7-8-2022

Toward Analyzing the Diversity of Extractive Summaries

Aaron David Hudson
Portland State University

Toward Analyzing the Diversity of Extractive Summaries

by

Aaron David Hudson

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science

Thesis Committee:
Ameeta Agrawal, Chair
Atul Ingle
Antonie Jetter

Portland State University
2022

Abstract

As the amount of text generated across the internet continues to increase, developing methods for processing that text to glean valuable insights is paramount. Automatic text summarization is one such method that aims to provide a concise and representative summary of input text, allowing users access to the most salient points from a large amount of textual data. However, in working with these summaries, especially those generated from social media data, questions arise about not only the quality of a summary, but also its ability to reflect the diversity of user perspectives. This work examines the quality of summaries with regards to dialect-diversity, as measured for human-written summaries as well as for those generated automatically. Specifically, in this work, we perform an extensive analysis on a dialect-diverse Twitter dataset, DivSumm. Our analysis suggests that humans typically write fairly diverse summaries. In addition, we also note that automatic clustering algorithms generate fairly well-representative clusters. Given these insights we propose a novel clustering-based approach for generating extractive summaries from dialect-diverse social media data. Our approach generates superior summaries than baseline methods when evaluated via ROUGE metrics.

Dedication

I would like to dedicate this work to my grandmothers Frances Quinn Morris and Dorothy Hudson, both of whom instilled within me the courage and self-love to be myself.

Acknowledgements

I would like to express my deepest gratitude to Professor Ameeta Agrawal for her invaluable guidance throughout this work. Her expertise has been instrumental in shaping this thesis, and much of my educational journey itself. When I first found her class on Natural Language Processing, it opened a door to an area of computer science that both excites me and I am passionate about, and the compassion she demonstrates in her mentoring gave me the courage to pursue research in the area. I have learned a tremendous amount in my time working with her, and she has forever altered the course of my life for the better. I will always treasure our time working together on this research.

Additionally, I would like to thank my thesis committee members Professor Atul Ingle and Professor Antonie Jetter for graciously providing their time and expertise. They both offered invaluable insights about our work and helped to elevate this thesis to the level it is today.

Furthermore, I would like to thank my mother Amy Quinn, my partner Linel Rogers, and Samantha Alt for their unwavering support throughout this process. Always knowing they are in my corner has given me the foundation I needed to pursue this research.

Lastly, I would like to thank Bēbē and Milo for keeping me company during the many long nights that went into this thesis. The cost of a few cat treats and belly rubs was a small price to pay for their constant company.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivation

Natural Language Processing (NLP) has seen a recent surge of growth, bringing with it sophisticated systems for performing a variety of tasks. We now have models that can generate fake news stories,[1] detect the sentiment of movie reviews (Koumpouri et al., 2015), and even make predictions about the mental health of people based on their social media posts (Krishnamurthy et al., 2016). While this growth has brought a lot of positive change to the field, these systems are highly susceptible to favoring certain groups while excluding others in the representation they provide. For example, if we were using a model to compare the average sentiment of movie reviews as a means of comparing the quality of movies, the system could unintentionally penalize movies containing African American names. It has been shown that sentiment analysis models often favor European American names in more positive sentiment contexts (ie: joy), whereas these systems will favor African American names in more negative sentiment contexts (ie: anger, fear, sadness) (Kiritchenko and Mohammad, 2018). If reviews mention character names, the system may unfairly penalize works with characters with African American names despite these names providing no insight into the actual sentiment of the review or the quality of the movies themselves. Scenarios like this are not uncommon, as the ways many of our NLP models are created perpetuate the societal biases present in the text they are trained on. Two such examples are female applicants

[1] https://app.inferkit.com/demo

being penalized in resume filtering systems[2] and hate speech detection models being more likely to flag tweets authored by African Americans as offensive (Sap et al., 2019). Thus, as society continues to adopt more and more NLP systems, it is important that we do our due diligence to try to mitigate the potential for excluding groups of people.

One task that has received a lot of recent attention is the automatic summarization of text, which stands to be immensely useful in providing a solution to processing the insurmountable amount of textual data generated daily on the internet. Specifically, with the influx of social media, users have greater power than ever to share their thoughts and opinions with the entire world in seconds with the click of a "post" button, but the ability of humans to individually ingest textual data has remained the same. Thus, humans alone can no longer keep up and need an intermediate tool to help close the gap. Automatic summarization offers the opportunity to have a computer parse large amounts of text and generate a condensed summary highlighting the most salient points.

While the high-level concept of automatic summarization may seem straightforward, upon deeper consideration numerous questions arise that require exploration. First and foremost, what are the characteristics of a *good* summary? Should it focus on highlighting the most frequent topics? Is its goal purely centered on representing the text itself, or should it also consider the authors of that text? Should unique perspectives be included along with the most frequent perspectives? What are the potential negative effects on less populous groups if only the most frequent perspectives

[2] https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G

are included? Thinking about these questions (and more) led us to the question of how diverse are summaries.

When considering what an automatic summary may be used for, it is evident to us that diversity in summaries is valuable. For example, a specific use case for automatic summarization of social media data could be trying to analyze public opinion on a current event via Twitter data. A summary could be automatically generated from all tweets that refer to the current event. In this case, would there not be value in understanding the perspectives of all communities involved, instead of only the most populous (which would likely correlate to the most frequently shared opinions)? We feel that maintaining the perspectives of various groups is important because it helps prevent the perspectives of less populous groups from being excluded during the summarization process. This helps ensure all groups of authors have a better chance of being represented in the final generated summary. Overall, we think there is value in reflecting the diverse perspectives in the summaries that are automatically generated, both in improving the overall quality of information gleaned from the summary itself and also ensuring well-balanced representation of all groups.

1.2 Applications

Automatic text summarization is used in a variety of tasks, ranging from information retrieval, information extraction, and even question answering (El-Kassass et al., 2021). It provides a method for condensing information quickly and efficiently, yielding

summaries dense in information that often see use in text mining and analytics applications (El-Kassass et al., 2021).

Use of automatic text summarizers has been conducted for a variety of domains, including news, opinion and sentiment, microblog and social media sites, books, emails, biomedical documents, legal documents, and scientific papers (El-Kassass et al., 2021). As with many tasks in NLP, automatic text summarizers are typically designed with a specific domain in mind, as this allows them to better handle any intricacies and nuances unique to the text they will be summarizing.

In the news domain specifically, one example of an automatic text summarizer is Newsblaster (McKeown et al, 2002). It automatically collects and summarizes news articles daily (while also clustering and categorizing them) from various sites across the internet in an effort to connect users with news stories that will be of greatest interest to them (El-Kassass et al., 2021).

Within the academia domain, work has been done to use automatic text summarization to generate a survey of input research papers on a given topic (El-Kassass et al., 2021). The summarizer consists of two methods: the first works with the citations of the papers to determine the overall structure and any pertinent connections between them, while the second hones in on the content of the papers to generate the survey paper summary (El-Kassass et al., 2021).

Overall, automatic text summarization has been applied in a variety of tasks and has seen considerable research in tuning it to work across many domains.

1.3 Our Contributions

The major contribution of this work is an analysis of the diversity of summaries generated from a collection of tweets that are categorized by topic and dialect. More specifically, this work contributes the following:

a. an extensive analysis of diversity in human-generated summaries;
b. an extensive analysis of diversity in automatically-generated clusters;
c. a novel approach of generating extractive summaries for dialect-diverse social media data; and,
d. a detailed evaluation of the proposed approach followed by a discussion of the results.

Overall, the findings of this research can be leveraged for future work in improving the textual quality and dialect diversity of automatic extractive summarization techniques.

1.4 Thesis Outline

The current chapter of this document outlines the motivation for this work, along with the applications of automatic text summarization and the contributions of this thesis. The next chapter provides an overview of related work and lays the theoretical groundwork for the techniques that will be used throughout the work.

Chapter 3 examines the dataset used for this work, providing some foundational information and the pre-processing required. Furthermore, it presents the results of the

work across four sections (3.2 through 3.5). Since the motivation of this work itself was grounded in exploration, we segmented the results based on the questions that were driving our research.

Finally, chapter 4 presents a discussion of the conclusions drawn and outlines future avenues for this work. Following the last chapter is a listing of references and appendices with supplementary materials, including details about the implementation of the project, outlining the environment that the research was performed in and the required libraries.

2 Related Work

This section will lay foundational groundwork for the research presented in this thesis by examining related work. It first outlines automatic summarization in general and with specific reference to social media text, then examines the state of bias and fairness in NLP, and lastly looks at various clustering techniques.

2.1 Automatic Summarization

Automatic summarization is a task within the NLP space where the goal is to have a model receive input text, transform that text into a meaningful representation, and then generate a summary of that text that is both cohesive and representative of the original input text. It is a difficult task in that it combines elements of both Natural Language Understanding (NLU) and Natural Language Generation (NLG). Automatic text summarization can be seen in many present-day applications, with search engines generating previews for results and news websites generating headlines to attract readers representing two of them (Allahyari et al., 2017). Additionally, Google recently announced the addition of an automatic summarizer to their Google Sheets application, shown in Figure 1 (Saleh and Kannan, 2022).

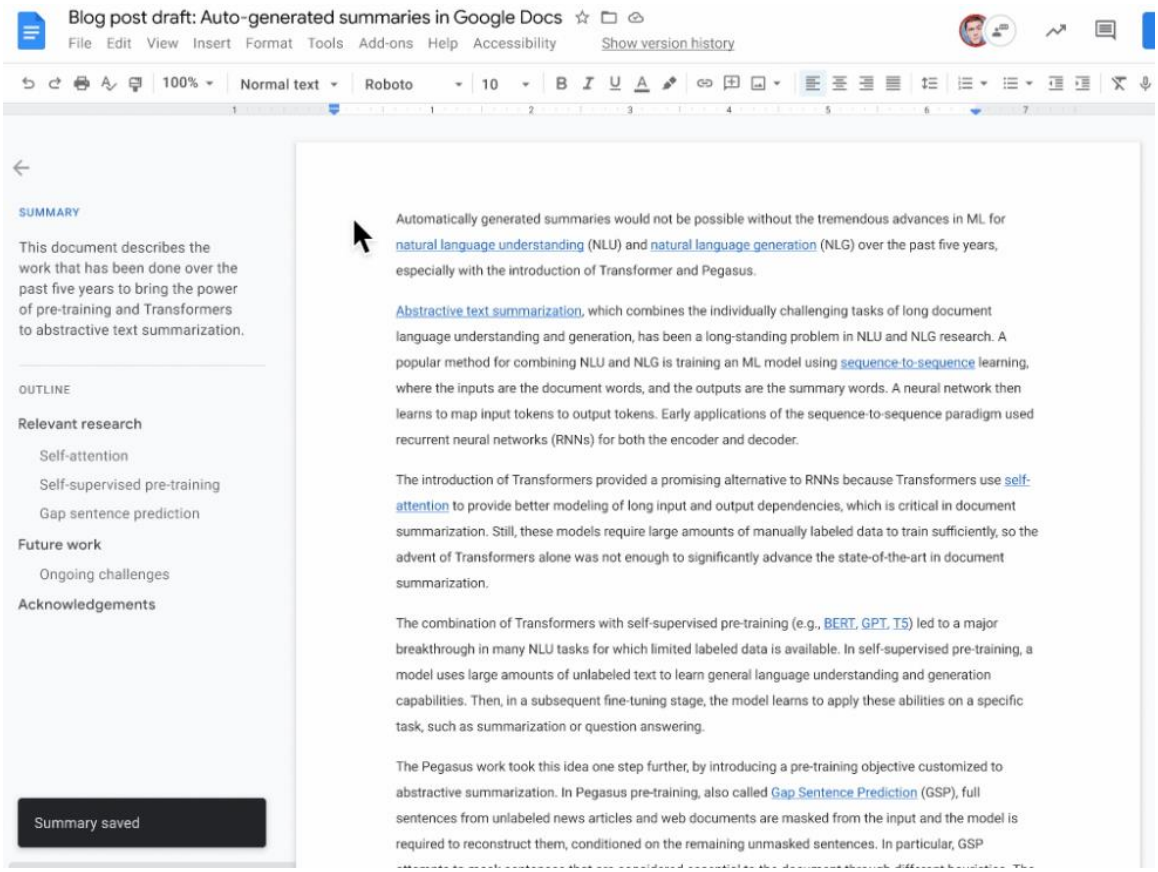

Figure 1: Image of the automatic text summarizer within the Google Sheets application (Saleh and Kannan, 2022).

This section will walk through various methods for automatic summarization, how the generated summaries can be evaluated, and specific work within this area with respect to tweets.

2.1.1 Methods

Automatic text summarization methods have been studied since the late 1950s (Luhn, 1958), and are especially of interest in the age of an increase of online textual data being generated (Allahyari et al., 2017). The importance of automatic summarization is evident as the amount of textual data being generated daily surges on the internet. Between all

forms of social media available, users are generating an amount of text that cannot be kept up with by normal human abilities. Thus, automatic summarization can enable us to comprehend a vast amount of textual data considerably faster than normal.

Automatic text summarization attempts to generate a condensed and fluent representation of a collection of text that maintains the overall meaning of that text (Allahyari et al., 2017). It is an exceedingly difficult task due to the fact that models used to generate these summaries lack the intuition and knowledge humans rely on when they create summaries, and as such other methods must be employed to work around these limitations.

The input text for automatic text summarization can be in two different forms: either as a single document, or in a multi-document format (Gambhir and Gupta, 2017). **Single-document summarization** is well-researched, where the goal is to summarize a single input document. Contrastingly, **multi-document summarization** is an emerging area of research, with the goal of representing multiple documents within a single summary. Multi-document inputs are most often either a collection of news articles, which are written from a single perspective, or social media data that contains diverse perspectives.

There are two main methods for automatically generating text summaries: extractive summarization and abstractive summarization (Allahyari et al., 2017). **Extractive summarization** is the simpler of the two, and is more widely researched. It consists of extracting exact excerpts of text from the input documents that are deemed important and representative of the overall text, and these excerpts are combined to form

the final summary. Oftentimes, a specified number of sentences are selected to generate the final summary. Figure 2 presents a simple pipeline for generating an extractive summary.

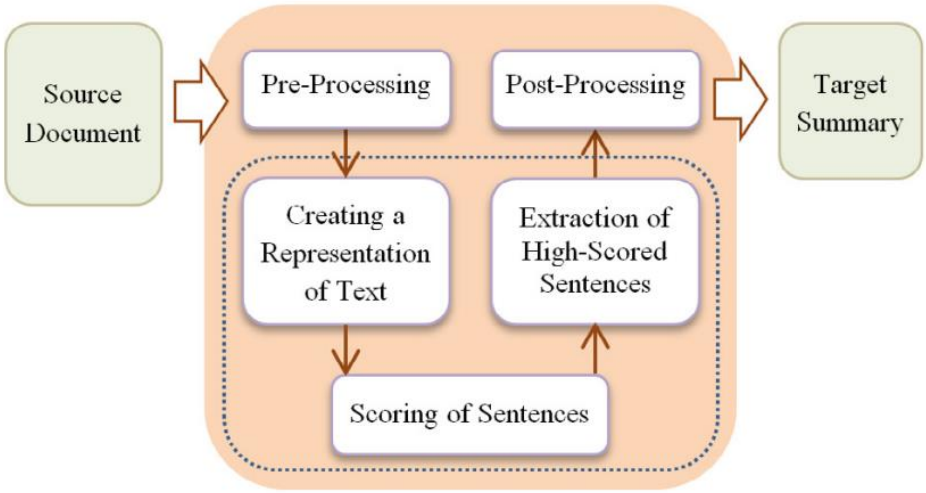

Figure 2: Example pipeline for generating an automatic extractive summary, where the summarization portion occurs inside the dotted rectangle (El-Kassass et al., 2021). The source documents, or input text, are pre-processed (ie: removing stopwords, stemming, lemmatization are examples) and features to represent the text numerically are generated. Then, the individual components of the text (ie: sentences, tweets) are scored, and the highest scoring are extracted to form the final summary. Finally, post-processing is performed (ie: reversing stemming as an example) to yield the final summary in its output form.

