almost 3000 CUDA cores. This made it possible for use to run all of our experiments locally without relying on a cluster. After filtering out noise and irrelevant articles, applying our deep learning algorithm was possible in a timely manner. In other words, we could run our experiments locally on this server with minimal waiting time (1-2 minutes). One thing that made running locally possible is that we started with a pre-trained model. Training the model is what can take a very long time and requires many resources. We also used Cronopete [12] to back-up our server as it performs

incremental back-ups of specified folders. It was also nice to work with as it was built to be the Linux version of the MacOS Time Machine.

We used the TeamViewer software [18] to remotely log-in to our server from anywhere so as to be able to run experiments when needed. This was especially useful when one of the group members, particularly the one who ran the deep learning experiments, was on travel; he could run experiments while away from the physical machine.

CNN/Daily Mail Dataset

As described earlier, the CNN/Daily Mail collection is the dataset that our pre-trained model was trained on. When this dataset was converted into the binary format for use with the PGN [19], a vocab file was created. This file contained the frequency of words in the dataset. Since we used the pre-trained model of this dataset, the best results we have had when decoding and generating a summary is with this vocab file.

This dataset also had labeled highlights (abstract). When decoding is done, this abstract is paired with the generated summary, called a reference summary. Since our data does not have any labeled abstracts, the reference summary generated is empty.

8.2 Data Processing

The processing of our collection entails pre- and post-processing. The pre-processing necessary was to convert our article collection into the desired binary (.bin) format for the PGN. All Python commands and scripts use Python 2.

Pre-Processing

The original instructions on how to convert articles into this format was in a github repository [14]. In this repository, there is a link to the already created CNN/Daily Mail dataset [6] in binary (.bin) format. This dataset is used for any text summarization approaches we came across. Plus, there are instructions on how to specifically process the CNN/Daily Mail data into .bin format. The provided code was specific to this dataset, making it challenging to use with our own data. Hence, we forked the repository and streamlined the process.

The articles for our use came in a JSON format with the important tags being for the URL of the article and the Sentences of the article. The original process had a hash for each article based on the URL and a respective file with <hash>.story naming. This was the data that one downloaded from the CNN/Daily Mail dataset. The original process is as follows:

- python make_datafiles.py /path/to/cnn/stories /path/to/dailymail/stories
- This will take the CNN and Daily Mail .story files and tokenize them.
- There is a separate directory (urls_lists) with three URL lists: all_test.txt, all_train.txt, and all_val.txt. These were text files each with a list of URLs.
- The script would make sure that the number of files in the tokenized directories matched up with the number in the directory that was just tokenized. This was a check to make sure the numbers of files before and after tokenization were the same.
- The output would be a directory called "finished_files" that contained a test.bin, train.bin, val.bin, and a vocab file. Plus there is a "chunked" directory where each of the respective text, train, and val .bins are split into 1000 story chunks.

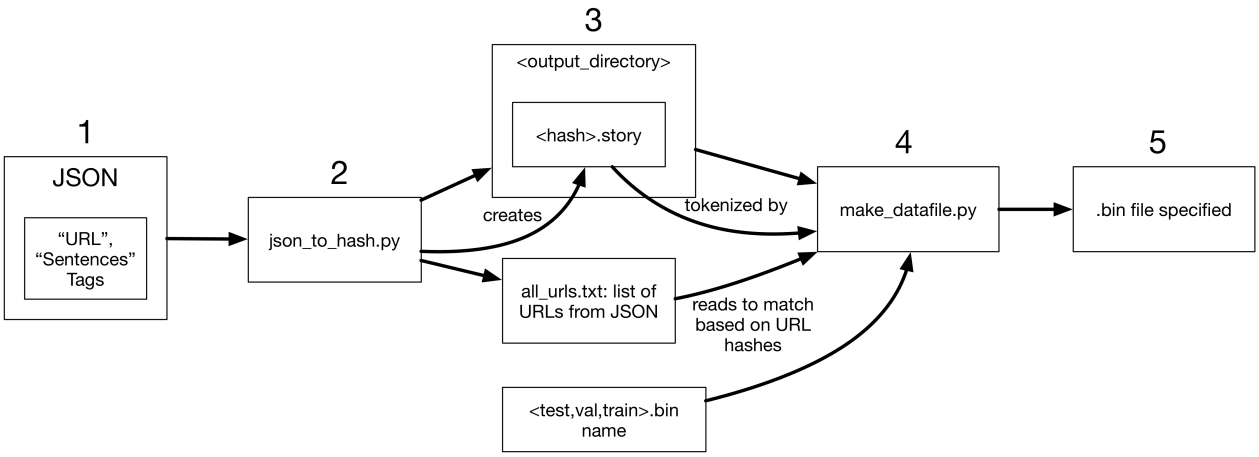

Figure 13: Process flow for article conversion to binary format.

This was a very tedious task with several challenges. If using MacOS, there is a hidden .DStore file in each directory. When the above process checks the before and after numbers for tokenization, it counts the .DStore file and throws an error about mismatched numbers. There is a mismatch since the .DStore file is not tokenized as it is not a .story file. This problem was not encountered in Ubuntu (the environment we eventually used for test and development). The way to get around this problem was to manually go into each of the directories to be tokenized and delete the .DStore file. However, this file is re-created every-so-often. This would cause errors again if it is created. After the .bin and vocab files were created, then they needed to be manually moved to the directory for the PGN software.

This script was too specific to the CNN/Daily Mail dataset, so we created a modified version that can take in the articles of a desired source and create a specific .bin file, whether train, val, or test. In order to make the .story files and their respective hashes as needed by the above process, we wrote a Python script (json_to_hash.py) to take in a JSON file and output the .story files and a text file with all the URLs. Its use is illustrated in Figure 13 and as follows:

- **python json_to_hash.py -f <.json file> -o <output dir>**

1. Takes in the JSON file (1), parses out the URLs ("URL" tags) and article body ("Sentences" tags) (2)
    - **NOTE:** The tags for the "URL" and "Sentences" can be manually set in the processJSON() function. This is useful for when the tag names differ between JSON files.
2. For each URL, make a unique hash.
3. For the respective article with that URL, make a file: <hash>.story (3)
4. Write these files to the output directory.
5. Write all the URLs to the file: all_urls.txt

Then, within the same directory call the modified make_datafiles.py, Figure 13 step 4:

- **python make_datafiles.py <data_stories_dir> <type: train.bin, test.bin, val.bin>**

1. Take in a directory with the articles in <hash>.story format and create a .bin.
2. Tokenized stories are written to directory: tokenized_stories.
    - **NOTE:** If this directory exists, it must be empty.
3. The all_urls.txt file is used by make_datafiles.py to verify that the number of .story files tokenized is the same as the number of URLs accounted for (4). This replaces the all_test.txt, all_train.txt, and all_val.txt files described above.
4. The specified .bin file is created (5).
5. A vocab file is also created containing an ordered list of words in the .bin file and their occurrence frequency.

