Personalized Extractive Text Summaries

Text Summarization Working Group:

Blake Centini, John M. Conroy, Priscilla C., Nick Gawron, James Hardaway,

Matt K., Joseph Kepler, Cas Laskowski, Neil Molino, Liz Richerson,

Josh Sheinberg, Josh T., Sophie Trotto, and Julia Y.

Summer Conference on Applied Data Science

NC State University Laboratory for Analytic Sciences

August 4, 2022

Table of Contents

Personalized Extractive Text Summaries

Abstract

Analysts within the US Intelligence Community spend an inordinate amount of time locating, reading, and understanding text documents to create insightful analytic products. Any process or tool refinement that facilitates a more efficient and potentially accurate analysis has the potential to save time and information management resources throughout the network of agencies dedicated to this work. In the inaugural Summer Conference on Applied Data Science, the Text Summarization Working Group explored various `occams` model parameters to produce extractive summaries of text and audio file transcripts. The result was a system that imports multiple text documents, processes them through an automated machine learning workflow, and produces summaries influenced by user-generated topics. The goal of the project was to represent research and production processes specific to the Intelligence Community. This work is designed to develop insights and recommendations for future tool development that would increase analyst efficiency in their daily workflow. The research team experimented with text transcripts generated by historians, journalists, and scientific researchers. Texts originated as audio files (Nixon tapes), news articles (MSN, CNN, Daily Mail), and technical research papers. While some documents were accompanied by human-generated summaries or abstracts, others remained in their original form without any processed summarizations. Multiple exploratory data analysis methods were employed to better understand the potential impacts of the types of data used in the experiment. From statistical inference to topic modeling, data biases were identified in an effort to understand the types of data structures best suited for these models. Machine summaries were built using the `occams` software suite, testing various parameters to adjust how the model weighted specific ideas in the reference texts. The results were evaluated by either comparing them to human-generated summaries of the original text or through a question answering model that queried the resultant text. From this work, the team gained valuable insights into how to structure reference and background data samples, represent analyst interests through parameter adjustment, and measure the results to gauge model efficiency.

1. Introduction and Background

1.1 The SCADS Grand Challenge

The **Summer Conference on Applied Data Science** (SCADS) is an annual 8-week workshop hosted by the NC State University Laboratory for Analytic Sciences (LAS). The overarching vision is to bring together industry, academic, and government professionals to collaboratively attack a Grand Challenge in the machine learning (ML) and artificial intelligence (AI) development space. The Grand Challenge (described below) is a somewhat lofty, 5 to 10-year research and development goal that can greatly influence knowledge workers' analysis practices in all sectors. Stepping-stone problems on the way to achieving the Grand Challenge offer near-term value potential.

For this inaugural conference, the Grand Challenge framework is proposed as a method to develop ML capabilities that proactively provide individuals (or organizations) with information relevant to their needs. The objective of a tailored daily report (TLDR) is a relatively short report, auto-generated, perhaps on-demand or on a schedule. This report is to be filled with new information of high interest to the user, drawing simultaneously from any number of sources and weighing the value of source materials relative to the user's objectives and interests. The US Intelligence Community, our target audience, manually produces a daily report of this nature for senior government decision-makers, summarizing new updates of intelligence value. Conceptually, natural language processing (NLP) technology, in the form of specific ML models, could produce something similar for a host of alternative use cases.

During the first week of the conference, LAS leadership facilitated onboarding sessions that reinforced the primary goals driving the SCADS program: grand challenge progress, partnerships, and learning. The program is comprised of participants with diverse experiences, skill sets, and research interests. While we are all encouraged to pursue research that interests us personally, the overarching grand challenge should be kept in the foreground of our work. The conference was purposefully organized to maximize knowledge transfer across academia, industry, and government boundaries. The DoD recognizes the efficiencies to solve these complex challenges at scale lie in partnering with like-minded organizations working on similar problems. Lastly, as SCADS is hosted by a university research laboratory, hands-on learning is critical to expanding a base of knowledge that can be passed from this conference to the next.

1.2 SCADS Working Group Organization

During the conference's second week, participants self-organized into four working groups, each focused on a contributing aspect to the text summarization grand challenge. The working groups were comprised of the following:

- Human-Computer Interaction (HCI),
- Knowledge Graphs (KG),

- Recommender Systems, and
- Text Summarization (TS)

Figure 1 describes working group contributions to the process of text summarization. The **HCI** team focused on how knowledge workers engaged with information through data tools designed to capture analysts' key interests and focus areas. A **knowledge graph** was chosen as a method to organize the large amounts of data and information that IC analysts could access to understand and update their daily work tasks. **Recommender systems** query the knowledge graph to determine analyst preferences and return documents whose summaries may satisfy their queries. The **text summarization** team developed methods to process the recommended documents to deliver a concise summary that would answer specific analyst questions and/or add to the knowledge graph through a feedback loop. At multiple stages throughout the process, results can be refined through additional recommender or topic modeling layers.