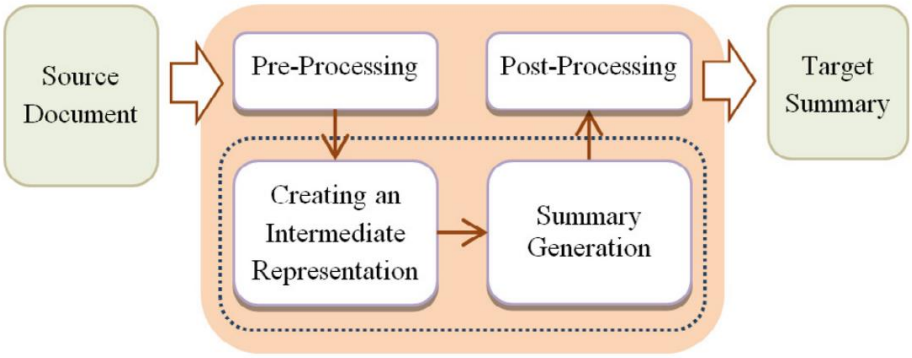

Figure 3: Example pipeline for generating an automatic abstractive summary, where the summarization portion occurs inside the dotted rectangle (El-Kassass et al., 2021). The source documents, or input text, are pre-processed to better situate the text for generating an intermediate representation. Then, from this representation the model generates a summary of entirely new text that is then post-processed to yield the final summary. Examples of pre-processing and post-processing can be seen in Figure 2.

Contrastingly, **abstractive summarization** is more complex in that the output summary is composed of new text generated by the model to represent the input text. This style of summarization requires the model to contain a much more thorough understanding of context and possess the ability to generate text, making it a considerably more difficult task. As such, it has been studied less than extractive summarization as a whole, but research into this area is increasing as the tools for automatic text generation improve. Figure 3 presents the general pipeline for generating an abstractive summary.

Lastly, Table 1 provides an example of an input text and both an abstractive and extractive summary generated from it.

Input Text	Dolly Rebecca Parton (born January 19, 1946) is an American singer-songwriter, actress, and businesswoman, known primarily for her work in country music. After achieving success as a songwriter for others, Parton made her album debut in 1967 with Hello, I'm Dolly, which led to success during the remainder of the 1960s (both as a solo artist and with a series of duet albums with Porter Wagoner), before her sales and chart peak came during the 1970s and continued into the 1980s. Parton's albums in the 1990s did not sell as well, but she achieved commercial success again in the new millennium and has released albums on various independent labels since 2000, including her own label, Dolly Records. She has sold more than 100 million records worldwide. Parton's music includes Recording Industry Association of America (RIAA)-certified gold, platinum and multi-platinum awards. She has had 25 songs reach no. 1 on the Billboard country music charts, a record for a female artist (tied with Reba McEntire). She has 44 career Top 10 country albums, a record for any artist, and she has 110 career-charted singles over the past 40 years. She has composed over 3,000 songs, including "I Will Always Love You" (a two-time U.S. country chart-topper, as well as an international pop hit for Whitney Houston), "Jolene", "Coat of Many Colors", and "9 to 5". As an actress, she has starred in films such as 9 to 5 (1980) and The Best Little Whorehouse in Texas (1982), for which she earned Golden Globe nominations for Best Actress, as well as Rhinestone (1984), Steel Magnolias (1989), Straight Talk (1992) and Joyful Noise (2012).
Extractive Summary	Dolly Rebecca Parton (born January 19, 1946) is an American singer-songwriter, actress, and businesswoman, known primarily for her work in country music. After achieving success as a songwriter for others, Parton made her album debut in 1967 with Hello, I'm Dolly, which led to success during the remainder of the 1960s (both as a solo artist and with a series of duet albums with Porter Wagoner), before her sales and chart peak came during the 1970s and continued into the 1980s. She has sold more than 100 million records worldwide. Parton's music includes Recording Industry Association of America (RIAA)-certified gold, platinum and multi-platinum awards. She has composed over 3,000 songs, including "I Will Always Love You" (a two-time U.S. country chart-topper, as well as an international pop hit for Whitney Houston), "Jolene", "Coat of Many Colors", and "9 to 5".
Abstractive Summary	Dolly Rebecca Parton is an American singer-songwriter, actress, and businesswoman . She made her album debut in 1967 with Hello, I'm Dolly, which led to success during the remainder of the 1960s . She has had 25 songs reach no. 1 on the Billboard country music charts, a record for a female artist (tied with Reba McEntire)

Table 1: An example of an input text and the resulting abstractive and extractive summaries. The input text is the first two paragraphs of the Dolly Parton Wikipedia page.[3] The extractive summary was generated using BERT Extractive Summarizer (Miller, 2019),[4] and the abstractive summary was generated using BART by following an example (Lewis et al., 2019).[5] The extractive summary consists of five sentences taken from within the input text (which have been underlined), while the abstractive summary is composed of entirely new sentences generated by the model.

[3] https://en.wikipedia.org/wiki/Dolly_Parton (accessed April 16, 2022)
[4] https://pypi.org/project/bert-extractive-summarizer/
[5] https://towardsdatascience.com/abstractive-summarization-using-pytorch-f5063e67510

A core component of either of these styles of summarization is deeming which portions of the input text are most important. There are numerous methods for extracting the most salient topics or textual sections from the input documents, and often these methods rely on the underlying statistics of the text being summarized. These statistics are pulled from an intermediate representation of the text that is generated by the summarizer in order to tease out salient points in the text (Allahyari et al., 2017). For instance, word and phrase frequency can be used to rank sentences in importance (Luhn, 1958). There has also been work exploring weighting words and sentences based on their location, similarity to the title of the document, and use of cue word contents (Edmundson, 1969). These techniques were some of the first methods explored, and since then an array of other techniques have been studied.

One such area of focus is topic-based approaches, where different techniques are used to focus on the actual topics being discussed in the text. For example, a popular frequency-driven technique is to utilize term frequency - inverse document frequency (tf-idf). This technique attempts to draw out the most important words in an individual document by penalizing words that appear frequently across all documents (Allahyari et al., 2017). Tf-idf is calculated by multiplying the term frequency (number of times a term appears in a document) by the inverse document frequency (number of total documents divided by the number of documents the term appears in), demonstrated in Figure 4. Then, the log of this value is taken to condense the values. This value acts as a weight for the given term, and in effect prioritizes terms that appear less frequently across all documents to avoid overly common words being rated highly. Besides

frequency-driven approaches, some other technique categories include latent semantic analysis and Bayesian topic models (Allahyari et al., 2017).

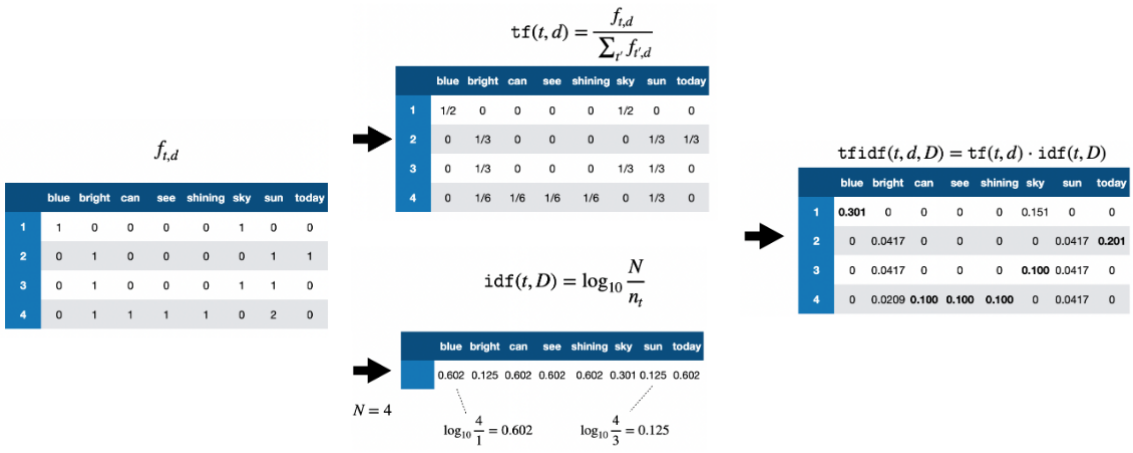

Figure 4: Steps for calculating tf-idf scores to create a vector representation of a document within a corpus (Brinton and Inouye, 2020). Tf-idf is a measure of how important a word is to a document (in our case individual tweets) with respect to the overall corpus (vocabulary of all documents being evaluated). $f_{t,d}$ represents the count of each word (represented by the columns) within each document (represented by the rows). These counts are used to calculate the individual *tf* and *idf* scores, which are then combined to yield the final *tf-idf* score for each word in each document. The *tf-idf* scores compose the vectors that represent each document within the representation blocks in Figure 2 and Figure 3.

Having chosen the most salient points to include in the summary using one of the aforementioned strategies, the final step is to generate the summary. For an extractive summary, a specific length (usually a number of sentences) is chosen by the user to determine the length of the generated summary. The model ranks the sentences based on a scoring mechanism with reference to the most salient points, and then chooses the top scoring sentences to construct the summary. Contrastingly, for an abstractive summary the model utilizes some sort of text generation component to generate a summary from scratch reflecting the salient points of the documents to be summarized.

Automatic summarization models are generally designed to work with traditional language, and much of the work in the area has been focused on news summarization (often as a byproduct of available datasets[6] and formality of the writing). As such, summarizing social media text offers an interesting and challenging problem, and Twitter specifically presents immense difficultly in automatic summarization due to the casual grammaticality of tweets (in regards to lax punctuation, non-traditional spelling, use of emojis/symbols, and other characteristics that do not align with more traditional text).

The social media sub-space of automatic summarization remains an active area of research, with work focused on gleaning information from the plethora of noisy, user-generated text. For example, research into using Twitter data to generate summaries of sporting events has been performed by Nichols et al. (2012), and there has also been work in summarizing events based on collected tweets (Chua and Asur, 2013). Furthermore, work has been done to summarize events via news articles with regard to social opinion/context provided by user-generated comments associated with those news articles (Ramón-Hernández et al., 2020). Additionally, there has been research into generating summaries on various topics of interest from Twitter data with a focus on maintaining the credibility of those summaries (Talarico and Viviani, 2021). It is clear that social media provides a wealth of textual information from which to glean insights, and the ability to condense the numerous opinions contained on Twitter and other social media platforms into a digestible size is incredibly beneficial.

[6] Summarization datasets are costly to make, as they usually require a human component in generating the target gold-reference summaries. News articles often contain an accompanying human-generated summary used as a preview of the article (such as in search engine results or social media posts), making them an efficient way to generate a dataset.

2.1.2 Evaluation of Automatically Generated Summaries

Evaluation of automatically generated summaries tends to rely on some human element, whether it is scoring them via human evaluation, or using automatic evaluation that compares against a human-generated gold-reference summary. Direct human evaluation is expensive, and as such automatic evaluation is a popular method. One of the foundational automatic summary evaluation suites is ROUGE, which stands for Recall-Oriented Understudy for Gisting Evaluation (Lin, 2004). ROUGE is a set of metrics that provide scores for an input summary when compared against a given reference, or gold, summary. These metrics can be classified into four categories: ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S (Lin, 2004). We now describe the different metrics used in this work.

ROUGE-1, a special case of ROUGE-N, measures the number of unigrams (ie: each single token or word) shared between the summary being evaluated and the gold target summary. Thus, it is the number of individual tokens shared between the two. ROUGE-2 is also a special case of ROUGE-N, where bigrams are used in calculating the scores. The output scores are broken down as recall, precision, and F1-score, which are defined as follows.

Recall represents the percentage of items present in the input that are also present in the target. It is calculated by taking the number of matches (ie: n-grams present in both the input summary and gold summary, where n = 1 to n) divided by the total number of n-grams in the gold summary.

Let $S_{input\ summary}$ be the set of all n-grams in the input summary and $S_{gold\ standard}$ be the set of n-grams in the gold standard summary. Recall is defined as the ratio of the number of elements in the intersection between these two sets divided by the number of elements in the gold standard set:

$$recall = \frac{S_{input\ summary} \cap S_{gold\ standard}}{S_{gold\ standard}} \quad (1)$$

Precision is very similar to recall, but instead of dividing by the total number of n-grams in the gold summary, the total number of n-grams in the input summary is used. This helps address cases where a model could learn to push out a very large number of words to attempt to "cheat" at its score.

Using the same definitions for $S_{input\ summary}$ $S_{gold\ standard}$ as above, precision is defined as the ratio of the number of elements in the intersection between these two sets divided by the number of elements in the input summary set:

$$precision = \frac{S_{input\ summary} \cap S_{gold\ standard}}{S_{input\ summary}} \quad (2)$$

Lastly, F1-score represents a combination of both recall and precision, and is calculated as follows:

$$F1\text{-}Score = \frac{2 * recall * precision}{recall + precision} \quad (3)$$

ROUGE-L score, a related metric, considers the longest common subsequence (LCS) shared between the input summary and the gold summary. It similarly is broken down into recall, precision, and F1-score for its output, and these are calculated as follows:

Recall is determined by dividing the number of tokens in the LCS shared between the input summary and gold summary by the total number of tokens in the gold summary.

$$recall = \frac{token\ length\ of\ LCS_{input\ summary\ and\ gold\ summary}}{total\ number\ of\ tokens_{gold\ summary}} \quad (4)$$

Like before, precision is calculated very similarly to recall, but instead the denominator is the total number of tokens in the input summary.

$$precision = \frac{token\ length\ of\ LCS_{input\ summary\ and\ gold\ summary}}{total\ number\ of\ tokens_{input\ summary}} \quad (5)$$

Lastly, F1-score is calculated the same as in Equation (3), as it still represents a combination of the recall and precision (Lin, 2004; Agrawal, 2021).[7]

[7] https://towardsdatascience.com/the-ultimate-performance-metric-in-nlp-111df6c64460#:~:text=ROUGE%2DN%20measures%20the%20number,consist%20of%20a%20single%20word

From ROUGE, the F1-score is often worth focusing on due to its nature of including both precision and recall. For this work, we opted to use ROUGE-1 and ROUGE-L F1-scores for our evaluation purposes.

2.2 Bias and Fairness in NLP

Bias and fairness research in NLP is a fairly young area, but is growing in popularity as NLP systems permeate our lives. In exploring these categories, there are a few core definitions that should be examined to lay the groundwork for looking at this work.

2.2.1 The Importance of Analyzing Bias

Since many of the biases[8] exhibited within NLP models are socially constructed, the definition of what a bias is can vary between people, cultures, languages, etc. Determining whether something constitutes bias is a highly subjective process, and it is important to develop a foundation for work exploring bias to build upon. As noted by Blodgett et al. (2020), many recent papers "fail to engage critically with what constitutes "bias" in the first place."

Bias is a very broad and difficult term to define with regards to NLP models, as it manifests differently depending on the use of the model. Additionally, bias impacts people to varying degrees depending on the groups that they belong to, or due to their personal characteristics. Certain instances of bias are more straightforward to address,

[8] Here the bias being discussed is different from the statistical term frequently used in machine learning literature.

such as displaying favoritism for a certain gender over others, but others can be more subjective, such as defining whether an online communication is "toxic."

One approach proposed by Blodgett et al. (2020) is for authors to outline the conceptualizations of bias that guide their work. Specifically, this would consist of outlining what harmful system behaviors are being explored within the research, why they are harmful, and who they harm. Including such a formal definition of the bias explored within a paper helps ensure the work is centered in normative reasoning, rather than based upon unstated assumptions (Blodgett et al., 2020).

Furthermore, researchers have developed definitions for different types of bias based on the source. This approach of breaking bias into different sub-types allows for research within bias to be more easily correlated to similar cases. Mehrabi et al. (2021) present nineteen types of bias with distinct definitions that they collected from prior research in the space, and determine that bias can be categorized by where it manifests within the data, algorithm and user interaction loop.