Example using the commands together:

1. python json_to_hash.py -f las_vegas.json -o ./las_vegas_stories
2. python make_datafiles.py ./las_vegas_stories/ test.bin

This results in the respective .bin files needed for the PGN. In terms of using the CNN/Daily Mail pre-trained model, we use its vocab file for best results. Make sure to install the Stanford CoreNLP [17] as described in [14] Option 2, step 2.

From the explanation above, the important data and program files are:

- Data files:
    - JSON containing articles
    - <hash>.story - is a single article parsed from the JSON file
    - all_urls.txt - a list of URLs of the articles parsed from the JSON file
    - <test,val, or train>.bin - the resulting binary file that the PGN uses as input
- Program Files:
    - json_to_hash.py - input is a JSON with tags for URL and article body and output is a list of <hash>.story files, one per article, and the all_urls.txt file.
    - make_datafile.py - takes in a set of <hash>.story files, tokenizes them and creates the binary file format for the PGN.

From the above process a "vocab" file is created along with the .bin file. This vocab file contains a list of the counts of each word starting with the most frequent (including punctuation). The format is:

- <word1>, <frequency number >
- <word2>, <frequency number >
- ...

The pre-processed CNN/Daily Mail data into test.bin, val.bin, train.bin, and vocab was completed by Jaffer Wilson [19]. We discovered that, for the best results, this vocab file, created when the .bins were created for the CNN/Daily Mail dataset, worked best. When running the system, the top 50,000 words from the vocab file were used. This is a parameter we did not experiment with, so that can be explored in future work.

We did experiment with using the vocab file generated when we created our .bin file for processing:

- Original test.bin (from pre-processed CNN/Daily Mail) with original vocab (from pre-processed CNN/Daily Mail) → This worked.
- Original test.bin and our generated vocab file from our data → Did not work. The system would hang-up in an infinite loop with the message: "INFO:tensorflow:Failed to load checkpoint from ./pretrained_model/train. Sleeping for 10 secs...". This was also the same message we would get when using a different TensorFlow version than that of the pre-trained model.
- Our test.bin and original vocab file → This did work but yielded an empty reference summary. This is due to our data not being labeled as is the CNN/Daily Mail dataset. But for our purposes this is not important. What is important is the decoded summary.
- Our test.bin and new vocab file → This did not work and resulted in the same hang-up message from the above second bullet item.

Post-Processing

Here we will describe the post-processing of the generated abstractive summary. This includes:

- Removing redundancy.
- Fixing flow issues.
- Rearranging sentences.
- Given the separate topics, arranging them in a logical order and creating transitions between them.

Removing *redundancy* was pretty straightforward. Any sentences that were repeated were removed. Most of the sentences made sense but there were some grammatical and flow issues that needed to be addressed. The order of some sentences did not make sense, so some rearranging of sentences was performed. Since we had multiple topics in our collection, we needed to arrange the topics in a logical order and provide some transitional material between them.

8.3 Installation

There are various pieces of software needed to develop, maintain, and run our system. We will go over the software and libraries necessary to install for the PGN and GPU acceleration.

We decided on a Linux environment, Ubuntu 16.04 LTS, as we could get all of our software and libraries working on this platform. We originally tried on Mac OS 10.12.6 with no luck. So the rest of our described work is on Ubuntu 16.04 LTS. First, we need to update any software needing an update through the software updater built into the OS, then update apt-get:

- sudo apt-get update

And install pip, the Python package manager:

- sudo apt-get install python-pip

There are two paths to take. We first tried with tensorflow-gpu version 1.5, which ended up not working for the specific version of the PGN. What was needed was version 1.2.1. Hence, we will provide the installation instructions for both for completeness.

For GPU acceleration, both the CUDA toolkit and cuDNN library are needed. TensorFlow 1.5 was initially chosen as it was the first version the authors could get to work for GPU. For tensorflow-gpu 1.5, the following versions are needed:

- Cuda toolkit version 9. Install instructions:
  1. Download the specific version of the toolkit: https://developer.nvidia.com/cuda-90-download-archive
  2. Chose: Linux → x86_64 → Ubuntu → 16.04 → deb (local). This will be a large file (over a gigabyte).
  3. sudo dpkg -i cuda-repo-ubuntu1604-9-0-local_9.0.176-1_amd64.deb
  4. sudo apt-key add /var/cuda-repo-<version>/7fa2af80.pub
     - <version> can be found in the /var/ directory.
  5. OR
  6. sudo apt-key adv –fetch-keys http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64/7fa2af80.pub
     - A message will show in the terminal to run this command.
  7. sudo apt-get update
  8. sudo apt-get install cuda=9.0.176-1. This will be a large file (over a gig).
  9. sudo dpkg -i cuda-repo-ubuntu1604-8-0-local-ga2_8.0.61-1_amd64.deb
  10. sudo apt-get update
- cuDNN version 7.3.0 for CUDA 9.0
  1. https://developer.nvidia.com/rdp/cudnn-download, One will need a log-in to get it.
  2. sudo dpkg -i libcudnn7_7.3.0.29-1+cuda9.0_amd64.deb

For TensorFlow-gpu 1.2.1:

- Cuda toolkit version 8 for tensorflow-gpu 1.2.1
  1. https://developer.nvidia.com/cuda-80-ga2-download-archive
  2. Chose: Linux → x86_64 → Ubuntu → 16.04 → deb (local). This will be a large file (over a gigabyte).
  3. sudo dpkg -i cuda-repo-ubuntu1604-8-0-local-ga2_8.0.61-1_amd64.deb
  4. sudo apt-get update
  5. sudo apt-get install cuda=8.0.61-1
- cuDNN version 5.0 for CUDA 8
  1. Need a log-in to get to it.

2. Unzip cudnn-8.0-linux-x64-v5.0-ga.tgz (right-click: Extract here).
3. Results in a directory *cuda* with *lib64* and *include* directories.
4. lib64 contents go in /usr/local/cuda/lib64.
5. *include* file goes in /usr/local/cuda/include.
6. export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib64
   - Add this line at the end of the .bash_profile file in your home directory.

After the installations are complete, then call the following command to install the GPU version of TensorFlow desired:

- pip install tensorflow-gpu==1.5
- OR
- pip install tensorflow-gpu==1.2.1

Once the above is complete, then one can test to see if it is working by:

1. Open a terminal.
2. Type 'python' to enter a Python interpreter environment.
3. Type the following into the interpreter:
   (a) import tensorflow as tf <enter>.
   (b) sess = tf.Session(config=tf.ConfigProto(log_device_placement=True)) <enter>
4. Among the output you should see your GPU device name and compute capabilities plus total memory and how much memory is free.

As a side-note, the Pointer Summarizer uses PyTorch. The version we tested with was 0.4.1 and the install command:

- sudo pip install torch torchvision

However, we get an error related to network issues:

- TypeError: unsupported operand type(s) for -=: 'Retry' and 'int')

We believe when we try this with an ISP, say at a residence, it causes problems. We were able to get around this problem through using the network infrastructure on Virginia Tech's campus. This is our reasoning but it has not been verified. All we know is that we ran into this error when trying to install from a home installation (residence with an ISP) with a different computer than our Linux server. Installation on our server, which is located on Virginia Tech's network, resulted in a successful installation.