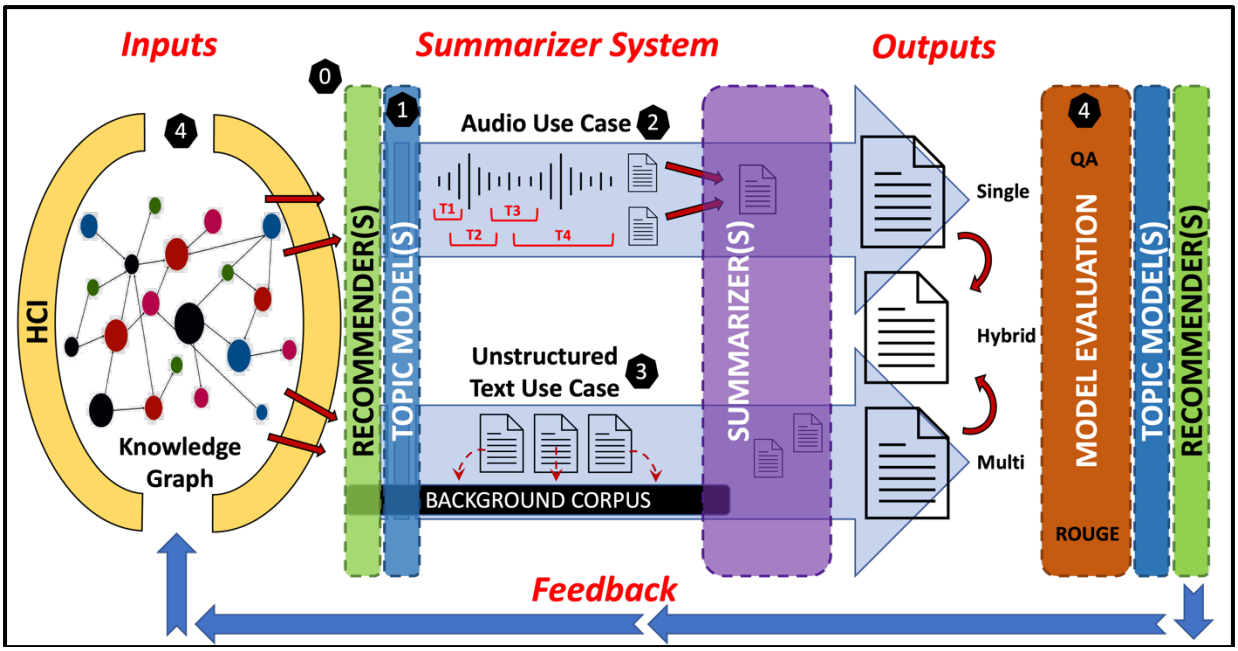

Figure 1 - Summarization Process Overview

For this first SCADS iteration, the TS group prioritized three use cases for experimentation: one for audio files and two for unstructured text documents. Those use cases are highlighted by the numbered polygons (#2 and #3) in Figure 1. The additional polygons represent work group efforts that either enabled the primary use cases (#0 and #1) or built on their results to deliver feedback during the output phase of the process (#4). Those specific efforts are detailed in Figure 2:

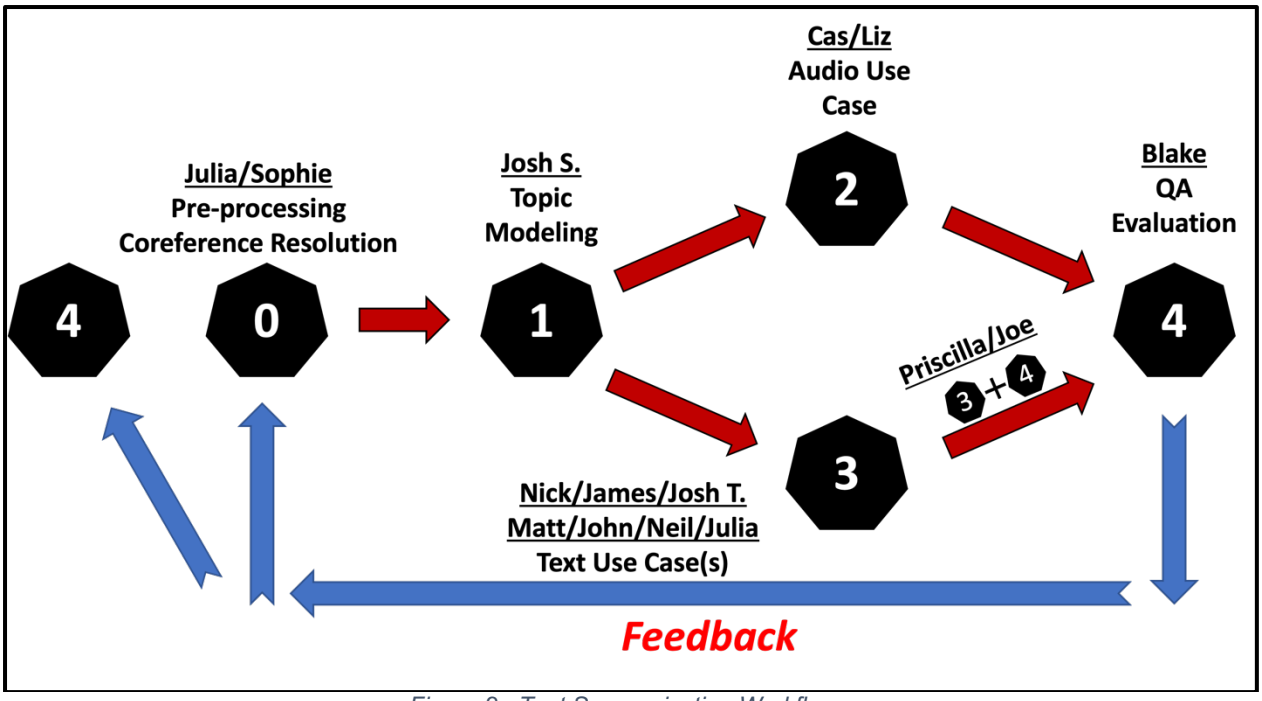

Figure 2 - Text Summarization Workflow

Figure 2 depicts how the text summarization group allocated team member resources to the multiple facets of the challenge. Beginning at #0, participants worked on pre-processing data we would normally obtain via the knowledge graph and recommender groups as input for the summarization use cases. For the unstructured text documents, a topic model was applied to determine potential document groupings prior to the summarization model. Post-summarization, three group members applied question answering (QA) modeling to measure the ability of the summaries to answer specific user requirements. This resulted in a feedback loop that informed updates to the HCI and knowledge graph teams.

1.3 Research Purpose and Scope

This paper describes the path taken by the TS working group to build a method to import multiple text documents, process them through an automated machine learning workflow, and produce summaries personalized according to user interests. The achievement of the SCADS grand challenge will require multiple iterations of experimentation and testing to determine the optimal solution workflow. To that end, this first iteration is designed to understand critical aspects of the problem space and employ a small set of data science tools to explore representative datasets. While the target audience for these tools and methods works primarily in the classified information space, this conference is limited to unclassified datasets that do not match exactly the types and formats seen in the IC.

To address both procedural and contextual challenges associated with natural language processing, the aim of this project is to develop a text summarization capability

employing an analyst-specific background corpus to drive term weighting and selection. This aim will be addressed through three primary research questions:

1. Does providing a domain-specific corpus improve summarization accuracy or informational value?
2. How can pre-processing improve summarization accuracy or information value?
3. Without reference summaries, how do we determine machine summarization quality?

The remainder of this paper is organized into three sections. Section 2 defines the specific research problem the group focused on for the conference as well as other works related to the problem area. Section 3 discusses methods applied to achieve a solution that includes dataset details, results, and evaluation metrics. Finally, section 4 concludes the paper with findings, insights, and recommendations for further research.

2. Challenges of Personalized Text Summarization

2.1 Challenge 1: Extractive versus Abstractive Summarization

Automated text summarization is not a new challenge. As far back as the 1950s, researchers sought machine-enabled solutions for extracting “objective” summaries to expedite literature reviews (Luhn, 1958, p. 159). Thankfully, technology has advanced beyond parsing sentences via punch cards! Most text summarization techniques used today reside within one of two broad categories: extractive or abstractive. Extractive summarization seeks to identify and consolidate the most important sentences in a document. This method reproduces key sentences word for word in the summary. Alternatively, abstractive summarization creates new sentences representing the key ideas within a document. Semantic or structural methods drive how these new sentences are structured and consolidated into a complete summary (Mahajani et al., 2019, p. 340).

Extractive summarization expresses sentence value in terms of key word frequencies, prompts, or location. A sentence value is then determined by summing the values of the words or terms in that sentence. In this context, “term” indicates a feature that is bigger than a single word. Some experiments find value in two-word or three-word phrases (bigrams or trigrams) in addition to or, in rare cases, instead of individual words. In those cases, the sentence values are composed of n-gram term values instead of word values. In either case, the final summary is constructed by pulling the highest valued sentences together according to some length-limiting factor established ahead of time. For example, if the limitation is to build a summary no longer than 200 words, the model will be built to assemble the top sentences that, when combined, don’t go beyond the length threshold. The challenge with extractive summaries is defining how the model identifies the word/term values. Once that’s done, the rest can be formulated as a mathematical problem of pulling together the highest valued sentences.