Overall, it has become apparent that a foundational definition for bias in NLP is vital for us to build upon past work, rather than continuously starting from scratch with each new attempt. Bias is a very broad term, and as such determining categories to break it into sub-groups makes it more manageable. Additionally, setting expectations for information provided when presenting research on bias in NLP helps unify work in the field.

2.2.2 What it Means to be "Fair"

In working with bias, it is also important to reference fairness since the goal of mitigating bias is to achieve fairness within model performance. Examination of fairness has occurred in numerous disciplines for more than fifty years (Hutchinson and Mitchell, 2019), but is a more recent topic within machine learning.

One of the earliest academic explorations of fairness was in the realm of testing (Hutchinson and Mitchell, 2019). This research explored bias in tests based on race, and worked to determine metrics for performing this evaluation. This work evolved over the years to incorporate various metrics, and much of this work can be connected to current work within the realm of fairness in machine learning (Hutchinson and Mitchell, 2019). Within their work, Hutchinson and Mitchell (2019) provide an analysis of these connections, grounding current work with historical research. An area of interest explored within their work is the idea of whether fairness is a property of a model, or of its use. They cite that this is an area that has been highly contended within historical research, but is largely absent from research of machine learning models.

However, while this specific discussion may be largely unexplored within machine learning, work has been done on both sides of the fence. Many researchers have performed work to mitigate bias within models (Rathore et al., 2021; Krasanakis et al., 2018; Dixon et al., 2018), while other work has demonstrated that the choice of dataset and model directly impacts bias manifestation (Hovy and Prabhumoye, 2021).

Similar to defining bias, defining fairness is also an area of interest to researchers. Within their work, Hutchinson and Mitchell (2019) provide a summary of ten definitions

of fairness they deem important, which they ultimately categorize into three core types: individual fairness, group fairness, and subgroup fairness. Individual fairness and group fairness are based on whether the fairness definition is applicable to a singular person or a group. Subgroup fairness, on the other hand, attempts to pull from both of these categories to generate a blend of the best properties of each.

Overall, fairness is a foundation for work in bias, and having clear definitions of it to work with is pivotal. The works above help shed some light on how fairness can be measured and displayed within research, and how these methods can be applied to research into bias in NLP.

2.2.3 Fairness and Bias in Automatic Summarization

Researching bias in NLP is often task specific, as the intended use of a model dictates how any biases it contains could manifest. A specific area that is fairly new to bias research is in the field of automatic summarization. Even so, there have been some initial explorations into the area.

Dash et al. (2019) presented a first attempt at developing an algorithm for summarizing focused on producing fair summaries. Within their work, they outline that "most existing summarization algorithms do *not* fairly represent different groups," and recommend approaching summarization from a new evaluation perspective that places importance on fair representation of different social groups within the generated summaries (Dash et al., 2019). In order to accomplish this, they present new notions of fairness to use in measuring generated summaries. They identify equal representation

and proportional representation as guides for evaluating extractive summaries, and their work demonstrates that existing algorithms do not generate fair summaries. To combat this, they provide pre-processing, in-processing, and post-processing methodology for achieving fairer summaries and use these methodologies to generate corresponding algorithms. A weakness of their work, though, is that their methods require the input documents to be labeled as the classes that are being measured for representation. Overall, the work of Dash et al. (2019) identified the lack of fairness in current automatic summarization algorithms at their time of publishing and the motivations for why this is an area worthy of further research.

A subsequent work in this area proposes a method for using a fairness-aware summarizer to condense inputs to a model by extracting relevant information and removing demographic information (Keymanesh et al., 2021). The authors present their own fairness-aware summarization model, and also demonstrate its use in their overall methodology, along with some comparison to some other models available.

An area lacking research in this space is the application of fairness concepts to abstractive summarization (Dash et al., 2019). It is much easier to gauge representation for extractive summarization since it directly pulls text from the original source documents, as opposed to measuring fairness for abstractive summarization which generates new text that can be more difficult to link back to the original input text. However, despite extractive summarization presenting easier fairness analysis, there is still a gap in considering bias and fairness in automatic summarization research in general

and techniques to counterbalance unfair behavior in models, which is a major motivator for performing this work.

This work examines ways to glean salient and more diverse perspectives as a way to incorporate them in generated summaries in addition to more common perspectives. This, in turn, can help ensure more diversity within automatically generated summaries by preventing only the most popular and common perspectives from being included.

2.3 Clustering Techniques

2.3.1 K-Means

K-means clustering is an algorithm that separates input data into clusters based on some sort of vectorization of that data. It is effective at identifying commonalities amongst data, and can be used to label data by determining which cluster it belongs to. The technique was named back in 1967 (MacQueen, 1967), and has been studied extensively ever since.

The algorithm for *k*-means consists of the following steps: first, the user must identify how many clusters to organize their data into, *k*. The best value of *k* is often not obvious, and typically relies on some sort of statistical method or trial and error to determine. This often is in the form of an elbow graph or silhouette scores. Once the value of *k* has been determined, the centroids of each cluster are randomly determined (MacKay, 2003). From this point onward the algorithm assigns each data point to the cluster it resides nearest to, and once all data points have been assigned the cluster

centroids are updated to best reflect the true center of the data points they contain (MacKay, 2003). From this point on, the algorithm repeats the cycle of assigning data points to clusters and then updating the cluster centroids until convergence of the cluster centroids occurs (MacKay, 2003). Convergence of the cluster centroids is determined by the centroids remaining fixed between iterations. Once convergence is complete, the clustering of data is obtained. Please see Figure 5 for an illustrated example of this process.

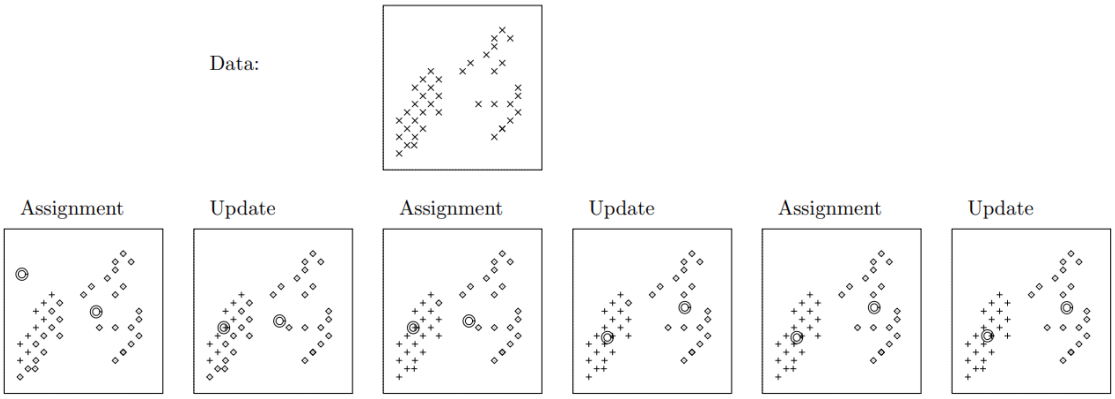

Figure 5: A simple example of *k*-means clustering applied to a set of 40 data points (MacKay, 2003). For this example, *k* is set to 2 indicating there are two centroids (the hollow circles), and the data is grouped into two clusters (consisting of + and ◊ shapes). Initially in the first assignment box, the cluster centroids are initialized and the data is grouped to each centroid based on proximity (indicated by the different shapes). Then, during the update phase the centroids are moved to the center of the data assigned to them, and the process reassigns the data points based on the new centroid locations. This repeats until convergence is achieved in the final Update box.

Use of *k*-means requires that the data points be vectorized in some way, and for the clustering to be most successful this vectorization must effectively reflect the nuances of the data. For NLP tasks, this consists of converting textual qualities of the data into a numerical vector format, called embeddings, and there are many techniques that can be

used to generate these embeddings (Li and Yang, 2018). The length of the vectors is referred to as the number of dimensions present, and this dimensionality is constant amongst all of the data points. Also, similar to the data vectors, the cluster centroids are also represented by vectors of identical dimensionality to the data points. Then, when the comparisons occur within the *k*-means algorithm, each dimension of the data points is directly compared to the corresponding dimension of the cluster centroid. Thus, in essence the clustering exists in *n*-dimensional space, where *n* is the size of the vectors being used. This can make visualizing clusters difficult when a high dimensionality is used.

While the basis for *k*-means is simple, there is plenty of room for modification to the base algorithm to achieve desired results. Two specific instances relevant to our work are multi-view clustering and fairness-aware clustering (which are discussed later in the chapter).

2.3.2 Cluster Analysis

A core component of *k*-means clustering is the analysis of the clustering. There are various methods for accomplishing this, and one of the core places this analysis takes place is during the iteration phase of *k*-means that adjusts the cluster centroids until they converge. In comparing the distance of a data point from a cluster centroid, there are different metric options available that are better for different cases. The first and most common distance metric is Euclidean distance, which represents the distance between two points and is calculated using the Pythagorean theorem (Mulak and Talhar, 2015).

More specifically, it is calculated by taking the square root of the sum of squared differences between the objects being compared (Grabusts, 2015). It is considered to be an efficient choice (Mulak and Talhar, 2015), but can pose issues in high-dimensional spaces due to becoming inflated (Aggarwal et al., 2001).[9] Manhattan distance, also referred to as city block distance, is also used in cluster analysis. It is calculated by summing the absolute differences between objects being compared (Grabusts, 2015). Lastly, another metric used is the Chebychev distance, also known as the maximum value distance. It is calculated by determining the largest absolute difference between any given dimension between objects being compared (Mulak and Talhar, 2015).

Another aspect of clustering that requires analysis is determining the correct number of clusters for the data, which relies on measuring the cluster quality within clusterings of various sizes (ie: various numbers of clusters) to assess which value of *k* yields the highest cluster quality. A majority of cluster quality metrics are extrinsic, meaning they rely on supervised data where the ground truth label is available (Rizk, 2020). Extrinsic metrics, such as purity, precision, recall, and normalized mutual information, compare the ground truth label against a clustering to determine how well the data was clustered (Rizk, 2020). However, oftentimes NLP data does not have ground truth labels to work with, and thus requires looking beyond extrinsic metrics for measuring cluster quality.

In cases where data is unsupervised, intrinsic metrics must be used. They use measures of cluster separation and compactness to evaluate the quality of a clustering

[9] https://scikit-learn.org/stable/modules/clustering.html#k-means

(Rizk, 2020). A popular intrinsic metric for analyzing text-based clustering from this regard is the silhouette coefficient (Rizk, 2020). This technique develops a silhouette for each cluster based on its tightness and separation to give an assessment of overall cluster quality (Rousseeuw, 1987). The silhouette is constructed using both the partition itself provided by the clustering algorithm and all of the proximities between objects in the data set (Rousseeuw, 1987). The silhouette coefficient used in silhouette analysis represents the cohesion and separation of both the individual data points and the clusters themselves (Jin, 2008). Silhouette coefficients are calculated for each data point, and the process for obtaining these values is shown in Figure 6 and Equation (6). The silhouette coefficients that are calculated per data point are then averaged for all data points within a cluster or clustering to obtain an average silhouette width representation of that cluster or clustering, respectively (Jin, 2008).

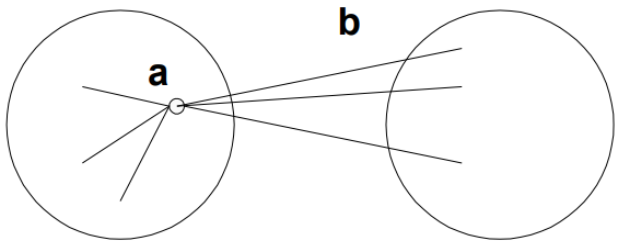

Figure 6: A representation of two clusters and a data point for demonstrating calculating the silhouette coefficient (Jin, 2008). To calculate the silhouette coefficient for an individual data point, first calculate (a) the average distance of the data point to all other points within its own cluster, and (b) the minimum of the average distances of the data point to other data points in another cluster. Then, using these values *a* and *b* the silhouette coefficient *s* can be calculated as shown in Equation (6) (Jin, 2008). The resulting silhouette coefficient is typically between 0 and 1, and values closer to 1 are considered better.

$$s = 1 - \frac{a}{b} \quad \text{if} \quad a < b$$

$$\text{or} \tag{6}$$

$$s = \frac{b}{a} - 1 \quad \text{if} \quad a \geq b$$

The silhouette representation of a cluster is comprised of two dimensions: the horizontal spread represents the confidence in each clustered piece of data, and the vertical spread represents the size of the cluster. In looking at the vertical spread, this represents the size of the cluster in that each data point contained in that cluster is stacked on top of one another. Thus, more data points will result in greater vertical spread. Then, each of these data points plotted has a horizontal spread that represents the confidence in clustering that piece of data. This confidence is represented by a score from -1 to 1, where a greater value equates to greater confidence (ie: a score of 1 represents the highest possible confidence) (Rousseeuw, 1987). The wider the silhouette, the more pronounced the cluster is (Rousseeuw, 1987). An example silhouette plot can be seen in Figure 7.

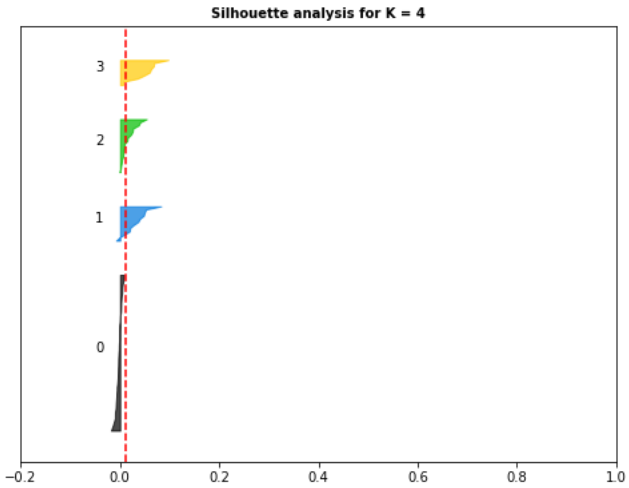

Figure 7: Silhouette plot of a clustering of size k=4. Each color represents a separate cluster, where the vertical breadth is representative of the number of data items contained in that cluster and the horizontal breadth is the confidence in categorizing each of those items. The dotted red line represents the silhouette coefficient, which is an overall representation of the clustering. Thus, cluster 0 is the largest cluster and cluster 3 contains the highest confidence values.

Beyond silhouette analysis, cluster cohesion and separation can also be measured in other ways and used to evaluate the quality of clusters. Cluster cohesion is a measure of how closely related the data within a cluster is (Jin, 2008). It can be calculated using the sum of squares of all items within the cluster, and can also be measured using a graph-based approach that considers the weight of the connections between all data points in the cluster (Jin, 2008). Separation, in contrast, represents how distinct an individual cluster is from all other clusters (Jin, 2008). It can be calculated using the between cluster sum of squares, or using a graph-based approach that considers the weight of the connections between all data points within the cluster to those outside of the cluster (Jin, 2008).

2.3.3 Multi-view Clustering

Multi-view clustering is a technique very similar to *k*-means in that it attempts to cluster data into similar groups based on certain metrics. However, where *k*-means relies on one vector, or view, to accomplish this, multi-view clustering considers multiple forms of feature information to develop different views of the data being clustered (Chao et al., 2017). More specifically, multi-view clustering relies on different available descriptions of the dataset to form different views that are all used to cluster the data simultaneously. These different views can be formed using any metric that can be used to describe all of the data in the data set to be clustered. For example, if trying to classify tweets views could be developed consisting of: 1. textual data of the tweet, 2. hashtags, 3. mentions, and 4. author biography content. Considering all of this information, as opposed to one singular view, can lead to a more nuanced clustering of data. For NLP consideration, multi-view clustering has seen use in clustering documents that exist in multiple languages (Kim et al., 2010) and also for clustering hashtags on Twitter (Cruickshank and Carley, 2020).