On a side-note, we tried to get a GPU to work on Mac OS by trying nvidia-docker through docker, but could not get it to install. We also learned that one cannot use a virtual machine like VirtualBox or Parallels as there is no connection straight to the hardware; it is all virtualized.

GitHub Repositories

Our PGN github repository [10] has 4 branches:

- Master - deployable version.
- Dev - used for development and test before merge into master.
- tf_1_2_1 - environment using TensorFlow 1.2.1 to use the pre-trained model for PGN.
- tf_1_5 - environment using Tensorflow 1.5 to see if a model can be trained with this version of Tensorflow for use with GPU. This has been abandoned (at least for the moment) as we got the GPU working for version 1.2.1.

Our pre-processing github repository [11] has 2 branches:

- Master - deployable version.
- Dev - used for development and test before merge into master.

Compatibility Issues

The entire PGN process has also been verified on Windows except for one caveat: There is no Python 2 pip package for TensorFlow version 1.2.1 on Windows. So, "make_datafiles.py" has to run on Linux. But, all the other tools we used for pre-processing, summarization, and visualization can work on Windows. If you already have the latest NVIDA driver installed, using the default option to install CUDA TookKits 8.0 on Winodws will overwrite the driver version to a much older version. You can manually deselect the option while installing the driver. One GPU we tested, NVIDIA GTX 1080, is running the driver version 416.16 and it is working fine with CUDA 8.0 so far. If you use Python 3, there is a GitHub repository available for it, but the repository is only for summarization; pre-processing has to be done with a Python 2 environment. Link to the GitHub repository for Python 3 [5]. We suggest Ubuntu 16 or 18 as the go-to platform for running the PGN.

8.4 Future Work

There are several areas in which our system could be tweaked and expanded. As mentioned earlier, the first 50K words from the CNN/Daily Mail vocab file are used for the PGN. However, there are more than 50K words in the vocab file. Experimenting with the number of words to include would be one area of future work.

Another area is how to feed the articles into the PGN. Currently all articles are concatenated together to make one large article and this large article is given to the PGN for summariztion. This is a simple approach but investigating other approaches may produce even better results.

9 Lessons Learned

Here we provide a history of the project, challenges we faced, and the methods we used and attempted to reach our final product.

9.1 Timeline

- Week 1(08/27——09/02)
  - Attend class lecture on ArchiveSpark
  - Set up the communication and organization systems include slack, Google Team Drive, and github.
- Week 2 (09/03——09/09)
  - Attend class lecture on Solr Index
  - Finalize weekly meeting time
  - Set up Solr Index (login, set password)
  - Decide on the road map for the project
    1. Pre-Process
       (a) Frequent Words
       (b) Lowercase
       (c) Stop words
       (d) Bigrams
       (e) Lemmatization
       (f) Punctuation
    2. Topic Modeling
    3. Semantic Analysis
    4. Deep Learning for Text Summarization
  - Decide on the technologies for deep learning summarization——PyTorch
- Week 3 (09/10——09/16)
  - Attend Product Defect Discovery Summarization from Online User Reviews presentation by Xuan Zhang
  - Prepare the first class presentation
  - Finish part of pre-process
    * Raw JSON to Pandas Dataframe
    * Dropped Null and Duplicate values
    * Tokenized all articles into a list of lists
    * Removed stopwords and punctuation
    * Basic Lemmatization (without parts of speech (POS) tagging)
    * Generated most frequent useful words
  - Get deep learning code
    * https://github.com/atulkum/pointer-summarizer - pointer generator networks
- Week 4 (09/17——09/23)
  - Prepare second class presentation
  - Set up Solr Indexing for small and large collection

  - Made Pre-Processing for new collection:
    * WARC/CDX conversion to JSON
    * Dropped Null and Duplicate values
    * Part of Speech Tagging (Treebank Tags)
    * Treebank Tags → WordNet Tags (which NLTK recognizes)
    * Lemmatization with Part of Speech included
    * Most frequent useful words generation
  - Study Deep Learning architecture: https://github.com/yaserkl/RLSeq2Seq

- Week 5 (09/24——09/30)
  - Attend class about deep learning technologies. Get a brief introduction on PyTorch technology involving Neural Networks in PyTorch, PyTorch on GPUs using the CIFAR dataset, and word embeddings using PyTorch: Implementing Continuous Bag of Words and Skipgrams using PyTorch.
  - Pre-processing update
    * Moved everything to Zeppelin, running on PySpark
  - Learn PySpark
  - Deep Learning update
    * Have working: https://github.com/atulkum/pointer-summarizer
    * Forked on github: https://github.com/chmille3/pointer-summarizer
    * Training model on GPU (local)
    * Small tests show promise
    * Updating script that creates the needed file format
    * Collaborating with Team 16

- Week 6 (10/01——10/07)
  - Prepare third class presentation
  - Pre-Processing update (of new collection - NeverAgain)
    * Most frequent and important Named Entities
    * Most frequent Bigrams
  - Brainstorm ideas on multi-document summarization approaches
    * Merge all documents into one —— Adjust hyperparameters/parameters to generate longer summaries
    * Split articles along boundaries
    * Make summaries for all articles, then summarize the summaries
    * Summarize in batches of X (e.g., 1000), and merge, then summarize

- Week 7 (10/08——10/14)
  - Attend class on generating summarization
  - Have brief discussion on information extraction
  - Gold Standard update

        * Extended outline and now making the first draft
    - Deep learning update
        * Clean small collection - decent results
        * Clean large collection - distracting articles caused bad results
            · Need to clean some more with a classifier or something
        * Work on plan for the rest of the semester
            · Finishing gold standard
            · Cleaning classifier for articles
            · Extractive summary: important sentences
            · Fine-tuning PGN and approach for a clean abstractive summary
            · Final Report
            · All done by Thanksgiving break

- Week 8 (10/15——10/21)
    - Prepare fourth class presentation
    - Work on Noise classifier and LDA
    - Update gold standard summary
    - Deep learning update
        * Simplify pipeline for processing data into the right format for the PGN algorithm
        * Multi-document summarization
        * Working on the cluster

- Week 9 (10/22——10/28)
    - Prepare fifth class presentation
    - Gold standards draft due
    - Review final report section in syllabus
    - Review other project reports in VTechWorks (linked from syllabus)
    - Writing on our own parts

- Week 10 (10/29——11/04)
    - Work on visualizing results (shown in original pointer-generator paper)
    - Start Overleaf template

- Week 11 (11/05——11/11)
    - Prepare sixth class presentation
    - Work on final report
        * Editing tex file in Overleaf
        * Decide who is working on what section
        * Timeline draft
    - Make abstractive summary draft
    - Deep learning update