Abstractive summarization employs various classification schemes to encode key ideas that present the most valuable information from a document. These central points must then be paraphrased and linked in a way that mimics human-generated meaning. The encoding on the front end is usually accomplished through either structural or semantic approaches, where the terms are classified according to where they appear in context or how they appear relative to either domain or language-specific characteristics (Dineshnath & Saraswathi, 2018). These NLP techniques are generally much more complex than the models used to extract whole sentences in the methods cited above. Rather than just valuing words and phrases, abstraction requires sentence compression, concept fusion, and scoring the paraphrases before generating the summary. These steps require significant pre-processing of the text, inferencing, and natural language generation to produce summaries consistent with the informational value and intent of the original documents (Gupta & Gupta, 2018, p. 59-60).

Based on how each method derives summaries, distinct advantages and disadvantages were explored before determining this experiment's use cases. Extractive methods are generally faster as they rely on basic statistical approaches to value terms and sentences. While these summaries include sentences from the originating text, the models tend to be biased towards longer sentences (higher value) and may misrepresent conflicting information (sequencing). Abstractive techniques require much more computing power, therefore taking longer, to train models to understand and predict summary sentences. Grammar is sometimes sacrificed and there is always the chance that a hallucination will result in the abstractive summary. This occurs when the summary contains a concept that is not resident in the original text. That said, abstractive summaries can perform better at informational value as they aren't reliant on existing sentence structure (Munot & S. Govilkar, 2014, p. 35). The use cases in this project favored the extractive approach, though we did experiment with a hybrid use case employing both methods.

2.2 Challenge 2: Analyst-driven Context

Language models (LMs) apply probability distributions to either a specific sequence of words or to strings based on the vocabulary on which it was trained (Manning et al., 2008, p. 219-235). As a corpus expands, model performance increases for general language processing tasks (Veres, 2022, p. 8). Since this research effort is tied to the IC and the broader DoD, specific domain knowledge will be required to produce products of informational value to this field. Further, individual analyst interests and queries will add another layer of specificity driving text summarization. To construct a background corpus within these contextual guidelines, the project datasets should include proxies for background analyst knowledge. The challenge is that each analyst is likely to be at different levels of domain understanding, so the model may need to be continuously updated as that domain knowledge changes.

For this project, a large background corpus was built as a proxy for domain context. The documents are of the same type as the targeted text to be summarized so the model will learn terminology specific to the domain (news stories in this case). Additionally,

analyst queries were approximated through topic modeling on the background corpus. The goal is to provide opportunities for the LM to learn from both domain-specific phrases and topics of interest. The potential benefits of embedding domain knowledge and topics into the pre-training phase include a reduction in the processing power required to run the model as well as less redundancy in the summaries.

2.3 Related Work

2.3.1 Extractive Summaries using the Optimal Combinatorial Covering Algorithm for Multi-document Summarization (`occams`)

This project builds on the work of (Davis et al., 2012) in their development of an algorithm that improves summary term selection by maximizing sentence value and minimizing redundancy. The software package `occams` is designed to provide a state-of-the-art multilingual extractive summarization method using first principles of the statistics of natural language. It formulates the extractive summarization task using an optimal approximation of a combinatorial covering algorithm. The covering problem approximates an integer linear programming problem proposed by (Gillick & Favre, 2009). The approximation algorithm has been shown to give over 90% of the optimal coverage with over an order of magnitude performance improvement (Conroy & Davis, 2018). The current release of `occams` has a new highly optimized Rust implementation of the covering algorithm.

The `occams` package tackles the extractive summarization task in three subtasks:

1. The input document or documents (the input text) is segmented into sentences and terms.
2. Term weights are computed indicating the relative importance of the terms in the input.
3. Given a target summary length in characters or words, an optimal approximate algorithm is used to maximize the weighted coverage of terms by selecting a subset of sentences, the sum of whose lengths does not exceed the budgeted target length.

Our focus at SCADS was to adapt and extend the existing library of term-weighting methods to the many aspects of the TLDR Grand Challenge.

While neural-based summarization systems have produced great advances in text summarization, there is still a need for low-cost extractive methods which give a state-of-the-art performance on a wide range of multi-document summarization tasks. With that said, some exploratory analyses of pre-trained abstractive summarizers, from Huggingface.co, were explored at the workshop.

2.3.2 Abstractive Summaries

Unlike extraction, which uses existing sentences, abstractive summarization can produce a shorter version of a given sentence or combine multiple sentences, while preserving its intended meaning (Chopra et al., 2018). To generate these paraphrases from scratch requires much more complex NLP models. Currently, the best abstractive models are based on transformer language models. They are trained on a large background corpus, e.g., CNN/Daily Mail data set, and sometimes fine-tuned for other tasks. The task of training a model typically takes days of computation on multiple GPUs. The models have two parts, the encoder and decoder. The encoder takes input text typical of a part of a document (1024 tokens is typical) and forms a latent representation of the text using a multi-headed transformer model. The job of the decoder is to take this representation and generate an abstract. The models are trained typically with hundreds of thousands of text-human abstract pairs.