2.3.4 Fairness-aware Clustering

Clustering techniques can be used to make decisions that have great impact on people's lives. A technique to help ensure fairer clustering is fairness-aware clustering, which seeks to build clusters that are representative in distribution to the entire population (Abbasi et al., 2021). That is, in addition to clustering the data based on a feature vector, it also considers the distribution of data within each cluster and favors clusters that have a

distribution more similar to the overall dataset. However, to accomplish this the data must contain some label that can be used to measure representativeness of the clusters. For example, if data being clustered contains a gender label, that label could be used to build clusters that contain a distribution of genders most similar to the overall gender distribution of the data population being clustered. The basis for this technique is rooted in equitable representation, and achieves this by adding a metric in addition to the normal distance metric for cluster evaluation during the iterative clustering process (Abbasi et al., 2021). Rather than purely seeking convergence on cluster centroids, the fairness-aware clustering algorithm also considers the representation amongst clusters and considers solutions with more equal representation more favorably. This ultimately yields clusters that are better representations of the total population. An example of where this could be used is in determining polling locations that are equitable to different subgroups of voters (Abbasi et al., 2021).

3 Diversity of Extractive Summaries

There are various algorithms that take input text and return a summary of that text. In generating these summaries, questions about representation and diversity come to the forefront. Do automatic summarizers give preferential treatment to specific perspectives? Do they unfairly avoid including text from certain dialects? Exploring this subspace is the goal of this work.

This section outlines analysis performed on a dialect-diverse summarization dataset of social media data in an attempt to find a way to hone in on the quantity of diversity reflected in extractive summaries generated from these tweets. It first lays some foundation for the dataset being explored (sec. 3.1), then covers the analysis performed on human-written summaries (sec. 3.2) and machine-generated clusters (sec. 3.3), followed by a novel approach for generating extractive summaries (sec. 3.4) and a discussion of the evaluation results (sec. 3.5), and finally outlines the implementation setup used to perform this work.

3.1 Dataset

To begin exploring this problem, we decided to use the DivSumm dataset (Olabisi et al., 2022), which is a collection of 450 tweets and corresponding extractive and abstractive summaries evenly distributed across five topics: `Beyonce`, `Christmas`, `NBA`, `Netflix`, and `Obama`. Each topic subset of tweets contains 90 tweets evenly distributed across three dialects (African American English, Hispanic English, and white

English) resulting in 30 tweets per dialect. This yields a breakdown of 20 total subsets of tweets within the 450 total tweets (4 tweet subsets per topic: All tweets, African American English tweets, Hispanic English tweets, white English tweets) that are each accompanied by two human-generated abstractive summaries and two human-generated extractive summaries. The annotation was performed by six annotators whose self-expressed identities were diverse amongst the dimensions of gender, ethnicity, and sexual orientation. For a visual presentation of this breakdown, please refer to Table 2.

Topic	Tweet Subsets			
	African American English	Hispanic English	White English	All Tweets
Beyonce	30	30	30	90
Christmas	30	30	30	90
NBA	30	30	30	90
Netflix	30	30	30	90
Obama	30	30	30	90

Table 2: Breakdown of tweets contained in DivSumm. The tweets are categorized by five topics and three English dialects.

DivSumm was built using tweets from the Twitter AAE Corpus (Blodgett et al., 2016), which is a corpus consisting of 59.1 million tweets with corresponding dialect confidence scores. These confidence scores were provided by a model that was used to identify "demographically-aligned text and language from geo-located messages" (Blodgett et al., 2016).

In selecting tweets from Twitter AAE Corpus to use in DivSumm, there were some requirements created to ensure their quality. First, tweets were only considered if they had an associated dialect confidence score of greater than or equal to 0.7 for a given

dialect. Additionally, tweets needed to be at least seven tokens in length, and any duplicate tweets were also not considered. A tweet also could not be a mention (ie: @username), as this presented a high chance of it being a duplicate tweet, as well. Using these specifications, the viable tweets were collected and analyzed for potential topics that yielded enough tweets across the three dialects to yield the desired distribution of thirty tweets per dialect per topic. This resulted in the topics selected as `Beyonce`, `Christmas`, `NBA`, `Netflix`, and `Obama`.

A core component of DivSumm is that hashtags and emojis were purposefully left intact, as they are a ubiquitous part of the Twitter universe and may provide useful information. This must be taken into consideration when working with DivSumm, as some traditional NLP techniques will require additional steps to work around the presence of hashtags and emojis. A sample tweet is depicted in Table 3.

Original Tweet	Beyonce flow right now \ud83c\udfa4\ud83c\udfa7\ud83c\udfbc \nThat's always though . #teamBey.
Dialect	African American English
Tweet With Emojis	Beyonce flow right now 🎤🎧🎼 \nThat's always though . #teamBey.
Tweet With Emojis as Text	Beyonce flow right now :microphone: :headphone: :musical_score: \nThat's always though . #teamBey.

Table 3: Sample tweet from the DivSumm dataset included in the `Beyonce` topic. It depicts the original tweet, the tweet's dialect (where AA is the abbreviation for African American English), the tweet with emoji codes converted into the emoji pictures, and the tweet with the emoji pictures converted into textual representations.

DivSumm presents an interesting opportunity in that it contains dialect labels associated with its tweets. This is a unique characteristic that is not common-place, and as such allows for novel exploration in how NLP models interact with tweets of varying dialects.

Pre-processing: We pre-processed the dataset by converting emojis into text form so that the emojis would be considered in our work. For example, if ❤️‍🩹 was present in the original tweet, the version of the tweet we used would contain ":red_heart:" in its place. Additionally, we performed some pre-processing to better prepare the tweets for clustering. First, we removed commas from the tweets and also made the tweets entirely lowercase. This was to avoid instances of the same word with different capitalization or punctuation being treated as different words. Next, we removed stop words using the NLTK[10] stopwords list since they are often plentiful but provide minimal information about the content of a tweet. Following the removal of stop words, we also removed the name of the DivSumm topic corresponding to the tweets. For example, we removed 'beyonce' from all of the tweets in the `Beyonce` set of tweets. Due to the process of generating the DivSumm dataset, every tweet within a category contains the category title within the tweet, which causes those words to contribute very little actual information about the content of the tweet. By removing them, we make room for more interesting information to come forth. Lastly, we performed stemming on the tweets to help lump words that belong to the same family but have different endings together. For example, words can end in suffixes like -ed and -ing but refer to the same base word and stemming

[10] NLTK: https://www.nltk.org/data.html

will reduce them to that base word allowing them to be treated as the same instance of that word.

Once this pre-processing of the tweets was complete, we were ready to explore our research questions.

3.2 Exploration 1: How diverse are human-generated summaries?

Since a major focus of our work centers around diversity and representation within automatic summaries, we first decided to examine the diversity of the gold extractive summaries that have been created by human annotators.

Extractive summaries are generated by selecting direct subsets of the input documents to generate a summary of the entire set of input documents. In the case of DivSumm, the annotators were instructed to select the five most representative tweets from the set of tweets on each topic. For this dialect representation exploration, we examined the extractive summaries generated for each topic's set of 90 tweets containing all three dialects. This resulted in two extractive summaries per topic to look at, for a total of ten extractive summaries.

Figure 8 displays the overall representation of each dialect across all ten of the extractive summaries. The representation value was determined by calculating the number of tweets per dialect present in the extractive summaries divided by the total number of tweets contained within all of the summaries. The results show that each dialect's representation fell within the 28-38% range, indicating a fairly equal distribution

across the dialects. **This was very encouraging to us, as it demonstrates that the human annotators developed diverse summaries that represented all dialects well.**

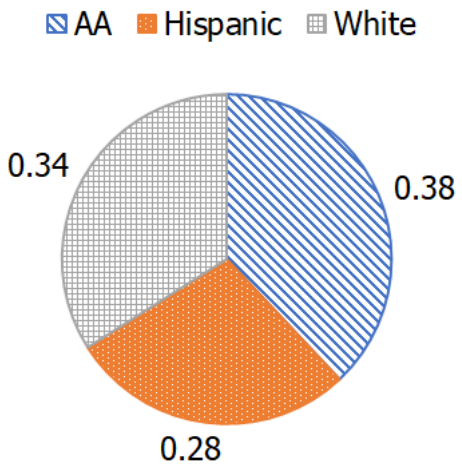

Figure 8: Displays the representation for each dialect within the two human-generated extractive summaries per topic in DivSumm. Each dialect's representation is in the range of 28-38%, which indicates fairly diverse representation across the extractive summaries.

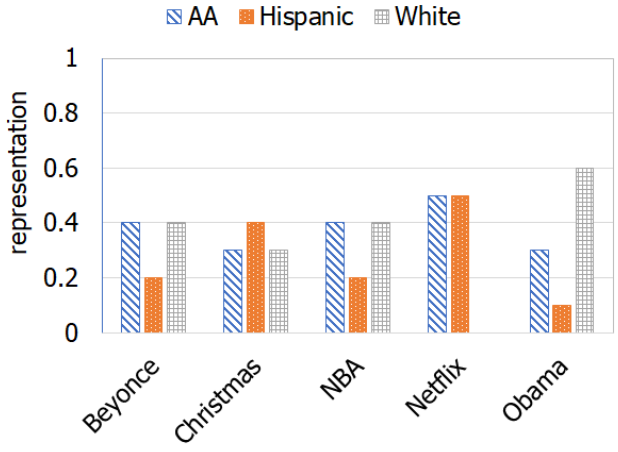

Figure 9: Presents a breakdown of dialect representation of each human-generated extractive summary per topic. This corresponds similarly to the representation trends seen in Figure 8, indicating that trend is consistent across topics.

To examine this further, Figure 9 provides a representation breakdown per topic that considers both of the extractive summaries generated per topic for the set of tweets containing all three dialects. As seen in the results, the trend of diverse summaries was consistent across topics.

Having determined that human-annotators developed diverse summaries for DivSumm, we were also curious to measure the inter-annotator agreement within the extractive summaries. To examine this, we compared the tweets selected per topic by each annotator to calculate how often the two annotators selected the same tweets for their summaries. Due to the fact that the annotators were tasked with selecting five out of a set of either thirty or ninety tweets that they deemed most representative of the entire set of tweets, and in selecting these tweets were likely focused on pulling out tweets with themes shared between multiple tweets yielding multiple options per theme, it was expected that the overlap would be quite low. As seen in Figure 10, this was the case for the white tweets, but the African American (AA) and Hispanic tweets were considerably higher in their overlap.

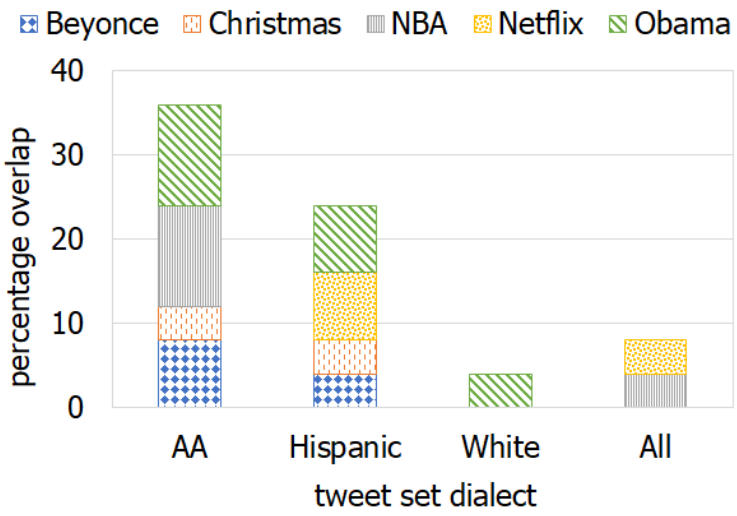

Figure 10: Depicts the percentage of overlap of tweets selected by a pair of annotators for their extractive summaries for each set of tweets (ie: a set of thirty tweets for each dialect, and a set of ninety tweets containing all tweets for that topic) within DivSumm. The percentage overlap was expected to be fairly small, since the annotators were each selecting five out of thirty/ninety tweets. Interestingly, though, we found that the percentage overlap was quite high for both the African American (AA) and Hispanic tweets, which is explored further in section 3.2.

Length-based analysis: To explore this result further, we performed a brief length analysis of the tweets to try to make sense of this trend. In Figure 11, the average length of the tweets chosen by annotators per set of tweets is displayed, as well as the average length of tweets in the overall sets. This comparison was made to determine if we could see any correlation between chosen tweet length and overall tweet length to help explain the annotator overlap. Considering that Twitter imposes a limit on the number of characters per tweet, and also that part of the creation process of DivSumm consisted of removing tweets under a certain length, it wasn't surprising that length did not appear to play a role in the annotator overlap. Across each dialect, the average tweet length of tweets chosen per dialect to compose the extractive summaries corresponded very closely to the average of all tweets within that dialect.

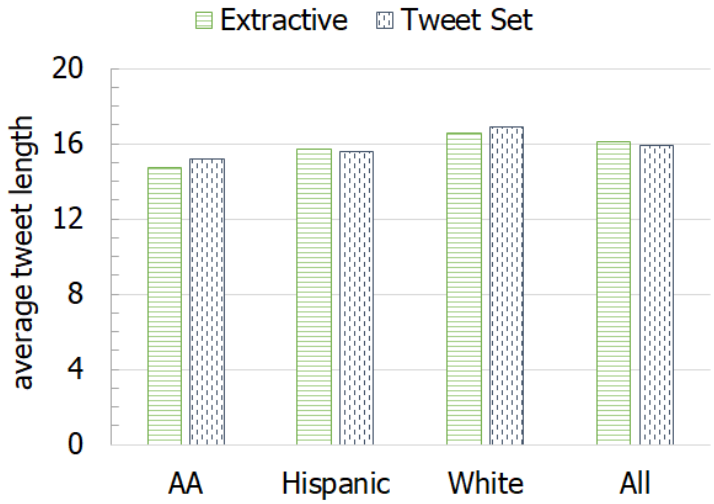

Figure 11: Depicts an analysis of the average tweet length of both the extractive summaries (Extractive) and the overall tweet set for each dialect (Tweet Set). Given the character limit imposed upon tweets, it is unsurprising that length did not appear to be relevant in explaining the annotator overlap.

Sentiment-based analysis: Having ruled out length as a factor, we also decided to explore the sentiment of tweets contained within each dialect and each topic. We speculated that humans may gravitate towards tweets of a specific sentiment in crafting their extractive summaries. More specifically, we hypothesized humans may choose more positive tweets and also may avoid tweets with profanity (which would be included in negative sentiment). Thus, we proposed the idea that if a dialect contains an above average amount of negative tweets, it would shrink the actual pool of tweets the annotators chose from and would lead to increased annotator overlap. Therefore, we counted the number of tweets with negative sentiment per dialect per topic. Concretely, the sentiment of the tweets was measured using NLTK's VADER module,[11] which

[11] NLTK's VADER module: https://www.nltk.org/api/nltk.sentiment.vader.html

calculates an overall sentiment intensity score for a piece of text by summing the sentiment intensity values of each token/word present in that text obtained using a sentiment lexicon (Hutto and Gilbert, 2014). A score of greater than zero corresponds to an overall positive sentiment, while a negative score corresponds to an overall negative sentiment, and a score of zero indicates an overall neutral sentiment. For this analysis, each tweet was binarily classified as either positive/neutral or negative based on the corresponding VADER score, and counted accordingly.

The results of this analysis are shown in Figure 12, where we notice a small trend in that the `Obama` white tweets contained more negativity, and `Obama` was also the only topic that contained any overlap amongst White tweets. However, this minor correlation does not explain the trend of how much higher the annotator overlap was for both the AA and Hispanic dialects. Thus, at this point we were unable to find any specific correlations explaining the increased overlap for the AA and Hispanic dialects.