        * Generated abstractive summary draft achieved after experimenting with hyperparameters for topics and LDA topic modeling
        * Slightly different parameters for each topic - to tease out wanted information for summary
        * Working on post-processing of generated abstractive summary - draft complete

- Week 12 (11/12——11/18)
    - Keep working on final report
        * Combine all the material we have
        * Improve timeline
        * Finish user manual part, conclude visualization
        * Gold standard summary procedure
        * Brainstorm on introduction and abstract section

- Week 13 (11/19——11/25)
    - Thanksgiving break

- Week 14 (11/26——12/02)
    - Prepare and make final presentation
    - Final report first draft due

- Week 15 (12/03——)
    - Update final report

9.2 Challenges faced

General Challenges

We tried running the PGN on pre-processed data, e.g., lower cased, punctuation removed, stop words removed, and lemmatization. The results were not good. It produced a string of words that kind of made sense but did not make readable sentences. E.g.:

- columbine shoot clinton administration sue manufacturer violence commit product death threat boycott layoff near bankruptcy fire smith wessons ceo change agree settlement make let fight right now investor gun company decide go problem solution may chance stand thank help stand nra washington post feb 27 2018 .

At the beginning of researching deep learning strategies, we tried to have things run on MacOS Sierra. We were unable to get GPU acceleration working for Mac and then moved to Ubuntu 16.04. On Ubuntu we were able to get the GPU versions of PyTorch and TensorFlow to work and speed up our processing.

GPU acceleration on our local Ubuntu server was sufficient for our collection and we did not need to use the ARC clusters or the Hadoop cluster.

One definitely needs to be careful with what version of software is being used. For example, we tried to use TensorFlow version 1.5 to run code that was written for version 1.2.1. The code did not work until we had version 1.2.1 installed. The code did run with the mismatched version but a error catch part of the code was triggered each time resulting in a non-functioning program.

We tried segmenting the articles so that all the introductions are in one article and tried to summarize that. The results were mixed as what we got was repeated introduction material.

During development and testing of our deep learning architecture, we used a Ubuntu version 16.04 virtual machine (VM) for simplicity. We could work on the VM, then pull down the changes on the local Ubuntu server. However, our version control mechanism git could not push changes to the git repository at github.com, meaning our local Ubuntu server could not pull any code changes. Once we realized this, we migrated our work to our local Ubuntu server.

GPU Acceleration

Ideally we wanted to be able to run a GPU version of the code so processing would be much faster. However, figuring out the required prerequisites needed for specific library installations was non-trivial. See Section 8.3 for the details of installation. We were not sure how important this was, so we only invested some time to find a solution. Trying to do this ended up being very challenging.

For the specific software needed (TensorFlow) to run the PGN, we ran into complications when we initially tried installing the necessary libraries on Mac OS 10.12. The Mac OS support for the GPU version of Tensorflow was dropped for the version we needed. We then looked into trying on Ubuntu 16.04 LTS, which still was supported. We were able to obtain the correct versions of libraries needed for Ubuntu, but the challenge was figuring out which versions were needed for our specific version of TensorFlow (version 1.2.1). After we identified which software versions are needed for TensorFlow 1.2.1, we were able to perform trial-and-error installations until we discovered the correct commands. The sequence for installation of the libraries is described in Section 8.3.

Once it worked, we were very glad, as it sped up the processing of our collection tremendously. Using the GPU allowed us to apply the PGN to our collection and get results back in less than two minutes. Hence, we did not need to go through the trouble of relying on a cluster for processing.

Multi-Document Summarization

One challenge faced was how to perform multi-document summarization, i.e., summarizing a collection of documents. Our chosen deep learning mechanism (PGN) was designed to summarize a single article. This we were able to accomplish simply by following the PGN author's instructions. However, we needed to develop a means to summarize multiple documents.

To accomplish this, we brainstormed three ideas. The *first* was to break articles up into their general sections:

- Brief summary
- More Details
- Then lesser details, and so forth

This is the general structure of an article. There are subsequent sections of an article that have varying degrees of information depth. So what if we chopped up the articles into sections and summarized those sections into collections? For each section we would have a summary; put those summaries together and we have an overall summary. As described earlier in this section, we experimented with just the brief summary section of the articles. This resulted in repetitive introduction (brief summary) material.

This did not have as good results as expected. We *first* tried the introductions which resulted in a summary full of introduction material, meaning the summary read as the start of an article multiple times in a row. Some parts of this summary were good ways to start a summary but they

```
1 a person named stephen paddock reserved a room at chicago 's blackstone hotel during the city 's lollapalooza music festival .
2 but that person never checked into the hotel , which represents the hotel .
3 danley was not placed under arrest and it was not known when , or if , danley would agree to be interviewed about stephen paddock , who
  killed himself in the las vegas hotel room from which he launched his deadly shooting spree .
4 the shooting started as a crowd of more than 22,000 people watched country music star jason aldean perform on sunday night at the end of
  the three-day route 91 harvest festival .
5 massachusetts is the first state since the deadly shooting in las vegas last month to ban bump stocks , the gun accessory the shooter used
  to increase his rate of fire .
6 a bump stock device could be seen from the hotel across las vegas boulevard from the festival as the sound of automatic gunfire took over
  from the music.please look at the time stamp on the story to see .
7 massachusetts is the first state since the deadly shooting in las vegas last month .
8 the plane is shown after arriving at lax on october 3 , 2017 .
9 read the full story on latimes.multiple victims were being transported to hospitals after a shooting late sunday at a music festival on the
  las vegas strip .
10 the gunman killed himself inside the room on the 32nd floor of the mandalay bay resort and casino , from where he had carried out his
  killing spree .
11 the crowd screamed and fled in panic as victims dropped to the ground while others hid and crawled under parked cars .
12 please look at the time stamp on the story to see when it was last updated .
13 before the massacre , stephen paddock rented a room at a las vegas condo complex that overlooked another music festival .
14 read the full story on latimes.multiple victims were being transported to hospitals after a shooting late sunday at a music festival on the
  las vegas strip .
15 muzzle flashes could be seen from the hotel across las vegas boulevard from the festival as the sound of automatic gunfire took over from
  the music.please look at the time stamp on the story .
16 we do n't know yet .
```

Figure 14: Example of summary based solely on the merging of the first section of all articles.

were within the body of the summary. Overall, this summary did not flow together. An example of this is illustrated in Figure 14.

Second, our group identified important words for topics from LDA and then identify articles that contain a certain number of those words and concatenate those articles based on each topic to produce a summary. This worked fairly well. The *last step* is to take the top X number of most frequent words and bigrams and rank articles from those with the most to the least, and concatenate them in that order. Then prune articles with Y or less words/bigrams, if needed. Then the PGN will start with the most relevant articles. We did not get around to testing this as the second approach worked well.

9.3 Approaches Tried

In this section we record other deep learning approaches tried and our experiences with them.