Huggingface.co has several pre-trained models, some of which a number of participants explored during the workshop. In particular, some illustrations of hallucinations from these models were observed. These hallucinations, or "fake facts," are a known concern when applying abstractive summarization models. A study designed to capture the faithfulness of abstractive summaries estimated that nearly 30% of the output results of these systems suffer from an ability to create ideas not resident in the reference documents (Cao et al., 2018).

2.3.3 Personalized (Contextualized) Summaries

Developing document summaries specific to individual analysts was initially informed by the work of two groups of researchers (Divoli et al., 2012; Elkiss et al., 2008) who observed readers seeing different insights into the same paper. This is analogous to IC knowledge workers who each focus on different topics specific to their work roles.

In 2008, a team of researchers from the University of Hyderabad, India, endeavored to make a text summarization process user-specific and presented a design for producing personalized summaries of online news articles tailored to each person's interest. The user's online persona was used to model their background and interest. A controlled, qualitative evaluation focused on science and technology articles, indicated better user satisfaction with personalized summaries compared to generic summaries (Kumar et al., 2008).

In a paper on personalizing mobile learning content, researchers introduced a content summarizer as a method for students to retrieve and process information more quickly, based on their interests and preferences (Yang, G. et al., 2012). In this work, probabilistic language modeling techniques informed a user framework and an extractive text summarization system to generate an automatic and personalized for mobile platforms. Their proposal provided an efficient approach to assist students by adaptively summarizing important content.

2.3.4 Audio Summarization

Audio summarization is the task of summarizing audio data. This can be done using solely audio input, using text transcriptions of the audio, or a combination of the two. Working with audio source data introduces additional challenges to the automatic summarization task and there are many active research areas addressing different aspects of audio summarization. Some challenges include (Tur, 2011)

- noisy or low-quality input,
- poorly formed and incomplete input with unclear sentence boundaries,
- multiple speakers that overlap speech, and
- non-word information, such as tone and non-speech noises

Multiple approaches exist to segment audio for summarization, which is the focus area for the SCADS audio summarization effort. Common segmentation techniques leverage duration and content similarity to create meaningful subsections within the data. When summarizing broadcast news videos, (Haloi et al., 2021) assumed a two-second or longer pause in audio indicated the separation between news stories and segmented on any such pause. Hidden Markov Models (HMM) are often used to identify topic segments within the audio. (Yu & Shao, 2022) investigated a method to automatically infer the number of hidden states, or topics, to define for a given audio input to improve upon typical HMM-based approaches to audio segmentation. The approach performed better than traditional HMM approaches but not as well as neural network-based approaches on the same data. One advantage to their approach is that the number of topics into which the audio should be segmented does not need to be determined prior to segmentation.

2.3.5 Question Answering (QA) Evaluations

In addition to work focused on creating summaries, other related work focuses on developing methods to evaluate summary performance. ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a widely accepted metric for evaluating summaries, however, some noted drawbacks include that ROUGE relies on an existing reference summary to be computed and focus on overlapping tokens between the reference summary and generated summary (Eyal et al., 2019).

An alternate approach to evaluating summaries utilizes question answering to determine whether the summary contains salient information from the source document. The Answering Performance for Evaluation of Summaries, or APES, metric uses questions and a QA system to identify how many of the questions can be answered based on the summary. APES focuses on questions about salient entities in a document and can leverage named entity recognition (NER) to generate questions for a given document (Eyal et al., 2019).

The use of QA Evaluations with text summaries assists in identifying hallucinations, thereby leading to an increase in the potential for higher factual accuracy. Most QA

evaluations are applied to abstractive summaries, but some can work with extractive summaries as well. A combination of extractive summaries to answer questions developed by an abstractive summary can be a unique and effective method to increase the factual accuracy of an extractive summary.

3. Text Summary Solution Pathways

3.1 Datasets

3.1.1 Text Analysis Conference (TAC) 2008 – 2011

Initiated in 2008, TAC grew out of the National Institute of Standards and Technology (NIST) Document Understanding Conference (DUC) for text summarization. “TAC is sponsored by NIST and other U.S. government agencies and is overseen by an Advisory Committee consisting of representatives from government, industry, and academia” (Dang & Owczarzak, 2008).

The data includes full-text news articles spanning the period of October 2004 – March 2006. The articles are in English and come from a variety of news sources. The data for each year (2008 – 2011) consists of approximately 48 topics with 20 relevant documents per topic which have been divided into 2 sets of 10: set “A” and set “B”. All documents in set “A” for a topic chronologically precede the documents in set “B” for that topic. The articles in the dataset were organized according to the following variables:

- ID – unique numerical identifier for each article
- Topic – unique alphanumeric code representation for the 48 topics
- Text – the body of the news article

Four human summaries were also provided for each topic and set. These were designed in two batches: straightforward, multi-document summaries for set “A” where summaries were limited to knowledge of set “A” topic documents and updated multi-document summaries for set “B” with knowledge of set “A” and set “B” topic documents.