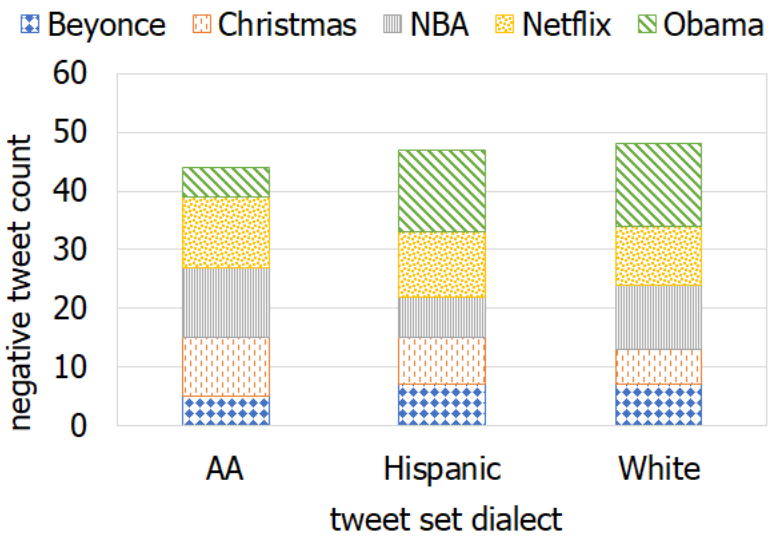

Figure 12: Displays an exploration of the distribution of negative tweets per topic per dialect, with the idea that annotators may gravitate towards tweets of a specific sentiment in their tweet choices for the extractive summarization task. These results indicate that while sentiment may display some weak correlation (ie: the `Obama` tweets in White dialect had a higher than average negative sentiment), it does not fully explain the overlap.

Recap: Having analyzed the human-generated extractive summaries of DivSumm, and determined that the human annotators created fairly balanced summaries with respect to the dialect of tweets chosen, we next explore the diversity of automatically clustered tweets.

3.3 Exploration 2: Analyzing automatic clustering of dialect-diverse tweets

K-means clustering is a popular algorithm within extractive summarization tasks, as it provides a way to separate sentences (or other textual components, such as tweets) into separate groups and score the sentences based on their proximity to the cluster centroid (Akter et al., 2017; Agrawal and Gupta, 2014; Prathima and Divakar, 2018). This

scoring provides a way to use various heuristics for extracting sentences to be used in summary generation. For example, research has been done selecting the sentences closest to the centroid from only the densest cluster to generate the final summary (Agrawal and Gupta, 2014), and another method selects sentences closest to the centroids of both clusters in a clustering of size two (Akter et al., 2017).

3.3.1 K-means clustering and cluster analysis

Having explored the gold extractive summaries within DivSumm, we next sought to answer the question of what would happen if we put dialect-distinct tweets through a clustering algorithm. Would the resulting clusters be focused on topic or dialect? To answer this, we decided to use *k*-means clustering with term frequency – inverse document frequency (tf-idf) as the feature vector to cluster DivSumm. We applied *k*-means clustering with tf-idf to each set of 90 tweets (containing all dialects) for each topic in DivSumm[12]. We generated clusterings of size k=2 through k=8 for performing the initial cluster quality assessment via silhouette analysis (please refer to Appendix A for the silhouette plots), which helped us determine the number of clusters that best fit our data.

Next, we examined whether our clusters were based on topic or dialect. We found that across all topics there was no instance of clustering with a distinct reference to clustering by dialect, and a majority of clusters contained fairly diverse representation.

[12] Built upon code in the notebook https://nbviewer.org/github/LucasTurtle/national-anthems-clustering/blob/master/Cluster_Anthems.ipynb for clustering, silhouette analysis, and generating the top word plots and word clouds.

Table 4 shows an example of the representation breakdown per cluster for the `Beyonce` tweets, while Table 5 presents the range of the distribution (i.e., the difference between the highest and the smallest values) as an indicator of the differences in the representation. Smaller range scores indicate a more balanced cluster, and we note that a majority of scores of dialect representations per cluster fell within the 0.1-0.5 range. The highest range encountered was 0.714, indicating that cluster contained substantially more tweets of one dialect from the others, but still was not entirely composed of one dialect. These findings were promising to us, as they indicate that the clustering algorithm did not cluster based on dialect and instead likely focused on topic.

Beyonce Dialect Distributions Per Cluster					
Clustering	Cluster	AA	Hispanic	White	Total Tweets
k=6	0	2	2	5	9
	1	5	6	2	13
	2	14	11	18	43
	3	2	4	0	6
	4	1	6	3	10
	5	6	1	2	9
k=2	0	22	22	26	70
	1	8	8	4	20

Table 4: Number of tweets per dialect per cluster for the *k*=6 and *k*=2 clusterings on the `Beyonce` set of 90 tweets.

Dialect Distribution Ranges Per Topic and Cluster						
Clustering	Cluster	Beyonce	Christmas	NBA	Netflix	Obama
k=6	0	0.333	0.152	0.100	0.375	0.344
	1	0.308	0.333	0.241	0.500	0.462
	2	0.163	0.500	0.500	0.114	0.100
	3	0.667	0.273	0.167	0.182	0.400
	4	0.500	0.133	0.118	0.154	0.154
	5	0.556	**0.714**	0.643	0.500	0.417
	Average	0.421	0.351	0.293	0.304	0.313
k=2	0	0.057	0.088	0.261	0.115	0.038
	1	0.200	0.700	0.090	0.158	0.300
	Average	0.129	0.394	0.175	0.137	0.169

Table 5: Ranges of representation difference between dialects for each cluster within each topic. The range was calculated by taking the difference between the maximum dialect representation value and the minimum dialect representation value. Dialect representation was calculated by taking the total number of tweets of a specific dialect within a cluster divided by the total number of tweets within that cluster. The average range represents the average across all clusters per topic. The highest range within the entire table is depicted in bold.

Recap: Having determined that the *k*-means tf-idf clustering approach clustered the DivSumm dataset with regards to topic and not dialect, and yielded clusters of diverse dialect representation, we decided to next explore the topics present in the clusters.

3.4 Exploration 3: Extractive summarization with and without clustering

After showing that *k*-means clusters DivSumm based on topic rather than dialect, we consider incorporating clustering into our multi-document extractive summarization algorithm to assess whether this process can improve the final summaries. This required the following steps: cluster the tweets with *k*=2, generate automatic extractive summaries

of each cluster, and evaluate their effectiveness by calculating ROUGE scores for each summary against each gold extractive summary.

We decided to use BERT-Ext[13] model to generate the extractive summaries of each cluster (Miller, 2019). BERT-Ext is built upon BERT (Bidirectional Encoder Representations from Transformers), which is a bidirectional encoder model published by Google AI Language in 2019. BERT was trained on BooksCorpus and English Wikipedia, and initially was released with two main variants: BERT-base consisting of 12 layers, 768-dimension hidden states, 12 attention heads, and 110 million parameters, and BERT-large consisting of 24 layers, 1024-dimension hidden states, 16 attention heads, and 340 million parameters. BERT comes with a pre-trained base model that the user can fine-tune on their desired downstream task. The pre-training part of BERT is very expensive, and consists of using two unsupervised tasks: masked language modeling and next sentence prediction. Luckily, the pre-trained model is available for use, allowing users to skip ahead to the fine-tuning portion which is considerably cheaper. Fine-tuning allows the user to modify BERT to better work for their desired downstream task, making BERT a very flexible model (Devlin et al., 2018).

Unsurprisingly, the arrival of BERT has yielded many modified models honing in on specialized areas. BERT-Ext is no exception, as it is a modification of BERT designed with extractive summarization in mind (Miller, 2019). It relies on BERT-large for the text embedding component of their method, and then utilizes *k*-means clustering

[13] BERT-Ext: https://pypi.org/project/bert-extractive-summarizer/

to cluster the embeddings and extracts sentences closest to the centroid to generate the extractive summaries.

In working with BERT-Ext, its base use case is designed to take in a contiguous piece of text, break it apart in to sentences, and then utilize those as the individual components of the document for analysis and in returning an extractive summary of the desired length. This does not work exactly as intended for tweets, which can be composed of multiple sentences each, and also tend to have non-traditional punction. This makes breaking the tweets apart more difficult, and as such the resulting summary is not exactly five tweets in length. However, since all of our summaries generated were constrained to the same five "sentences" in length stipulation, we felt that comparisons between them were fair.

We used this process to generate four summaries per topic using the following subsets of tweets:

- **(Baseline) All tweets approach:** generate a summary by using all 90 tweets as input to the summarizer;
- **Cluster 0 approach:** generate a summary by using only the tweets contained in cluster 0;
- **Cluster 1 approach:** generate a summary by using only the tweets contained in cluster 1;
- **Combo approach:** generate a summary by using the summaries generated via the Cluster 0 approach and Cluster 1 approach above.

Please refer to Figure 13 for a visualization of the pipeline for each of the approaches, Figure 14 for a sample of the first three summary types, and to Figure 15 for the fourth summary type.

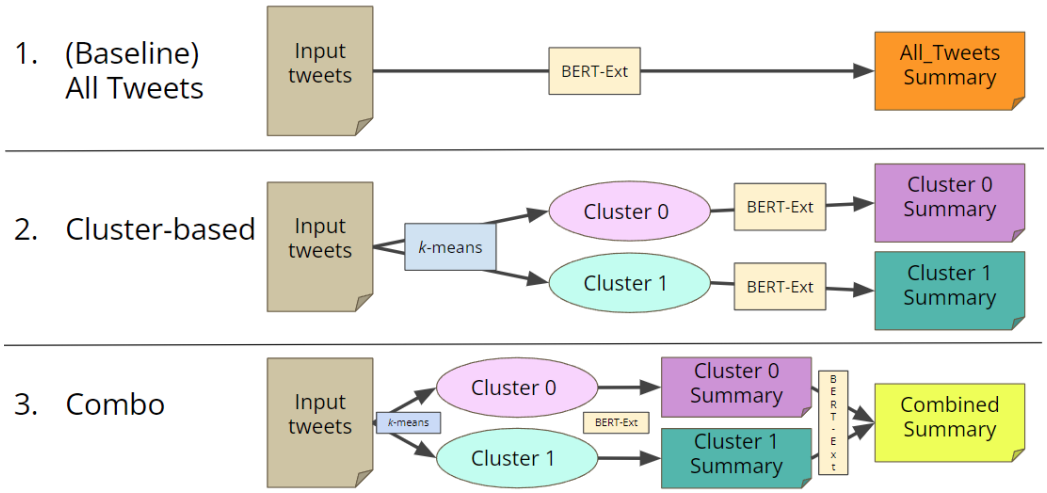

Figure 13: Visual depiction of the approach pipelines for generating each of the four summaries.

Summary of `Beyonce` All Tweets:
Beyonce flow right now :microphone: :headphone: :musical_score: \nThat's always though . # Snoozer Ok I know I'm gonna get stoned for this but I just heard Beyonce star spangled banner and it was actually average to me ... I like Michelle's hair better with bangs. Even My Mother Most marriages go sour due to money problems. I Just Entered #SummerBySony So I Could Win Beyonce Tickets:) Wow Beyonce's song diva just came on, i love it so much If Jay-Z cheat on Beyonce no guy will ever feel the need to be faithful in a relationship ever again. some serious Beyonce sing along going on right now #listen :microphone: :musical_notes: :musical_notes: :woman_dancing: :woman_dancing: Glad to see the most powerful couple in the country is there.... Clearly Beyonce and Jay-Z. Has anyone seen the Feruary GQ cover with Beyonce?

Summary of `Beyonce` Cluster 0 Tweets:
Beyonce flow right now :microphone: :headphone: :musical_score: \nThat's always though . # Lik a big bitty black diva like Beyonce like Trina I think Beyonce could've done better so not feeling her performance. On GQ naked last week, singing at the Inauguration this week Is it just me or did Beyonce sound like some ish on this New Year's Eve show. # Even My Mother u are the greatest Beyonce do a Texas shout out The first was being born as someone who would not grow up to be Batman, Beyonce, or Kobe. Hate mayweather .. Beyonce has an amazing body, and she had a baby a while back :smiling_face_with_heart-eyes: :loudly_crying_face: :red_heart: I can't listen to Beyonce without wanting to sing and dance for everyone around:speak-no-evil_monkey: Back in the day I got a lot of compliments when I had highlights ;) \nBut idk cause I'm really feeling Beyonce's hair tone Look, if you still don't understand... Let me put it to you this way... \n\nBeyonce could wake me up from a nap and I would still be pissed.

Summary of `Beyonce` Cluster 1 Tweets:
I wish everybody get off Beyonce dick stop like y'all don't know she can sing #thanks #Managment:face_with_steam_from_nose: Wow Beyonce! Too great of a singer to fake it but I get it LOL Yall need to get off of Beyonce nuts.. Actin' like she the first celeb to ever lip sync! Lol Ok I know I'm gonna get stoned for this but I just heard Beyonce star spangled banner and it was actually average to me ... did she (Kenya) just say Everyday people thinks shes Beyonce sumbody get her meds #sheforgotagain The same hoes talkin shit bout Beyonce was the same ones that was on that Barbie swag when Nicki got hot.... FOH man lol Most marriages go sour due to money problems. Jay-Z dont have no issues on Money with Beyonce. Ladies take note lol as bold as Kenya thinking she looks like Beyonce!!!!

Figure 14: Extractive summaries generated by BERT-Ext for various sets of tweets from the `Beyonce` topic of DivSumm. The first summary represents all 90 of the tweets, the second summary of those tweets clustered in cluster 0, and the third summary of those tweets clustered in cluster 1. Each summary is five "sentences" in length as determined by BERT-Ext.

Summary of the Summaries of `Beyonce` Cluster 0 and Cluster 1:
Beyonce flow right now :microphone: :headphone: :musical_score: \nThat's always though . # Lik a big bitty black diva like Beyonce like Trina I think Beyonce could've done better so not feeling her performance. Even My Mother u are the greatest Beyonce do a Texas shout out The first was being born as someone who would not grow up to be Batman, Beyonce, or Kobe. Too great of a singer to fake it but I get it LOL Yall need to get off of Beyonce nuts.. Actin' like she the first celeb to ever lip sync! Lol Ok I know I'm gonna get stoned for this but I just heard Beyonce star spangled banner and it was actually average to me ... did she (Kenya) just say Everyday people thinks shes Beyonce sumbody get her meds #sheforgotagain The same hoes talkin shit bout Beyonce was the same ones that was on that Barbie swag when Nicki got hot.... FOH man lol Most marriages go sour due to money problems.

Figure 15: Extractive summary generated by BERT-Ext from the extractive summaries for cluster 0 and cluster 1 of the `Beyonce` topic of DivSumm. The two extractive summaries of cluster 0 and cluster 1 were combined into a one contiguous document that was then input into BERT-Ext which output a summary of five "sentences" in length.

To evaluate the quality of all the summaries, we calculate ROUGE scores, specifically, the ROUGE-1 and ROUGE-L scores by comparing each machine-generated summary with two reference gold extractive summaries from DivSumm. The ROUGE-2 scores often contained values of 0, and did not seem to present much insight beyond insights already derived from ROUGE-1. The average ROUGE scores are presented in Table 6.

		F-Score	
Topic	Approach	ROUGE-1	ROUGE-L
Beyonce	all tweets	27.23	23.04
	cluster 0	**28.98**	**27.42**
	cluster 1	25.37	21.16
	combo	27.39	24.16
Christmas	all tweets	23.99	22.15
	cluster 0	23.56	20.20
	cluster 1	25.99	24.72
	combo	**30.22**	**28.85**
NBA	all tweets	**40.65**	**40.65**
	cluster 0	25.93	24.25
	cluster 1	20.92	19.33
	combo	25.32	24.11
Netflix	all tweets	30.09	26.01
	cluster 0	36.60	31.72
	cluster 1	**38.08**	**34.51**
	combo	33.11	29.48
Obama	all tweets	21.72	20.44
	cluster 0	**25.17**	**24.54**
	cluster 1	13.52	12.17
	combo	13.76	12.34

Table 6: Average ROUGE-1 and ROUGE-L scores for comparing both extractive gold summaries from DivSumm with the summaries generated by BERT-Ext for the following sets of tweets per topic: all tweets, cluster 0 tweets, cluster 1 tweets, and the summary of the cluster summaries combined. For each topic, the italicized cluster name is the cluster containing more tweets, and the bolded cluster name is the cluster with a higher intra-similarity score. The bolded and purple F-scores indicate the highest scores for that topic.