Pointer Summarizer

We initially began investigating using the Pointer Summarizer, the PyTorch implementation of a Pointer-Generator. This is where we learned how to convert our own data into the desired format for the Pointer Summarizer/Generator (described in Section 8.2). When we got the Pointer Summarizer and Pointer-Generator working, the Pointer-Generator appeared to have better results, so we decided to use the Pointer-Generator. This is strange as they supposedly are each an implementation of the same algorithm. For completeness, we include our developer notes for the Pointer Summarizer.

We forked the original Pointer Summarizer code with our fork being at:

- https://github.com/chmille3/pointer_summarizer.git

To download it:

- git clone https://github.com/chmille3/pointer_summarizer.git

We initially tried to get everything set up on MacOS, but soon found it very difficult to get all the necessary libraries installed properly, and dealing with the hidden file .DStore detailed in Section 8.2.

In order to run the Pointer Summarizer, one must update the necessary paths in the data_util/config.py. These paths are:

- train_data_path
- eval_data_path
- decode_data_path
- vocab_path
- log_root

The Pointer-Summarizer did not come with a pre-trained model, so we needed to train our own. Once the paths above are set to the appropriate values (current values in the code explain what they should be set to), one can train by running the bash script:

- start_train.sh

The original bash script redirected the output to a log file, but for some reason it did not work, so that part of the script had to be commented out:

- python training_ptr_gen/train.py # >& ./log/training_log.txt
  - Where the # denotes where a comment starts, i.e., start of where we commented out part of the command.

To decode and generate the summaries, one needs to run the bash script:

- start_decode.sh /path/to/model
  - Note: the model was created during training in the log_root directory.
  - We also needed to comment out the redirection for a log file as in the start_train.sh

In the end, we decided against using the Pointer Summarizer as it did not give as good results with the default parameters as the PGN did. This may have been due to the model that we trained. We trained for 400k iterations where the authors did 500k but the authors did note that 500k was more than needed.

RLSeq2Seq

Another approach we explored was a sequence-to-sequence approach call RLSeq2Seq [7]. This was another deep learning approach that also used the CNN/Daily Mail dataset. The documentation (README) was extensive with a lot of information, but we could not figure out the correct commands for training and decoding (generating summaries). We even posted some questions for the authors which helped some. But in the end we decided to not pursue this approach.

10 Conclusion

In this project we were able to successfully provide a working system that provides abstractive summarization of a large article collection. The collection given to us had three main topics which we summarized using a Pointer-Generator network (PGN). After that we post-processed the summaries and merged them into one single summary.

Performing automatic abstractive summarization is very challenging and requires advanced techniques to achieve good results. We learned that true abstractive summarization is not available yet, but techniques are improving. This project shows the potential for automatic summarization and we hope the efforts of our team will be fruitful for other researchers.

11 Acknowledgements

We would like to thank Dr. Edward Fox (fox@vt.edu) for his insight and guidance on this project. We would also like to thank Frank Wanye (wanyef@vt.edu), from team 12, for providing us with an Overleaf template for our final report. This project was supported under NSF IIS-1619028.

Bibliography

[1] Advanced research computing (arc.vt.edu), December 2018.

[2] Dakota access pipeline, https://en.wikipedia.org/wiki/dakota_access_pipeline, 2018 December.

[3] Wikipedia, https://en.wikipedia.org, 2018 December.

[4] Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., Kochut, K., and Trippe, E.-B. D. Text Summarization Techniques: A Brief Survey.

[5] Back, S. https://github.com/becxer/pointer-generator, December 2018.

[6] Cho, K. https://cs.nyu.edu/%7ekcho/dmqa/, December 2018.

[7] Keneshloo, Y. https://github.com/yaserkl/rlseq2seq, December 2018.

[8] Keneshloo, Y., Shi, T., Ramakrishnan, N., and Reddy, C. K. Deep reinforcement learning for sequence to sequence models. *arXiv preprint arXiv:1805.09461* (2018).

[9] Kumar, A. https://github.com/atulkum/pointer_summarizer, December 2018.

[10] Miller, C. https://github.com/chmille3/pointer-generator, December 2018.

[11] Miller, C. https://github.com/chmille3/process_data_for_pointer_summrizer, December 2018.

[12] Rastersoft. https://www.rastersoft.com/programas/cronopete.html, December 2018.

[13] See, A. https://github.com/abisee/attn_vis, December 2018.

[14] See, A. https://github.com/abisee/cnn-dailymail, December 2018.

[15] See, A. https://github.com/abisee/pointer-generator, December 2018.

[16] See, A., Liu, P. J., and Manning, C. D. Get To The Point: Summarization with Pointer-Generator Networks. *ArXiv e-prints* (Apr. 2017).

[17] Stanford. https://stanfordnlp.github.io/corenlp/, December 2018.

[18] TeamViewer. https://www.teamviewer.com/en-us/, December 2018.

[19] Wilson, J. https://github.com/jafferwilson/process-data-of-cnn-dailymail, December 2018.

Appendix A

Golden Standard Summary for NoDAPL

1 Introduction

The Dakota Access Pipeline, also known as DAPL, is a $3.8 Billion construction program that began in July 2014. It is a 1,172-mile-long (1,886 km) underground oil pipeline project that starts in the Bakken shale oil fields in northwest North Dakota, passes through South Dakota and Iowa, and ends in Illinois at the oil tank farm near Patoka. The ambition for this project is to transport 450,000 barrels of crude oil from Dakota to Illinois.[1] The DAPL project is the result of an extensive process that involved review and approval by the U.S. Army Corps of Engineers and regulators in North Dakota, South Dakota, Iowa, and Illinois.

NoDAPL is a Twitter hashtag from 2014 referring to the Dakota Access Pipeline protests. Some of the discussion is archived at [1]. Indigenous activities led the movement, which continued into 2018 since activists were still doing time for their actions to stop the pipeline.

2 Detail on Affected Area

The affected area included North Dakota in the early stages of planning. The pipeline passed through disputed Sioux land, which was promised to the tribe in the 1851 Fort Laramie treaty but was later taken away. The project also was designed to cross the Missouri river, which is the main source of drinking water for the Sioux people.[3] Large-scale protests at Lake Oahe were held against possible water pollution and destruction of sacred tribal sites. According to a report from The Pipeline and Hazardous Materials Safety Administration (PHMSA), since 2010 there are more than 3300 incidents reported about oil and gas pipeline leaks and ruptures.[2] In fact, even a tiny spill could damage and destroy the local environment and water supply.

3 Timeline

The project was announced and approved on July 2014 by Energy Transfer Partners. There was an information session held in October 2014 with South Dakota and Illinois landowners.

The project was submitted to the IUB (Iowa Utilities Board[2]). Dakota Access announced that

[1] https://www.nodaplarchive.com/

[2] https://en.wikipedia.org/wiki/Iowa_Utilities_Board

they were approved by the North Dakota Public Service Commission on Jan. 25, 2016. Attorneys for the Standing Rock Sioux Tribe filed a lawsuit on July 27, 2016 and soon after the lawsuit, many demonstrations and protests begin.