For topic modeling experiments, the 2008 data was used with the “A” set acting as the training data and the “B” set as the test data. This was a proxy for analyst interests while exploring ways of supervising summarization models.

The data was also used for a “Two Days in the Life of an Analyst” text summarization problem which more closely tracked the original conference “update” task: create summaries for set “B” under the assumption that the analyst has already read the older documents in set “A” and wants a topic update. Years 2008 and 2010 were used for development and 2009 and 2011 were used for testing.

3.1.2 CNN / Daily Mail (CNN/DM)

The CNN/DM dataset is an English-language corpus containing just over 300k unique news articles as written by journalists at CNN and the United Kingdom's Daily Mail (Hermann et al., 2015; Huggingface.co, 2020). The CNN articles were written between 2007 and 2015 while the Daily Mail articles were written between 2010 and 2015. The data was curated using the web archive retrieval site, the Wayback Machine. Each instance in the dataset was organized according to the following variables:

- ID: a string containing a hexadecimal hash of the article web location
- Article: the body of the news article
- Highlights: summary of the article as written by the article author

For this project, the dataset was separated into training, validation, and test sets containing roughly 287k, 13k, and 11k instances respectively.

3.1.3 Microsoft News (MIND)

The MIND data originates from the Microsoft News website and serves as a large-scale benchmark dataset often used for news recommender system development (MSNews, 2020; Wu et al., 2020). Comprised of approximately 160k English news articles, the dataset represents a random sampling of user intake from the fall of 2019. The full dataset includes both article data and user behaviors. For this text summarization project, a small subset of the data was used focusing solely on article data, not user behaviors. Each observation (article) in the dataset was organized according to the following variables:

- News ID – random alphanumeric code specific to that article
- Category – type of news (sports, politics, etc.)
- SubCategory – further division within a category, such as "golf" within sports
- Title – article title
- Abstract – human-generated tagline or short article summary
- URL – web location of the original article
- *Text – full text of target article**
- Title Entities – Wikidata knowledge graph entities contained in the title
- Abstract Entities – Wikidata knowledge graph entities contained in the abstract

* While the original dataset did not contain the full article text, obtaining that information was done by using a utility script to parse data located at the URLs of each article.

The small MIND dataset was used for topic modeling experiments including training and validation sets containing roughly 51k and 42k instances respectively. (Those datasets

were created for this project and differ from the “small” datasets provided by the official MIND website.)

The MIND dataset is designed for news recommendation research. MIND is *not* appropriate for building multi-sentence document summarization models (algorithms) since the data cannot be used for benchmarking due to a low percentage of multi-sentence human-written abstracts (Julia Y., 2022). This does not preclude using traditional document summarization datasets such as CNN/DM or TAC for building document summarization models and then applying those models to frameworks that recommend MIND articles for summarization.

3.1.4 Nixon Tapes (PandaJam)

The PandaJam data set was created by LAS and builds on existing versions of the Nixon-era tapes released by the National Archives and Records Administration (NARA) and further curated by NixonTapes.org. The original tapes contained 3,700 hours of recordings secretly made by President Richard Nixon between February 1971 and July 1973, which were then digitized and annotated with metadata such as recording dates, tape logs, and relevant timestamps within a tape. LAS began working with a version of the data that consisted of approximately 5000 MP3 files, each containing about 2 hours of audio. LAS further split the recordings into shorter segments, ran the segments through automatic speech recognition, and stored the results in a consistent JSON format. The PandaJam data set consists of 12,717 JSON files that contain the following information:

- Duration - length of LAS audio file
- Conversation Title - original file name
- Audio URL - audio file storage location within LAS
- Start Date Time - conversation start date and time, from Tape logs
- Speakers - speakers involved in conversation, from Tape logs
- Transcript by Speaker - text transcript with timestamp and speaker id for each utterance, all automatically generated using AWS services
- Conversation Number - conversation identifier, from NixonTapes.org
- Location - location where conversation was recorded
- UUID - unique identifier, generated by LAS
- Wave File Data URL - file storage location within LAS

3.2 Methods and Results

3.2.1 Coreference Resolution[1]

[1] An introduction to coreference resolution: https://neurosys.com/blog/intro-to-coreference-resolution-in-nlp

There are several reasons why applying coreference resolution to documents is a useful pre-processing step before text summarization methods, particularly extractive text summarization. It has been shown that extractive text summarizers value sentences less when they contain more pronoun references than proper noun references. This is an issue as sentences with important meaning to a reader, given the context of the reference, may be excluded from summaries. Conversely, if a sentence that contains a pronoun reference *is* included in a summary and lacks the appropriate referential context, the reader may not understand or potentially misinterpret the identity of the referent (Sonawane & Kulkarni, 2016).