In studying the average ROUGE scores, the first thing we immediately noticed is that for four out of five topics, the cluster-based summaries scored higher than the summaries generated from the entire set of tweets, i.e., all tweets. On average, when a cluster approach outperformed the all tweets approach, it did so by an average of 4.86 points with respect to ROUGE-1. The topic that does not adhere to this pattern is NBA, where the all tweets approach performed significantly better than any of the clustered approaches. We hypothesize that this may be due to the domain of the NBA topic

containing more niche language than the others, and having the full set of tweets possibly provides greater context in extracting tweets.

This was an exciting finding for us, as for some reason our *k*=2 cluster approaches provide improvement in four of the five topics of DivSumm to the accuracy of summaries generated in comparison to summaries from the all tweets approach. We further corroborated these results by performing the same analysis using summaries generated with an off the shelf generic summarizer, which can be found in Appendix B. We continue exploring this further and attempt to look into a reason for this interesting trend in the next section.

Recap: We generated automatic extractive summaries with and without clustering and found that the summaries generated using one of the cluster approaches outperformed the summaries generated without clustering.

3.5 Exploration 4: Investigating why one cluster results in significantly better summaries

Having discovered the performance gain from summarizing one of the two clusters in the prior section, we decided to analyze the *k*=2 clustering more thoroughly to see if we could determine a cause.

3.5.1 Cohesion, size, and ROUGE analysis of clusters

First, we decided to compute cluster cohesion guided by our intuition that a more cohesive cluster might generate a better summary. The cohesion of a cluster is a measure of how similar the data points within the cluster are to one another (Jin, 2008). In this case, the pairwise similarity scores act as a measure of the cohesiveness of the top words of the cluster with one another, in turn demonstrating how cohesive the cluster is overall. This method of determining cohesion aligns with the proximity graph-based approach for determining cluster cohesion (Jin, 2008).

In examining the cohesion of the clusters, we first had to determine the top words for each cluster. We accomplished this by utilizing the tf-idf scores for each word used in the clustering process. More specifically, the top words were determined using the following steps: first, we grouped the tweets into their corresponding clusters. We then took the tf-idf vectors for each tweet and calculated an average vector to represent each cluster. Using the averages, we pulled out the words with the highest scores and attributed those words as the top words of the cluster. For example, if tweets 2, 32, 54, and 78 out of the set of 90 tweets were clustered together in cluster 1, we isolated the tf-idf vectors associated with those tweets. Then, we averaged all of those vectors together to compute the mean vector. Looking at the mean vector, we organized each dimension (ie: each word) in descending score order and chose the top thirteen scoring features to act as our top words for that cluster and generated plots like the ones shown in Figure 16.

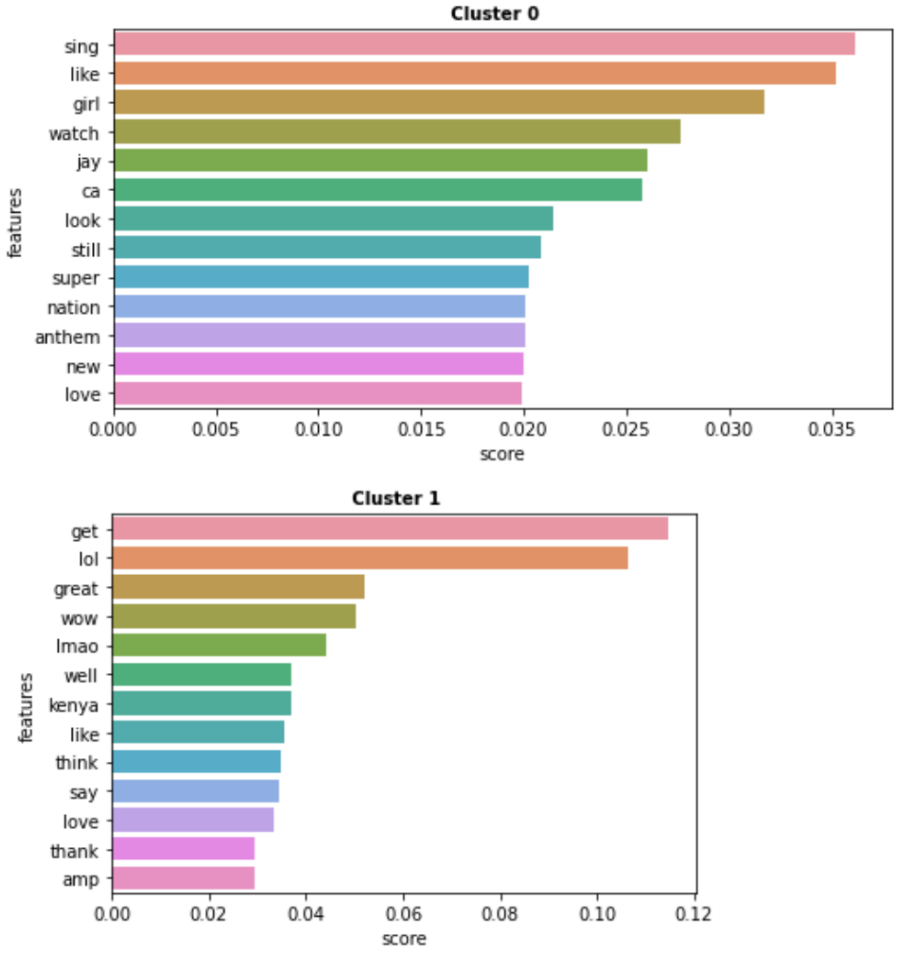

Figure 16: Plots of top words for the *k*=2 clusters for the `Beyonce` set of tweets using average tf-idf. The x-axis represents the average tf-idf score of that word across all tweets within that cluster. The y-axis represents the stemmed representations of the top words themselves (stemming was used to calculate the tf-idf scores to combine similar words in different forms within the tweets, ie: 'amazing' and 'amazed' would both be stemmed to 'amaz'). The top words were determined by calculating the tf-idf values for each token in each tweet, where each tweet was represented as a tf-idf vector of dimension length equal to the total vocabulary of the tweet set. Then, these tweet vectors were separated per cluster and averaged to yield a final vector of averaged tf-idf values per cluster as a representative of that cluster. Finally, the average tf-idf scores per cluster were sorted in descending order and the top thirteen scoring words per cluster were plotted above.

The top words depicted in these plots represent stemmed versions of the actual words present in the `Beyonce` set of tweets in DivSumm. Stemming was used to ensure different versions of the same words were combined to yield one tf-idf score that

accurately represents all versions of that word, as opposed to diluting the scores by separating different versions of the same words out. We also generated word clouds of these top words for a visual exploration of each cluster, as shown in Figure 17.

Figure 17: Word clouds generated for the *k*=2 clusters for the `Beyonce` set of tweets using average tf-idf scores of the words. These scores were calculated in the same way as the scores used in generated the top words plots in Figure 16. The larger the word, the higher the average tf-idf score of that word with respect to the other words in the plot.

Having determined the top words per cluster, we next calculated the pairwise cosine similarity of the top words in each cluster. To calculate the vector similarity, we

used spaCy's "en_core_web_lg" pretrained model.[14] We also did some pruning of the top-words during this process based on if they were present in the pretrained model. If a top word was not present in the model, it resulted in a score of zero similarity when compared to other words, artificially lowering the average similarity score of the cluster. Specifically, we removed any textual emojis (ie: red_heart), any duplicate words, and any symbol text (ie: amp, lt, gt) that resulted in scores of zero. However, interestingly we were able to leave some abbreviations of words and words that were misspelled, as they did have vectors associated with them for comparison within the model (ie: "smh" and "sooo" were two that resulted in score values). An example of the calculated pairwise similarity scores is shown in Table 7. We then averaged the pairwise similarity across the entire cluster to determine an overall top words similarity score to represent that cluster, and these values are shown in Table 8 along with the corresponding cluster sizes.

[14] SpaCy's "en_core_web_lg" pretrained model: https://spacy.io/usage/linguistic-features#vectors-similarity

	sing	like	girl	watch	jay	can't	look	still	super	nation	anthem	new	love
sing		0.43	0.39	0.35	0.29	0.42	0.30	0.37	0.24	0.21	0.55	0.19	0.51
like			0.43	0.48	0.25	0.70	0.69	0.66	0.43	0.32	0.22	0.40	0.66
girl				0.37	0.31	0.42	0.43	0.42	0.39	0.21	0.24	0.29	0.56
watch					0.26	0.45	0.45	0.39	0.31	0.21	0.17	0.33	0.45
jay						0.20	0.18	0.18	0.27	0.18	0.32	0.14	0.29
can't							0.62	0.76	0.38	0.32	0.19	0.35	0.58
look								0.63	0.46	0.29	0.11	0.49	0.56
still									0.37	0.42	0.17	0.42	0.53
super										0.17	0.17	0.39	0.40
nation											0.36	0.32	0.28
anthem												0.19	0.27
new													0.43
love													
Average:		0.366											

Table 7: Pairwise cosine similarity scores for the top words of the `Beyonce` *k*=2 cluster 0. Since the comparisons were pairwise, the matrix is mirrored across the diagonal, and for this analysis we only utilized one unique portion of the matrix and we omitted the similarity scores along the diagonal (which would all be 1.0 since that represents comparing a word to itself). The average of all of the values is shown at the bottom of the table. (For brevity and table size constraints, the similarity scores are shown to the second decimal, but for averaging higher granularity values were used.)

Topic		Cluster 0	Cluster 1
Beyonce	Cohesion (top words)	0.366	**0.417**
	Size	**70**	20
	ROUGE-1	**28.98**	25.37
Christmas	Cohesion (top words)	**0.445**	0.222
	Size	**80**	10
	ROUGE-1	23.56	**25.99**
NBA	Cohesion (top words)	0.291	**0.469**
	Size	23	**67**
	ROUGE-1	**25.93**	20.92
Netflix	Cohesion (top words)	0.349	**0.439**
	Size	**52**	38
	ROUGE-1	36.60	**38.08**
Obama	Cohesion (top words)	0.267	**0.303**
	Size	**80**	10
	ROUGE-1	**25.17**	13.52

Table 8: This table presents analysis of the two *k*=2 clusters with regards to similarity, size, and ROUGE-1. The cohesion represents the average pairwise cosine similarity of the top words of the cluster, the size represents the number of tweets contained in the cluster, and ROUGE-1 represents the average ROUGE-1 F-score for the automatic extractive summary generating using that cluster (please refer to Table 6 for more information about how the ROUGE-1 F-scores were obtained).

Upon reviewing the results in Table 8, we did not notice any connection between the cohesion and the size of the clusters. What we found is that for three of the five topics (Beyonce, Netflix, and Obama) the smaller cluster had a higher similarity amongst

the top words, while for the other two topics (Christmas and NBA) the larger cluster's top words were more similar. Furthermore, when averaging the cohesion scores for all of the larger clusters and all of the smaller clusters separately, we found that on average the larger clusters had an average similarity of 0.386 while the smaller clusters had an average similarity of 0.317.

We also examined the average ROUGE-1 F-scores of the clusters with respect to the similarity and size. We were not able to determine any specific correlation here, either. For one of the five topics, the higher ROUGE-1 score aligned with the higher similarity, and for two of the five topics the higher ROUGE-1 score aligned with the larger cluster. From these results, we were unable to notice any distinct correlation between ROUGE-1 and the cohesion and size of the cluster.

As a measure of thoroughness, we also decided to examine the pairwise similarity of the tweets themselves within each cluster using the same method as before. We performed this analysis for the first two topics to see if they mimicked the trend in Table 8.

Topic		Cluster 0	Cluster 1
Beyonce	Cohesion (tweets)	0.795	**0.838**
	Size	**70**	20
	ROUGE-1	**28.98**	25.37
Christmas	Cohesion (tweets)	**0.807**	0.753
	Size	**80**	10
	ROUGE-1	23.56	**25.99**

Table 9: This table presents similar data to Table 8 for each k=2 cluster, with a slight change to how the similarity is determined. In this case, the cohesion represents a pairwise similarity comparison of tweets in each cluster for k=2. The size and ROUGE-1 are determined the same as before.

As shown in Table 9, we found no deviation in trend for these scores in comparison to the top-words trends. `Beyonce`'s smaller cluster had a higher pairwise tweet similarity score and a lower ROUGE-1 score, while `Christmas`' larger cluster had a higher similarity score and a lower ROUGE-1 score. Thus, we concluded that there was no discernable correlation between the cohesion, size or ROUGE-1 scores of the clusters.

Recap: We attempted to investigate why one cluster's summaries yielded significantly superior summaries than the other cluster. Our results indicate little correlation between the cohesion, size, and the ROUGE-1 score of the cluster.

3.5.2 Correlation between human summaries and clusters

Next, we look at how the clusters correlate to the human generated gold extractive summaries of DivSumm. Our idea was that if we could develop a mapping of the human-generated summaries to the clusters, it would help us understand why certain clusters would result in better summaries.

For this analysis, we first determined which cluster each tweet chosen by an annotator belonged to, and Table 10 shows these results. For the k=2 clustering, on average the tweets chosen by our annotators were mostly clustered into the larger cluster. The only exception to this trend was for `Netflix`, where for both summaries three of

the five tweets chosen belong to the smaller cluster. However, the size of the clusters for Netflix only differed by 14 tweets (cluster 0 contained 52 tweets while cluster 1 contained 38 tweets), so this deviation is not surprising considering how close in size they are. Also, for four out of five of the topics both clusters contained tweets from the extractive gold summaries. Obama was the only deviation from this, where no tweets selected by the human annotators were clustered into cluster 1. Of note in this case is that cluster 1 was the smallest cluster size of all of the *k*=2 clusters across all topics (it tied with the smaller cluster of Christmas – both contained ten tweets), so this is not overly surprising.

	Beyonce		Christmas		NBA		Netflix		Obama	
Clustering	Ext. 1	Ext. 2	Ext. 1	Ext. 2	Ext. 1	Ext. 2	Ext. 1	Ext. 2	Ext. 1	Ext. 2
k=2	1	0	1	0	0	1	1	0	0	0
	0	0	1	0	1	1	0	1	0	0
	0	1	0	0	0	1	0	1	0	0
	0	0	0	1	1	0	1	1	0	0
	0	0	0	0	1	1	1	0	0	0
k=6	2	3	2	3	2	0	1	2	5	4
	5	2	2	3	3	1	2	1	0	4
	4	2	0	4	3	1	2	4	3	0
	2	2	4	2	2	4	4	2	5	4
	2	2	4	1	1	2	4	2	0	5

Table 10: Displays which cluster each of the tweets composing the two gold extractive summaries in DivSumm belong to with respect to the *k*=2 and *k*=6 clusterings. Each extractive summary consists of five tweets chosen by the human annotators, where Ext. 1 and Ext. 2 refer to each of the two extractive, human-generated summaries.

For the *k*=6 clustering, the extractive summaries were each split between three to four clusters, with seven of the summaries containing tweets in three of the clusters and the remaining three containing tweets in four clusters. Interestingly, there isn't a single

extractive summary containing tweets from five different clusters. However, this isn't too surprising considering the spread of tweets per cluster isn't uniform (ie: with 90 tweets a uniform distribution would be roughly 15 tweets per cluster when k=6). Instead, we found that most of the clusters fell below fifteen tweets with one or two clusters containing a much greater number of tweets (please refer to Appendix C for the cluster sizes). `Beyonce`, `Christmas`, `NBA`, and `Obama` all had one cluster that contained many more tweets than the others, while `Netflix` contained two clusters that contained an above-average number of tweets.

While there is no discernible pattern between the two extractive summaries per topic and where their corresponding tweets were clustered, of interest is the fact that for each topic one or two of the clusters contained no tweets present in either extractive summary. That is, none of the topics present a case where every cluster is represented amongst the ten tweets selected by the annotators. Furthermore, for `Beyonce`, `Netflix`, and `Obama` two of the clusters are absent in the extractive summaries. The absence of clusters in the summaries may pertain to a hypothesis mentioned earlier that human annotators may gravitate away from certain characteristics of tweets when generating their extractive summaries. We explore this phenomenon in the following subsection.