- Aug. 31, 2016, the United Nations[3] Permanent Forum on Indigenous Issues offered support to the tribe.
- Sept. 3, 2016, the protests become more aggressive. The security personnel hired by Dakota Access started to use pepper spray.
- Sept. 9, 2016 a Federal judge denied the tribe's request for injunction.[6]
- Sept. 26 2016, with societal pressure, President Obama weighed in and kept in touch with USACE, trying to illustrate the possibility of rerouting the pipeline to avoid passing through North Dakota.
- Oct. 9, 2016 the Federal Appeals Court denied the appeal of the tribe.
- Oct. 10, 2016, actress Shailene Woodley was arrested during the protest.
- Oct. 26, 2016, authorities started to clear the protesters and removed a roadblock set by the protesters that was blocking a state highway.[7]
- Nov. 1, 2016 President Obama announced that he will be monitoring the situation closely.
- Dec. 4, 2016, USACE said that they will not grant easement for drilling the pipeline under Lake Oahe.
- Jan. 14, 2017, President Trump signed an executive order to advance the construction of the pipeline. The construction was completed by April 2017.[8]

4 Opponent Response

The opponents against the Dakota Access Pipeline project included local aboriginal Sioux, environment protectors, and veterans. At the beginning of the protest, the scale of opponents was quite small. With the development proceeding, many celebrities and public figures, like actress Shailene Woodley, traveled to North Dakota and supported the protest in public. Many more people started focusing and following this event from social media such as FaceBook and Twitter. To provide support for the opponents, on April 1, 2016, Standing Rock's Historic Preservation Officer founded and built a Sacred Stone Camp as a spiritual resistance to the Dakota Access pipeline. Protesters set up teepees and tent camps and threatened to block the highway to slow construction progress.[4] Meanwhile, protesters also sued the Army Corps of Engineers, claiming that they violated the National Historic Preservation Act (NHPA) and the National Environmental Policy Act (NEPA). They requested the government reconsider the cultural significance of federally-permitted sites and implications for the waterways.

[3]http://abcnews.go.com/topics/news/world/united-nations.htm

5 Federal Government/President Response

The federal government had asked the company to "voluntarily suspend" construction near the area until further research is done on the area. Kelcy Warren, chairman and CEO of Energy Transfer partners, responded to a request from the federal government, calling concerns about pipeline impacts on water supplies "unfounded". State historic preservation offices could not find sacred items on the route. Warren wrote that the company would meet officials in Washington "to understand their positions and reaffirm our commitment to getting Dakota access to the pipeline into operation". In December 2016, under President Barack Obama's administration, the Corps of Engineers refused to provide an easement to build a pipeline under the Missouri River.[5]

On January 24, 2017, US President Donald Trump signed an executive order that overturned Obama's legislation and advanced the pipeline under "terms and conditions that remain to be negotiated," speeding up environmental reviews of what Trump has called an "extremely burdensome, lengthy, and horrible approval process".[5] The number of protesters decreased after Trump approved the construction of the pipeline. National Guard and law enforcement officers evicted the remaining people on February 23, 2017. The pipeline was completed in April and the first oil was delivered on May 14, 2017.[5]

6 References

1. https://www.kcet.org/shows/earth-focus/a-living-breathing-movement-an-introduction-to-the-dakota-acce ss-pipeline-issue
2. http://time.com/4548566/dakota-access-pipeline-standing-rock-sioux/
3. http://time.com/money/4551726/dakota-access-pipeline-standing-rock-sioux-tribe-devastate-poorest-people/
4. https://en.wikipedia.org/wiki/Dakota_Access_Pipeline_protests
5. https://en.wikipedia.org/wiki/Dakota_Access_Pipeline
6. https://www.nbcnews.com/storyline/dakota-pipeline-protests/federal-judge-denies-tribe-s-request-halt-dakota-access-pipeline-n645616
7. https://fox59.com/2016/10/27/authorities-removing-protesters-at-dakota-access-pipeline/
8. https://www.usnews.com/news/national-news/articles/2017-01-24/trump-signs-executive-order-to-advance-keystone-xl-dakota-access-pipelines
9. https://daplpipelinefacts.com/
10. https://insideclimatenews.org/news/05092018/standing-rock-tribe-dapl-dakota-access-pipeline-oil-spill-ris k-report-army-corps
11. https://www.forbes.com/sites/brighammccown/2018/06/04/what-ever-happened-to-the-dakota-access-pip eline/7ad62fc24055
12. https://www.npr.org/tags/492631446/dakota-access-pipeline
13. http://dajia.qq.com/original/meiguo/ll20161229.html
14. https://abcnews.go.com/US/timeline-dakota-access-pipeline-protests/story?id=43131355

15. https://en.wikipedia.org/wiki/NODAPL

16. https://ccrjustice.org/home/blog/2018/02/21/nodapl-movement-was-powerful-factual-and-indigenous-led-lawsuit-lies-can-t

17. https://www.workers.org/2018/06/23/nodapl-water-protectors-continue-the-struggle/

18. https://www.ama.org/publications/MarketingNews/Pages/nodapl-movement-brings-native-voices-to-forefront-on-social-media.aspx

19. https://www.greenpeace.org/usa/nodapl-why-are-paid-mercenaries-watching-activists/

Appendix B

Golden Standard Summary for NeverAgain

This collection is about the NeverAgain student group. NeverAgain (also known by the Twitter hashtags NeverAgain and EnoughIsEnough; see also @NeverAgainMSD and neveragain.com) is a U.S. political action committee that promotes tighter regulation of guns to prevent gun violence. The group was formed in the aftermath of a mass shooting at the Marjory Stoneman Douglas High School (MSD) in Parkland, Florida by youth involved in the theatre program. This shooting occurred on February 14, 2018, when former student Nikolas Cruz gunned down 17 students, all of whom were killed, with an AR-15 rifle, and injured 17 others.

In response to the shooting, three MSD students–Cameron Kasky, Alex Wind, and Sofie Whitney–founded NeverAgain MSD on February 15, 2018. Other Parkland students that later joined were David Hogg, Emma Gonzalez, Delaney Tarr, Alfonso Calderon, Sarah Chadwick, Jaclyn Corin, and Ryan Deitsch. Over the next few days after the shooting, the group gained over 35,000 Facebook and Twitter followers and instantly became a national movement. Also within days of the shooting, MSD students boarded buses, demanding action from lawmakers on gun control, in Florida's capital, Tallahassee. A bestselling book "NeverAgain: A New Generation Draws the Line" appeared 19 June 2018 by David and Lauren Hogg.

Their mission is to make schools safe, and not allow one more child to be shot at school, nor one more teacher to die to save the lives of students. Their goals include for the US Congress to pass legislation addressing gun violence.

The group staged a nationwide protest against gun violence in Washington D.C. and 800 other locations on March 24, 2018. This protest, dubbed March for our Lives, was attended by millions of people from the U.S. and other countries. The group has also been present at several town hall events, with the intent to hold legislators accountable for their stance on gun control laws.