There are few available open-source coreference resolution models available. The Huggingface (Neural Coref) and AllenNLP coreference resolution models appear by far to be the most commonly utilized and referenced. Generally, it has been concluded that AllenNLP finds more possible coreference "clusters"[2] and has higher Type 1 error while Neural Coref finds fewer clusters and has higher Type 2 error. However, Neural Coref is currently incompatible with Python 3.8+ and spacy 3.x+[3]. We opted to use AllenNLP's model as a basis and supplement the model with custom rules to improve it for our use case, a method similar to a coreference resolution implementation performed by NeuroSYS.[4]

When choosing rules, we opt to minimize the number of replacements made, so as to minimize potential error in replacement and/or syntax after replacement. The rules implemented are as follows:

1. *One coreference is replaced per sentence.*

As the `occams` extractive summarizer extracts complete sentences, we would have the appropriate referential context in the extractive summary, while keeping the coreference resolution model parsimonious.

2. *If there are no meaningful coreference spans in a cluster, that cluster is removed.*

We opt for the same resolution for meaningful coreference clusters as NeuroSYS, for the same reasoning.

3. *Similarly, if there is a not-meaningful span in a cluster, that span is removed*

For simplicity, we define a not-meaningful span as one that does not contain a noun or pronoun phrase.

4. *If a cluster is redundant, that cluster is removed.*

---

[2] A coreference cluster in a group of tokens (spans) within a document that are identified as having the same referent.

[3] Neural Coref progress: https://github.com/explosion/spaCy/pull/7264#issuecomment-986957673

[4] Overview of NeuroSYS solution, as well as analyses of AllenNLP and HuggingFace models: https://neurosys.com/blog/effective-coreference-resolution-model

We identify a cluster as redundant if all spans of that cluster refer to the same noun phrase. Also, if the cluster is of length 1 after removing non-meaningful spans, we remove that cluster.

5. *Only the inner span of a nested cluster is replaced.*

We opt for the same resolution for nested clusters as NeuroSYS, for the same reasoning.

6. *The referent of each cluster is selected based on part of speech, entity, and placement within the text.*

By default, AllenNLP identifies the referent of each cluster as the span that appears first. However, the natural referent of a coreference cluster is not always the first referential phrase. We consider several "sub-rules," in order, to determine the likely referent within each cluster:

a) *If a span contains a proper noun, it is considered more likely to be the referent than a span that does not contain a proper noun.*
b) *If a span contains a noun, it is considered more likely to be the referent than a span that does not contain a noun.*
c) *If multiple spans contain identical parts of speech, spans that contain the most common entity across the cluster are most likely to be the referent than those that do not contain the most common entity across the cluster.*
d) *If a span appears earlier in the text, it is considered more likely to be the referent than a span that appears later in the text.*

Example solution:

Original document (sample from larger training text)[5]

International human rights groups on Saturday urged **Sri Lanka's new president** to immediately order security forces to cease use of force against **protesters** after troops and police cleared **their main camp** following months of demonstrations over the country's economic meltdown. A day after **President Ranil Wickremesinghe** was sworn, hundreds of armed troops raided **a protest camp outside the president's office** in the early hours of Friday, attacking demonstrators with batons.

AllenNLP resolution

International human rights groups on Saturday urged **Sri Lanka's new president** to immediately order security forces to cease use of force against **protesters** after troops and police cleared **protesters's main camp** following months of demonstrations over Sri Lanka's economic meltdown. A day after **Sri Lanka's new president** was sworn, hundreds of armed troops raided **their main camp** in the early hours of Friday, attacking demonstrators with batons.

[5] Source text from the Associated Press: https://apnews.com/article/united-nations-sri-lanka-colombo-ranil-wickremesinghe-bae88ffaafa63feda7af7efc62fcd031

SCADS resolution

International human rights groups on Saturday urged **President Ranil Wickremesinghe** to immediately order security forces to cease use of force against **protesters** after troops and police cleared a **protest camp outside the president's office** following months of demonstrations over the country's economic meltdown. A day after **President Ranil Wickremesinghe** was sworn, hundreds of armed troops raided a **protest camp outside the president's office** in the early hours of Friday, attacking demonstrators with batons.

While the rules identified appear to improve coreference resolution for the SCADS use case, we note that there were no metrics tested for this implementation. We also identify several areas for improvement. First, the code used to implement the coreference resolution can increase in efficiency and speed, and is not set up well enough for generalization of big data. Second, the choice of rules could be modified or added to. The effectiveness of *rule 5c*, for instance, can depend highly on the accuracy of spaCy's NER model (overall F1 of 85.5%). Similarly, *rule 4* relies on the accuracy of spaCy's POS tagging, which appears to fail more often for gerunds and participles.[6] Generally, a larger training corpus improves the accuracy of these rules. Third, we suggest that the AllenNLP model could be modified under the hood to improve accuracy of the generated coreference clusters, which was not measured.[7] Finally, once another model is available and up-to-date, we suggest that the choice of coreference resolution model could be changed entirely.