3.5.3 Why were certain clusters omitted from the extractive summaries?

Our next step was to attempt to explain why certain clusters were omitted from the extractive gold summaries, while others contained tweets selected by annotators with

the intuition that explaining this may provide a means for determining which clusters to use in summary generation to more closely match human-generated summaries. In other words, our idea behind analyzing this trend is that if we can retrofit the human-generated summaries to our clusters, and identify heuristics for this fitting, we could then reverse the logic to select the clusters that would yield better summaries.

Analyses of our clusters were performed with regards to cluster size, sentiment, tweet length, and semantic similarity. For the next portion, the clusters containing tweets not present in the extractive summaries will be referred to as omitted clusters, and clusters with tweet(s) present in the extractive summaries will be referred to as selected clusters.

	Size		Sentiment		Length	Cohesion	
Cluster	# of Tweets	% of All Tweets	Positive/Neutral % Composition	Negative % Composition	Token Count	Tweet Similarity	Top Words Similarity
Omitted	10.11	11.23%	66.48%	**33.52%**	16.08	**0.8401**	0.3172
Selected	**17.10**	**19.00%**	**72.93%**	27.07%	**16.21**	0.8323	**0.3916**

Table 11: Average values for *k*=6 clusters containing tweets that were both omitted and selected for human annotated extractive summaries. Averages were calculated for clusters from all topics of DivSumm.

Cluster size was examined by considering the number of tweets within each cluster. On average, the clusters that were omitted from the extractive summaries contained approximately 10.1 tweets, while the clusters selected on average contained approximately 17.1 tweets. Furthermore, the omitted clusters on average were each composed of 11.2% of all tweets being clustered, while the selected clusters consisted of 19.0% of the tweets on average. This indicates that on average the omitted clusters were

around 40% smaller than the selected cluster. Furthermore, for all five topics the largest cluster was never omitted, while the smallest cluster was omitted for two topics.

The sentiment of the clusters was analyzed by determining whether each tweet contained within the cluster was positive/neutral in sentiment or negative. We observe that, on average, 33.5% of tweets within the omitted clusters were negative in nature, while 27.1% of tweets within the selected clusters were negative. Furthermore, the cluster for each topic with the highest percent composition of negative tweets was omitted in four of the five topics, and in the one instance where this cluster was selected the overall negativity of tweets within the topic was lower than average.

The average token length of tweets for each cluster was also analyzed to determine if there were any discernable patterns. Individual tokens were determined by the location of spaces on the tweet, where each token was flanked on either side by a space. For omitted and selected clusters the average tweet length was approximately 16.1 and 16.2 tokens respectively, and the average range among topics between the highest average tweet length cluster and lowest average tweet length cluster was 4.8 tokens. Furthermore, when looking at this metric at a finer granularity there were no obvious trends. The cluster with the longest average tweet length was omitted for two out of five topics, and the cluster with the shortest average tweet length was omitted for one out of five topics. Overall, the lack of pattern in this metric is not surprising considering the minimum tweet length required by DivSumm for inclusion combined with Twitter's character restrictions.

The cohesion of the clusters, or the average similarity, was also considered to help discern why the tweet contents of certain clusters was more attractive to annotators. This analysis was performed in the same way the similarity analysis was performed in section 3.5.1. It was determined that the average tweet pairwise similarity for selected clusters was 0.83, while the average similarity for omitted clusters was 0.84. This indicates that the tweets within the omitted clusters were ever so slightly more similar, but likely not to a discernable degree. Furthermore, the average similarity of top words for the selected clusters was 0.39, while it was 0.32 for the omitted clusters. This indicates that the top words of the selected clusters were more cohesive than the omitted clusters, and this may have been an attractive trait to the human annotators. It also makes sense, as those clusters were likely more cohesive in their intra-cluster topics, indicating they contained more repeated ideas.

Overall, of the metrics analyzed the three that stood out the most were cluster size, sentiment and cluster cohesion. Specifically, the clusters selected by annotators were larger in the number of tweets they contained, were composed of less negative tweets, and had a higher intra-cluster cohesion. These results are a promising start in using clustering to yield higher quality automatic text summaries.

Recap: We performed an analysis of the clusters with regards to size, sentiment, tweet length, and cluster cohesion to try to discern why certain clusters were more appealing to human annotators. We found that of these metrics, cluster size, sentiment and cluster cohesion have the highest correlation to clusters selected by annotators.

4 Conclusion and Future Work

4.1 Discussion

At the start of this work, we identified the importance of diversity in automatic text summarization, and that guided our work in exploring summarization of a dialect-distinct set of social media text. We began by assessing how diverse extractive summaries generated by humans are, and discovered that the gold extractive summaries of DivSumm are diverse. We extended this work to also explore the annotator overlap, and while we noticed a pattern here, we were unable to fully provide any clear explanation for its occurrence.

Next, we examined what would happen if we clustered the dialect-distinct tweets, and found that the tweets were clustered based on topic and not dialect. We analyzed the diversity of the generated clusters and found that overall they were fairly diverse with respect to the dialects. We then generated multi-document automatic extractive summaries of each of the clusters and also one for the set of all of the tweets and one from combining the two cluster summaries, and then compared them all to the human-generated summaries using ROUGE metrics. Through this comparison we discovered that the summary of one of the cluster approaches outperformed the summary of all of the tweets for four out of five topics of DivSumm. With this finding we were able to propose a novel approach for extractive summarization of dialect-diverse text.

Next, we performed various empirical analyses to attempt to explain the phenomenon of one cluster summary performing best. We performed cluster cohesion analysis using pairwise similarity of the top words of each cluster, and then analyzed the

results with respect to the corresponding cluster size and ROUGE-1 metrics to determine if any correlation existed as a way to predict an expected cluster similarity. We were unable to discern any noticeable patterns between the three metrics, and leave further exploration of this phenomenon for future work.

Lastly, we examined the correlation between the human generated summaries and the clusters. We found that in the k=6 clustering, certain clusters were entirely omitted from the human summaries. We felt that if we could find an explanation for this, it may provide us with some heuristics for determining which cluster would perform better in the automatic summarization process. We performed analysis on the selected and omitted clusters with regards to cluster size, sentiment, tweet length, and cluster cohesion. Overall, we were unable to discern any distinct connections, but found that size, sentiment and cohesion are likely factors in determining whether a cluster will be selected or omitted by annotators.

Overall, we feel that we made solid contributions on understanding what clustering dialect-diverse social media text looks like. We were able to determine that the summaries generated by humans and the clustering of the dataset were dialect-diverse, and that performing clustering prior to summarization yields a summary that more closely resembles human extractive summaries with regards to a dialect-diverse set of text.

4.2 Future Work

While this thesis allowed us to draw some conclusions, it also opened up many new questions. We are excited to further examine the ROUGE performance gains demonstrated for the summaries generated for the k=2 clustering compared to the summary generated from the full set of tweets. While the summary generated from one of the clusters scored much higher on average than the summary of all of the tweets, the summary of the other cluster scored much lower. It is important to distinguish how to identify which cluster should be used in summarizing to consistently obtain these improved results. We would like to perform more analysis on the clusters to better understand the differences between them to determine why one yields a considerably higher score with ROUGE than the other. We hypothesize it is likely due to the coherence of tweets/topics within the clusters, but need to perform additional experimentation to examine this.

We would also like to develop a heuristic to understand why certain clusters contained tweets entirely omitted from the extractive gold summaries of DivSumm. While we were able to hone in on some metrics that we feel are correlated to this, we would like to perform additional analysis to develop some tangible results as to what is causing this split of chosen versus omitted clusters. We feel that this is a promising avenue because if we can manage to retrofit our clusters to the human-generated summaries, it could lead to a mapping that could be utilized to generate more human-like summaries from clustered data.

References

Mohsen Abbasi, Aditya Bhaskara, and Suresh Venkatasubramanian. 2021. Fair clustering via equitable group representations. In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, 504–514.

Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. 2001. On the surprising behavior of distance metrics in high dimensional space. In International conference on database theory, Springer, 420–434.

Ameeta Agrawal, 2021, *Lecture 4: Text Classification*, lecture notes, CS 410/510: Natural Language Processing, Portland State University, delivered Fall 2021.

Ayush Agrawal and Utsav Gupta. 2014. Extraction based approach for text summarization using k-means clustering. International Journal of Scientific and Research Publications 4, 11 (2014), 1–4.

Sumya Akter, Aysa Siddika Asa, Md Palash Uddin, Md Delowar Hossain, Shikhor Kumer Roy, and Masud Ibn Afjal. 2017. An extractive text summarization technique for Bengali document (s) using K-means clustering algorithm. In 2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR), IEEE, 1–6.

Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. 2017. Text summarization techniques: a brief survey. arXiv preprint arXiv:1707.02268 (2017).

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of" bias" in nlp. arXiv preprint arXiv:2005.14050 (2020).

Su Lin Blodgett, Lisa Green, and Brendan O'Connor. 2016. Demographic dialectal variation in social media: A case study of African-American English. arXiv preprint arXiv:1608.08868 (2016).

Chris Brinton and David Inouye, 2020, *n-grams and basic natural language processing*, lecture notes, ECE 20875: Python for Data Science, Purdue University, delivered 03/25/2020, https://www.cbrinton.net/ECE20875-2020-Spring/W10/ngrams.pdf.

Guoqing Chao, Shiliang Sun, and Jinbo Bi. 2017. A survey on multi-view clustering. arXiv preprint arXiv:1712.06246 (2017).

Freddy Chua and Sitaram Asur. 2013. Automatic summarization of events from social media. In Proceedings of the International AAAI Conference on Web and Social Media, 81–90.

Iain J Cruickshank and Kathleen M Carley. 2020. Characterizing communities of hashtag usage on twitter during the 2020 COVID-19 pandemic by multi-view clustering. Applied Network Science 5, 1 (2020), 1–40.

Abhisek Dash, Anurag Shandilya, Arindam Biswas, Kripabandhu Ghosh, Saptarshi Ghosh, and Abhijnan Chakraborty. 2019. Summarizing user-generated textual content: Motivation and methods for fairness in algorithmic summaries. Proceedings of the ACM on Human-Computer Interaction 3, CSCW (2019), 1–28.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018).

Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2018. Measuring and mitigating unintended bias in text classification. In Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society, 67–73.

Harold P Edmundson. 1969. New methods in automatic extracting. Journal of the ACM (JACM) 16, 2 (1969), 264–285.

Wafaa S El-Kassas, Cherif R Salama, Ahmed A Rafea, and Hoda K Mohamed. 2021. Automatic text summarization: A comprehensive survey. Expert Systems with Applications 165, (2021), 113679.

Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: a survey. Artificial Intelligence Review 47, 1 (2017), 1–66.

Peter Grabusts. 2011. The choice of metrics for clustering algorithms. In ENVIRONMENT. TECHNOLOGIES. RESOURCES. Proceedings of the International Scientific and Practical Conference, 70–76.

Dirk Hovy and Shrimai Prabhumoye. 2021. Five sources of bias in natural language processing. Language and Linguistics Compass 15, 8 (2021), e12432.

Ben Hutchinson and Margaret Mitchell. 2019. 50 years of test (un) fairness: Lessons for machine learning. In Proceedings of the Conference on Fairness, Accountability, and Transparency, 49–58.

Clayton Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In Proceedings of the international AAAI conference on web and social media, 216–225.

Ruoming Jin, 2008, *Cluster Validation*, lecture notes, CS 63015/73015: Data Mining Techniques, Kent State University, delivered 10/13/2008, http://www.cs.kent.edu/~jin/DM08/ClusterValidation.pdf.

Moniba Keymanesh, Tanya Berger-Wolf, Micha Elsner, and Srinivasan Parthasarathy. 2021. Fairness-aware summarization for justified decision-making. arXiv preprint arXiv:2107.06243 (2021).

Young-Min Kim, Massih-Reza Amini, Cyril Goutte, and Patrick Gallinari. 2010. Multi-view clustering of multilingual documents. In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, 821–822.

Svetlana Kiritchenko and Saif M Mohammad. 2018. Examining gender and race bias in two hundred sentiment analysis systems. arXiv preprint arXiv:1805.04508 (2018).

Athanasia Koumpouri, Iosif Mporas, and Vasileios Megalooikonomou. 2015. Evaluation of Four Approaches for" Sentiment Analysis on Movie Reviews" The Kaggle Competition. In Proceedings of the 16th International Conference on Engineering Applications of Neural Networks (INNS), 1–5.

Emmanouil Krasanakis, Eleftherios Spyromitros-Xioufis, Symeon Papadopoulos, and Yiannis Kompatsiaris. 2018. Adaptive sensitive reweighting to mitigate bias in fairness-aware classification. In Proceedings of the 2018 world wide web conference, 853–862.

Madan Krishnamurthy, Khalid Mahmood, and Pawel Marcinek. 2016. A hybrid statistical and semantic model for identification of mental health and behavioral disorders using social network analysis. In 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), IEEE, 1019–1026.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461 (2019).

Yang Li and Tao Yang. 2018. Word embedding for understanding natural language: a survey. In Guide to big data applications. Springer, 83–104.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Text summarization branches out, 74–81.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. IBM Journal of research and development 2, 2 (1958), 159–165.

David J.C. MacKay. 2003. An example inference task: clustering. In Information theory, inference and learning algorithms. Cambridge University Press, 284-292.

James MacQueen and others. 1967. Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Oakland, CA, USA, 281–297.

Kathleen R McKeown, Regina Barzilay, David Evans, Vasileios Hatzivassiloglou, Judith L Klavans, Ani Nenkova, Carl Sable, Barry Schiffman, and Sergey Sigelman. 2002. Tracking and summarizing news on a daily basis with Columbia's Newsblaster. In Proceedings of the human language technology conference, 280-285.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. ACM Computing Surveys (CSUR) 54, 6 (2021), 1–35.

Derek Miller. 2019. Leveraging BERT for extractive text summarization on lectures. arXiv preprint arXiv:1906.04165 (2019).

Punam Mulak and Nitin Talhar. 2015. Analysis of distance measures using k-nearest neighbor algorithm on kdd dataset. International Journal of Science and Research 4, 7 (2015), 2101–2104.

Jeffrey Nichols, Jalal Mahmud, and Clemens Drews. 2012. Summarizing sporting events using twitter. In Proceedings of the 2012 ACM international conference on Intelligent User Interfaces, 189–198.

Olubusayo Olabisi, Aaron Hudson, Antonie Jetter, and Ameeta Agrawal. 2022. Toward Analyzing and Improving the Dialect Diversity of Summaries, *in preparation*, 2022

MR Prathima and HR Divakar. 2018. Automatic Extractive Text Summarization Using K-Means Clustering. International Journal of Computer Sciences and Engineering (2018).

Alejandro Ramón-Hernández, Alfredo Simón-Cuevas, María Matilde García Lorenzo, Leticia Arco, and Jesús Serrano-Guerrero. 2020. Towards Context-Aware Opinion Summarization for Monitoring Social Impact of News. Information 11, 11 (2020), 535.

Archit Rathore, Sunipa Dev, Jeff M Phillips, Vivek Srikumar, and Bei Wang. 2021. A Visual Tour of Bias Mitigation Techniques for Word Representations. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 4064–4065.

Nouhad Rizk. 2020. Clustering Metrics and Cluster Validity. Building Skills for Data Science (2020). Retrieved February 09, 2022 from https://uhlibraries.pressbooks.pub/buildingskillsfordatascience/chapter/cluster-validity/

Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics 20, (1987), 53–65.

Koustav Rudra, Subham Ghosh, Niloy Ganguly, Pawan Goyal, and Saptarshi Ghosh. 2015. Extracting situational information from microblogs during disaster events: a classification-summarization approach. In Proceedings of the 24th ACM international on conference on information and knowledge management, 583–592.

Mohammad Saleh and Anjuli Kannan. 2022. Auto-generated Summaries in Google Docs. (March 2022). Retrieved May 4, 2022 from https://ai.googleblog.com/2022/03/auto-generated-summaries-in-google-docs.html.

Maarten Sap, Dallas Card, Saadia Gabriel, Yejin Choi, and A Noah Smith. 2019. The risk of racial bias in hate speech detection. In ACL.