One of the primary organizations targeted by NeverAgain is the National Rifle Association (NRA), a powerful American non-profit organization that promotes gun rights and generally opposes new gun control laws. The NeverAgain movement was able to pressure several U.S. companies to sever their ties with the NRA, including the First National Bank of Omaha, Hertz, Avis, Enterprise, Budget, MetLife, Symantec, SimpliSafe, Delta Airlines, and American Airlines.

The political reaction to the NeverAgain movement was largely partisan, with Republican groups being against gun control measures and Democratic groups being for gun control. Nevertheless, the group has managed to attain a few political victories. Most notably, in March 2018, the Florida legislature passed the Marjory Stoneman Douglas High School Public Safety Act, which involved various gun control measures, including raising the age to buy a gun in the state from 18 to 21,

banning bump stocks, and denying the mentally ill the right to buy guns. Illinois worked on similar legislation. Moreover, several notable retailers that sell guns, including Walmart and Dicks Sporting Goods, have raised the age to buy guns at their stores from 18 to 21. President Donald Trump also voiced support for a nationwide ban of bump stocks.

The NeverAgain movement has also been the subject of various misinformation and conspiracy theory campaigns. Notable attacks against the group came from Ted Nugent, Alex Jones, and Rick Santorum. In one example, various fake images of Emma Gonzalez tearing up the U.S. constitution were circulated on the Internet.

The group has also received large amounts of public and financial support from various notable people and organizations. Some celebrities who have donated large sums of money to the group include George Clooney, Oprah Winfrey, Jeffrey Katzenberg, and Steven Spielberg. Former president Barack Obama also wrote a letter expressing admiration of the group's goal to end gun violence. Several major colleges, universities, organizations, and companies have also voiced support of the movement.

Appendix C

Abstractive Summary Results

Here we present the raw results from the PGN for each of our topics, then the post-processed results.

1 Policy

nra lapdogs in the white house and congress are doing everything they can to block policies that keep weapons of war off the streets . if the senate confirms pro-gun extremist howard nielson to the federal bench , it would make the country 's mass shooting crisis even worse by giving the nra more power to undermine gun restrictions and expand access to firearms . howard nielson has called state laws that prevent people younger than 21 from purchasing handguns and publicly carrying firearms unconstitutional . he also tried to overturn a ban on assault weapons in illinois by arguing that high-powered rifles were not as dangerous as pistols and revolvers .3 in which 17 people were killed and others injured . the nra is expected to speak at the annual meeting on friday , a white house official confirmed monday . the nra , bolstered by trump , has been a vocal proponent of allowing more guns in public places , including at least one parent who lost his child in the feb. 14 shooting at marjory stoneman douglas high school in which 17 people were killed and ten wounded . the president is scheduled to speak at the annual nra gathering in dallas on friday , and many attendees will be packing guns , knives and other weapons for the event which includes more than 20 acres of firearms exhibits expected to draw 80,000 members . according to the nra , students , who say schools should be shot at the place they go online and tune in to nratv for a live stream . if at any time you would like to unsubscribe , they could go online and tune in to nratv is n't just for the convention . as a result , firearms and firearm accessories , knives or weapons of any kind will be prohibited in the forum prior to and during the convention , the nra says . we will send you updates on this and other important campaigns by friday mornings shooting at the vp at the nra annual convention , the nra says . the nra says the u.s. secret service will coordinate security for the pence speech and will not allow weapons in the arena while he is present . as a result , the nra says it will be to blame if it 's the future of the federal justice system . the nra says it will be " [UNK] " to protect people who want to get involved in the wake of gun violence . but the nra says it is " deeply concerned " about the impact of violence in the wake of the nra 's decision . the nra says it is " deeply concerned " about the future of the supreme court ruling . the nra says it will be held at the end of this year 's presidential elections . president obama says he will be " disappointed " by the decision not to indict him . he says he will be " disappointed " by the decision to protect the nra 's policies . he says he will be " disappointed " by the nra 's decision on social media . he says he will be able to play the role in the future of the 2016 election . he says he will be able to play the role in the future of the 2016 midterm elections . he says he will be " disappointed " by

the nra 's decision to take action against the president . he says he will be " disappointed " by the decision to protect the nra 's actions . he says he will be " disappointed " by the decision not to indict him . he says he does n't want to know if he does n't want to get involved in the wake of the violence . he says he will be able to play the role in the future of the 2016 election , he says . he says it will be an example of how many americans have been involved in recent years . he says it is n't clear if it will be an example of how to get involved in the future , he says . he says it will be an example of how to deal with some of the issues , he says . he says it will be an example of how to respond to the crisis , he says . he says it will be an example of how to deal with some of the world 's challenges . he says it will be an example of how to deal with some of the world 's challenges . he says it will be an example of how to do more than a million people in the world . he says it is n't clear if it will be an example of how to get involved in the future , he says . he says it will be an example of how to get involved , he says .

2 Sante Fe Shootings

suspect dimitrios pagourtzis , 17 , has cooperated with police . henry denied bail for the student , who is accused of capital murder of multiple people and aggravated assault on a public servant . students at santa fe high school , not far from houston in southeastern texas , scrambled for safety after they heard shots just after class began friday morning . nine students and one teacher were killed , a law enforcement official says . " we need to do more than just pray for the victims and their families , " gov. greg abbott tells cnn the alleged shooter was " really quiet and he wore like a trench coat almost every day " cnn affiliate ktrk where a bullet went in the back of his head and came out near his left ear , substitute teacher cynthia tisdale was also killed , her family says . cnn student who survived being shot in the head tweeted : " i 'm so greatful and blessed that god spared me today " rome shubert showed cnn affiliate ktrk where officers were on the campus and confronted the shooter " early on in the process , " one classmate says . " we need to do more than just pray for the victims and their families , " a magistrate says . the probable cause affidavit says investigators have found journals on a computer and cell phone owned by the suspect . the governor says he spared people he liked because he wanted a court-appointed attorney . " it 's time in texas that , " said galveston county magistrate mark henry said . the probable cause affidavit says the shooter used a shotgun and a .38 revolver that were legally owned by his father , gov. greg abbott says . the galveston county high school resource officers were on the campus , an official says . the galveston county sheriff 's office says it is investigating the cause of the death of a man who was shot in the head of the head , a witness says . cnn the alleged shooter was " really quiet and he wore like a trench coat almost every day , " one classmate says . " i 'm so greatful and blessed that god spared me today , " her family says . " it 's time in texas , " when asked whether he wanted a court-appointed attorney , he was not asked to enter a plea . the probable cause affidavit says he told an investigator he spared people he liked and meant to kill the ones he did n't shoot people he liked and meant to prevent more tragedies . " we need to do more than just pray for the victims and wound 10 others to come together to prevent more tragedies , " a county official says . he is accused of capital murder of multiple people and aggravated assault on a fellow teacher , a law enforcement official says . the probable cause affidavit says investigators have found journals on a computer and cell phone owned by the suspect . " this

is the process , " one witness tells cnn that he was shot dead in the early hours of saturday morning . latest developments a student survived being shot in the head , an official says . his sympathies to the victims then called for lawmakers and others to come together to scrambled for safety after they heard shots just after class began friday admitted he did n't shoot people he liked because he wanted his story . he is accused of killing 10 people and wounding 10 others at a texas high school , not far from houston in southeastern texas , scrambled for . his sympathies with police , mark henry says little during a video court appearance , answering " yes , sir " when asked whether he wanted a court-appointed attorney . " it 's time in the process , galveston county sheriff 's office , " 17 , has cooperated with pagourtzis , an official says . cnn affiliate reports of the death of a man who was shot in the head with a self-inflicted gunshot wound , an official says . " i 'm so greatful and i want to enter a plea for the student , " mark henry says .

3 Movements Stemming from Shootings

politicians refuse to place the lives of the people who they work for over receiving money and campaign funding . since then , almost 700 more lives have been taken in preventable acts of mass shootings . since then , the us saw our politicians deciding that funding for their career from the nra was more important than implementing common sense gun reform . since then , participants will drop to the ground for 12 minutes - the duration of 700 seconds representing the number of lives lost due to mass shootings . on february 14 , 2018 , seventeen-year-old david hogg and his wife patricia traveled to los angeles to paint another mural of his son . that day , with seventeen classmates and faculty dead , they had joined the leadership of a movement to save their own lives , and the lives of all other young people while changing the conversation on gun control . now he is a miami-based artist , plans to travel around the country to paint 17 murals total in honor of each person that was killed . the fight continues on june 12th , when student activists will stage die-in to protests all around the nation , including downtown la . join us at la city hall on the south lawn to fight for what 's to do that is to empower future leaders . in a march 10 video , now seen by more than 2.3 million people , a miami exhibit of his son joaquin guac appears in the middle of a white mural as oliver paints the words : we need to clean how we are represented right now , and the only way to have a fair game and the need to change the ref , oliver said in an interview with the washington post . the art murals are part of a larger movement begun by change the ref , a nonprofit organization founded by the oliver family in hopes of empowering and educating young people while changing the cost of our politicians inaction . the washington post newsroom was not involved in the creation of this content , and an in-depth look at the making of the neveragain movement . by the next morning , florida , shooting comes a declaration for our times , and published by wp brandstudio . fba set up a gofundme page to raise money for their efforts to register young people on-site as the murals are painted , and provide customer service for these products . something we hope you can start reading kindle books on your smartphone , tablet , or computer - and show other signs of previous use . this is the day that the pulse nightclub massacre is paid for by an advertiser and published by amazon -lrb- fba -rrb- is a service we offer sellers that lets them store their products in amazon 's fulfillment centers , and we directly pack , and provide customer service . but the washington post newsroom shows wear from use , but remains in good condition and . learn more about wp brandstudio : a new generation draws the line and millions of other books are available for . kindle — audible enter your mobile number or email address below and we 'll send you a link to download the free kindle app . this demonstration will begin at 10:30 am , and at 12 pm , oliver , who is a miami-based artist , went to school at marjory stoneman douglas high school . he has created three murals so far from the end of his son in the parkland , like many survivors

of the parkland , he says . the referee may have had connections with the other team , he says . he says he was inspired by a basketball referees calls during a game they were painted above his head . he says it is a movement of the cost of our kids is against any group that have agreements with powerful groups or lobbyists . he says the fight we are having next to these kids in the us stepped up to the table to put money in any politicians pocket , he says . he says it is the first time since he was killed in the wake of his son 's death in the wake of his son 's death . he says he was inspired by a new york times bestseller from two survivors of the [UNK] sister , who went to los angeles to honor the parkland victims . he says he was inspired by his son for a decade before he was killed in the wake of his son 's death . he says he was inspired by his son , who died in an interview with cnn ireport . he says he was inspired by cnn 's " larry king , " he says , he says . he says it 's time to show how many people are demanding for new leaders , he says .

4 Post-Processed Abstractive Summary

Here is the final post-processed abstractive summary.

4.1 Policy

National Rifle Association (NRA) supporters in the White House and Congress are doing everything they can to block policies that keep weapons of war off the streets. If the Senate confirms pro-gun extremist Howard Nielson to the federal bench, it would make the country's mass shooting crisis even worse by giving the NRA more power to undermine gun restrictions and expand access to firearms. Howard Nielson has called state laws that prevent people younger than 21 from purchasing handguns and publicly carrying firearms unconstitutional. He also tried to overturn a ban on assault weapons in Illinois by arguing that high-powered rifles were not as dangerous as pistols and revolvers.

The NRA, bolstered by President Trump, has been a vocal proponent of allowing more guns in public places. The president is scheduled to speak at the annual NRA gathering in Dallas on Friday, and many attendees will be packing guns, knives and other weapons for the event, which includes more than 20 acres of firearms exhibits that is expected to draw 80,000 members.

4.2 Focal Shooting: Santa Fe High School

These concerns stem from shootings such as the one at Santa Fe High School. Students at Santa Fe high school, not far from Houston in southeastern Texas, scrambled for safety after they heard shots just after class began Friday morning. Nine students and one teacher were killed, a law enforcement official says. Suspect Dimitrios Pagourtzis, 17, has cooperated with police, but was denied bail, who is accused of capital murder of multiple people and aggravated assault on a public servant. He is accused of killing 10 people and wounding 10 others. Law enforcement official says this resulted in the accusations of capital murder of multiple people and aggravated assault on a fellow teacher. The probable cause affidavit says the shooter used a shotgun and a .38 revolver that were legally owned by his Father and says investigators have found journals on a computer and cell phone owned by the suspect. CNN reports the alleged shooter was "really quiet and he wore like a trench coat almost every day".

4.3 Movements stemming from shootings

As a result of such incidents, politicians have refused to place the lives of the people who they work for over receiving money and campaign funding. Since then, almost 700 more lives have been taken

in preventable acts of mass shootings. As a form of protest, participants will drop to the ground for 12 minutes - the duration of 700 seconds representing the number of lives lost due to mass shootings. Incidents such as these led to the making of the #NeverAgain movement. Also since then, the US saw our politicians deciding that funding for their career from the NRA was more important than implementing common sense gun reform.

On February 14, 2018, David Hogg and his wife Patricia traveled to Los Angeles to paint another mural of their son. A Miami-based artist, Oliver plans to travel around the country to paint 17 murals total in honor of each person that was killed. In a March 10 video, now seen by more than 2.3 million people, a Miami exhibit of his son Joaquin Guac appears in the middle of a white mural as Oliver paints the words: we need to clean how we are represented right now, and the only way to have a fair game and the need to change the REF, Oliver said in an interview with the Washington Post. The art murals are part of a larger movement begun by change the REF, a nonprofit organization founded by the Oliver family in hopes of empowering and educating young people while changing the cost of our politicians inaction. With seventeen classmates and faculty dead, they had joined the leadership of a movement to save their own lives, and the lives of all other young people while changing the conversation on gun control.