Francesco Amedeo Emanuele Talarico and Marco Viviani. 2021. Credible Text Summarization in Social Media. In IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT '21), Association for Computing Machinery, Melbourne, VIC, Australia, 603–610. DOI:https://doi.org/10.1145/3486622.3493978

Appendix A Silhouette analysis plots

Appendix A contains silhouette plots corresponding to the analysis performed in sec. 3.3.1.

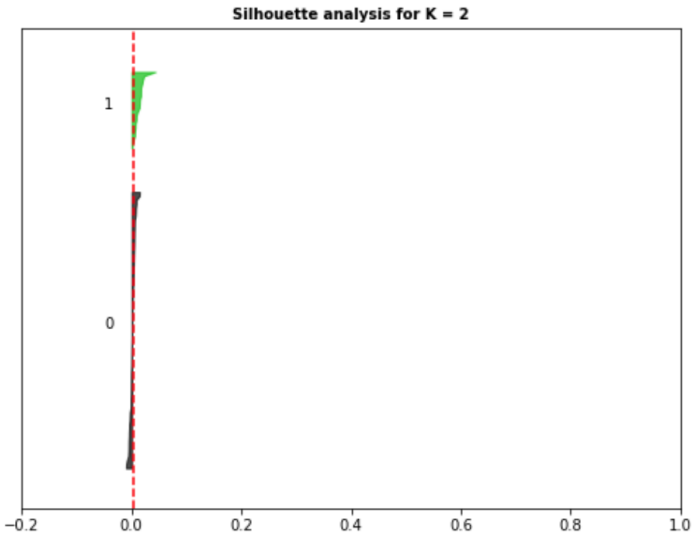

Figure 18: Silhouette analysis for the *k*=2 clustering of the `Beyonce` set of 90 tweets. The vertical spread of each cluster represents how many tweets the cluster contains, while the horizontal spread represents how confidently each tweet was categorized into the cluster (where a value closer to 1.0 indicates a greater confidence).

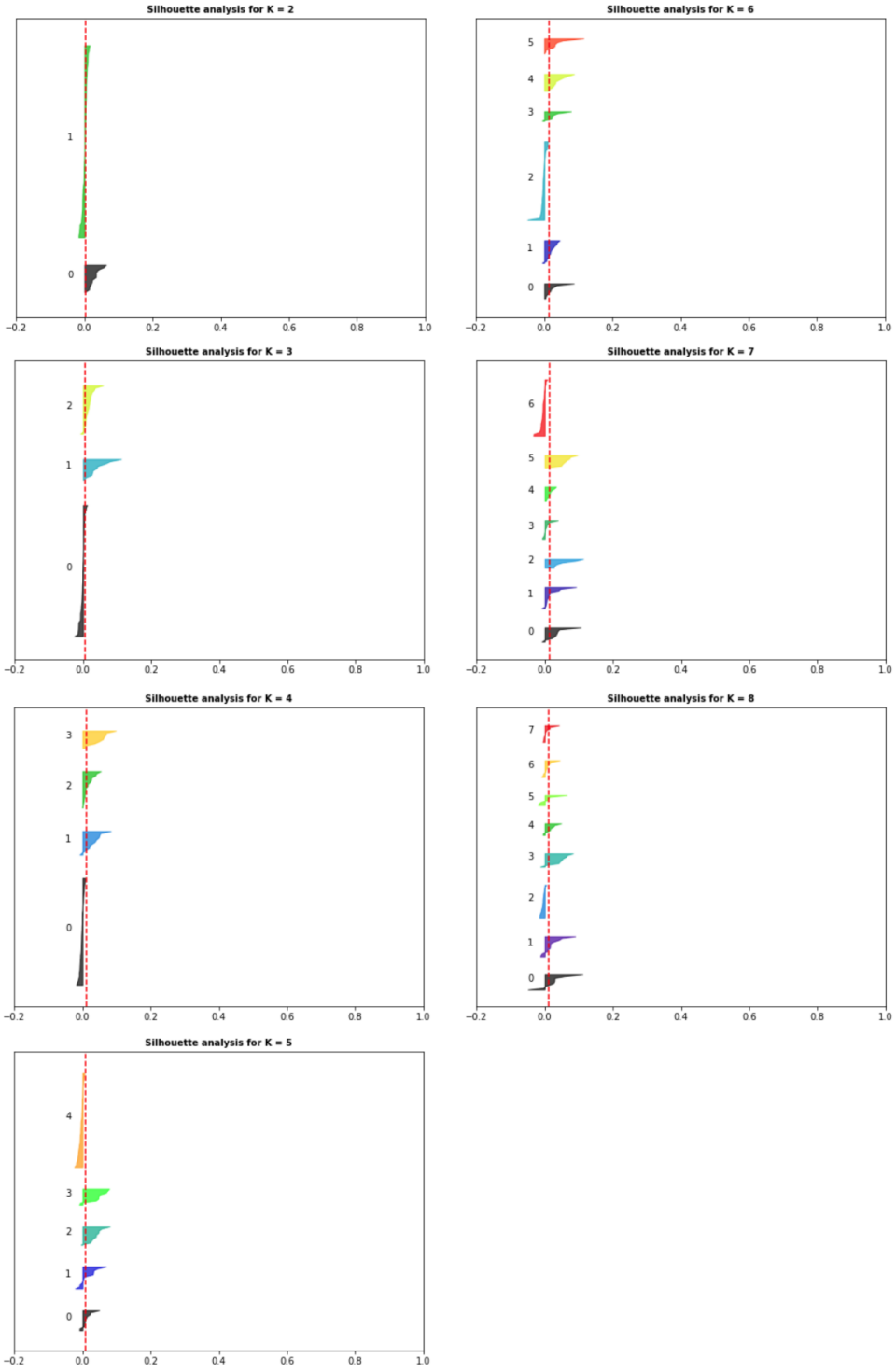

Figure 19: Silhouette analysis for the *k*=6 clustering of the `Beyonce` set of 90 tweets. Please refer to Figure 18 for a more detailed description of silhouette plots.

Appendix B Extractive summaries using a generic summarizer

Appendix B contains extractive summaries generated similarly to those generated in sec. 3.4 However, in this section we used an off the shelf generic summarizer[15] to generate the summaries instead (with the resulting summaries shown in Figure 20 and Figure 21). Additionally, we performed ROUGE analysis of these summaries using the same methods as in sec. 3.4.

When looking at the average ROUGE scores in Table 12, we noticed similar trends to the summaries generated by BERT-Ext in that for three out of five topics one of the summaries generated using a cluster approach scored higher than the summary generated from the all tweets approach. These results help corroborate the performance trends of the cluster versus all tweets approaches we noticed with our BERT-Ext summaries.

[15] https://pypi.org/project/summarizer/

Summary of `Beyonce` All Tweets:
Yall need to get off of Beyonce nuts.. Actin' like she the first celeb to ever lip sync!
And by that I mean where am I eating wings drinking and Watching the Beyonce performance...
Me nd ma thug gone be like Bonnie&Clyde ima be his ride or die his fucking Beyonce to Jay Z
I could be in the worst mood and hear love on top by Beyonce and be like :grinning_face_with_smiling_eyes: :grinning_face_with_smiling_eyes: :grinning_face_with_smiling_eyes: :grinning_face_with_smiling_eyes:
I don't care if Beyonce lip sang, I love her !! Hell I don't care if jay-z did it, it sounded beautiful!!

Summary of `Beyonce` Cluster 0 Tweets:
Beyonce my wife, Gabrielle union my baby, Zoe Saldana my mistress, Janelle monae is who I go to wen they act up
And by that I mean where am I eating wings drinking and Watching the Beyonce performance...
Me nd ma thug gone be like Bonnie&Clyde ima be his ride or die his fucking Beyonce to Jay Z
Destiny's Child releasing new music? Beyonce must need a babysitter while on tour. Who makes a better nanny Kelly or Michelle?
I don't care if Beyonce lip sang, I love her !! Hell I don't care if jay-z did it, it sounded beautiful!!

Summary of `Beyonce` Cluster 1 Tweets:
Wow Beyonce! Unbelievable! Too great of a singer to fake it but I get it LOL
Yall need to get off of Beyonce nuts.. Actin' like she the first celeb to ever lip sync!
Ritz crackers are mad lightskin now. Beyonce probably got something to do with it. Lol
Ok I know I'm gonna get stoned for this but I just heard Beyonce star spangled banner and it was actually average to me ...
Most marriages go sour due to money problems. Jay-Z dont have no issues on Money with Beyonce. Ladies take note lol

Figure 20: Extractive summaries generated by an off the shelf generic summarizer for various sets of tweets from the `Beyonce` topic of DivSumm. The first summary represents all 90 of the tweets, the second summary of those tweets clustered in cluster 0, and the third summary of those tweets clustered in cluster 1. Each summary is five tweets in length.

Summary of the Summaries of `Beyonce` Cluster 0 and Cluster 1:
And by that I mean where am I eating wings drinking and Watching the Beyonce performance...
Me nd ma thug gone be like Bonnie&Clyde ima be his ride or die his fucking Beyonce to Jay Z
Destiny's Child releasing new music? Beyonce must need a babysitter while on tour. Who makes a better nanny Kelly or Michelle?
I don't care if Beyonce lip sang, I love her !! Hell I don't care if jay-z did it, it sounded beautiful!!
Most marriages go sour due to money problems. Jay-Z dont have no issues on Money with Beyonce. Ladies take note lol

Figure 21: Extractive summary generated by an off the shelf generic summarizer from the extractive summaries for cluster 0 and cluster 1 of the `Beyonce` topic of DivSumm. The two extractive summaries of cluster 0 and cluster 1 were combined into a one set of ten tweets that was then input into the summarizer to generate a summary of five tweets in length.

		F-Score	
Topic	Approach	ROUGE-1	ROUGE-L
Beyonce	all tweets	16.48	15.66
	cluster 0	18.24	15.43
	cluster 1	**30.56**	**28.33**
	combo	29	26.16
Christmas	all tweets	11.64	10.91
	cluster 0	12.36	11.58
	cluster 1	**30.86**	**28.55**
	combo	15.38	13.18
NBA	all tweets	**16.9**	**14.14**
	cluster 0	14.08	13.43
	cluster 1	16.44	13.52
	combo	14.34	13.71
Netflix	all tweets	**39.64**	**39.64**
	cluster 0	22.17	18.6
	cluster 1	35.4	34.51
	combo	22.17	18.6
Obama	all tweets	20.71	20.04
	cluster 0	20.71	20.04
	cluster 1	16.67	14.47
	combo	**25.15**	**23.78**

Table 12: Average ROUGE-1 and ROUGE-L scores for comparing both extractive gold summaries from DivSumm with the summaries generated by the generic off the shelf summarizer for the following sets of tweets per topic: all tweets, cluster 0 tweets, cluster 1 tweets, and the summary of the cluster summaries combined. For each topic, the italicized cluster name is the cluster containing more tweets, and the bolded cluster name is the cluster with a higher intra-similarity score. The bolded and purple F-scores indicate the highest scores for that topic.

Appendix C Cluster sizes

Appendix C shows the cluster sizes for the *k*=6 clustering relevant to analysis performed in sec. 3.5.2.

k	**Cluster**	Beyonce	Christmas	NBA	Netflix	Obama
k=6	0	9	33	10	8	32
	1	13	12	29	4	13
	2	43	12	8	35	10
	3	6	11	12	11	10
	4	10	15	17	26	13
	5	9	7	14	6	12

Table 13: Displays the number of tweets per cluster after performing *k*-means clustering with *k*=6 on each set of ninety tweets per topic.

Appendix D Implementation

This section outlines the environment and libraries used to perform the work and analysis for this thesis.

Google Colaboratory

Google Colaboratory,[16] or Google Colab for short, is a free-to-use environment that hosts Jupyter notebooks that can execute python code. There are resource limits associated with free use, and resources are never guaranteed, but it is possible to obtain greater resources by paying a fee. For this project, the free amount of resources was viable, and there was never an issue with resources being unavailable while performing necessary work. DivSumm is not a very large dataset at the time of writing, and as such working with it did not require a large amount of compute power. This made Google Colab a prime candidate for performing a bulk of the work required for this research. A majority of the analysis performed was accomplished in a Google Colab Jupyter notebook, calling the various libraries as needed. Furthermore, the resulting analysis was either downloaded as CSV files, or in the case of charts saved as images into a spreadsheet for further exploration. Additional calculations were performed within these spreadsheets.

[16] Google Colaboratory: https://research.google.com/colaboratory/faq.html#:~:text=Colaboratory%2C%20or%20%E2%80%9CColab%E2%80%9D%20for,learning%2C%20data%20analysis%20and%20education.

NLTK

NLTK (Natural Language Toolkit)[17] is a collection of libraries that provide many utilities for working with text corpora, from classification, tokenization, parsing, stemming, semantic reasoning, tagging, etc. It also provides packages for downloading, like a collection of stopwords that was used within this work. Specifically, within this work NTLK was pivotal in preprocessing the tweets for clustering purposes, allowing easy tokenization, stemming, and removal of stopwords. It was also used for performing semantic analysis.

Pandas

Pandas[18] is a tool available in Python that allows for data analysis and manipulation at a large scale. It introduces the DataFrame as a mechanism for storing data, and allows working with that data efficiently for very large quantities of data. Furthermore, it also allows importing data from CSV files efficiently, and also exporting to CSV files. Pandas was used mainly to read in the tweet data, isolate the desired columns for further analysis, and then export results to CSV files for outside analysis.

NumPy

Numpy[19] is a powerful package for doing mathematical work within Python. A primary function is its ability to store and manipulate data in matrix form. For this project, it was

[17] NLTK: https://www.nltk.org/
[18] Pandas: https://pandas.pydata.org/
[19] Numpy: https://numpy.org/

used for various functions for working with numbers, such as finding minimum values, means, sorting, etc. The library can be imported directly into Google Colab via an "import numpy" command.

Scikit-learn

Scikit-learn[20] is an open-source package that provides access to a bevy of machine learning tools in Python. It is built ontop of NumPy, SciPy, and matplotlib, and allows directly calling many machine learning algorithms. For this work, we utilized the following components from the library: cluster, feature_extraction.text.TfidfVectorizer, preprossing.normalize, metrics.silhouette_samples, and metrics.silhouette_score. Specifically, this Scikit-learn was used to cluster tweets via k-Means (cluster) utilizing tf-idf (TfidfVectorizer), and then calculate Silhouette scores for the clustering results (silhouette_score and silhouette_samples). The library can be imported directly into Google Colab via an "import sklearn" command.

Matplotlib and Seaborn

Matplotlib[21] is a data visualization tool available within Python, and Seaborn[22] is a data visualization tool built ontop of Matplot lib. Both are used to develop plots of data, and can be imported via "import matplotlib" and "import seaborn" commands. The specific components of Matplotlib used within this work were pyplot and cm for developing

[20] Scikit-learn: https://scikit-learn.org/stable/index.html#
[21] Matplotlib: https://matplotlib.org/
[22] Seaborn: https://seaborn.pydata.org/

various plots of metrics related to the various clusterings we explored, specifically including silhouette score plots and top words per cluster plots. Seaborn was used to enhance the top words per cluster plots.

Wordcloud

Wordcloud[23] is a lightweight library for creating word cloud plots within Python. It was used to develop word clouds for the various clusters explored within this work. It can be imported directly into Google Colab via an "import wordcloud" command.

SciPy

SciPy[24] is a library that contains various algorithms for performing scientific calculations within Python. For this work, SciPy's spatial.distance.cdist component was utilized for calculating the distances between clusters. It can be accessed within Google Colab via an "import scipy" command.

SpaCy

SpaCy[25] is an package designed to provide an array of popular NLP tools with support for 64+ languages. It contains transformers, pretrained word vectors, and an array of tools for performing tokenization, part-of-speech tagging, text classification, and other similar NLP textual processing techniques. For this work, SpaCy was used for

[23] Wordcloud: https://pypi.org/project/wordcloud/
[24] SciPy: https://scipy.org/
[25] SpaCy: https://spacy.io/

calculating cosine similarity between words and tweets. For use in Google Colab, SpaCy must first be installed via a pip command, and also a pretrained model must be installed. Then, both SpaCy and the chosen model can be imported into the notebook and used similarly to other packages.