3.2.2 Topic Modeling (MIND and TAC data via SAS)

Although the MIND dataset already has categories, we wanted to test the results as if it didn't. Using SAS Visual Text Analytics, we use a pipeline to create an unsupervised topic model. The model consists of four nodes - Data, Text Parsing, Concepts, and Topics. In the Data node, variables are assigned to identify the text within the documents, as well as a unique ID. This node prepares the data for analysis down the pipeline.

After assigning and preparing the data, the Text Parsing node is next. This is where the text is tokenized and put through a stop list. We are then able to modify the stop list to allow for better topic extraction. The Concepts node allows the user to create custom concepts, which can also be called entities. Using LIDI programming language, we create concepts such as *Trump*, which categorizes words such as *President Trump, Trump's*, and *Donald Trump.* This also helps with topic modeling by recognizing synonymous terms as one entity. More work can be done identifying concepts. For the purpose of this research, only a few were defined.

Once the data is prepared and parsed, the topic model is run. SAS Visual Text Analytics uses the latent Dirichlet allocation (LDA) method to find topics within the data. The

---

[6] Accuracy metrics for spaCy: https://spacy.io/models/en

[7] AllenNLP model source code: https://github.com/allenai/allennlp-models/blob/v2.1.0/allennlp_models/coref/predictors/coref.py

model finds 18 topics, 17 of which are used. A few of the topics were also combined to better segment the data. The default name is the first 5 words of the topic, though we changed these to better represent the content. This dataset was then exported for additional research.

Along with unsupervised topic modeling, we did supervised topic categorization on both the MIND and TAC 2008 datasets. Supervised topic categorization in SAS Visual Text Analytics uses the pre-labeled topics within the data to create Boolean logic on a training dataset to identify these topics in a testing dataset. The nodes are the same as above, but rather than a Topics node, there is a Categories node. Also, in the Data node, the *category* variable is identified alongside the *text* and *unique ID* variables.

In the category node, SAS generates Boolean logic to categorize documents into the different pre-labeled topics. Once this model is created and tuned, we then put testing data through the model. With that, we see the ratio of true positives, false negatives, and so on.

Figures 3 and 4 show the proportion of true positives by category for MIND and TAC data respectively. The categories in MIND vary by how well we can identify them, with sports and news being the best, and Middle East and kids being the most difficult. This could be because there were fewer total documents for these categories, as well as the underlying nature of them. For the TAC data, most of the topics were correctly identified. Only nine topics were not perfect, and they were still all above a 0.8 true positive rate.

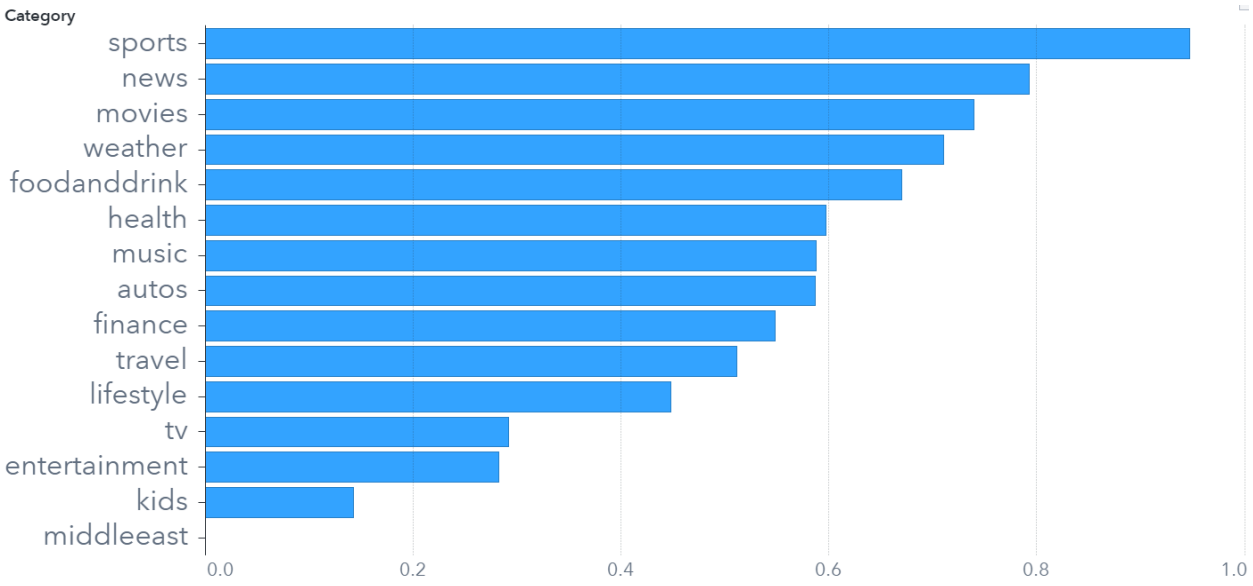

Figure 3 - Proportion of True Positives by Category (MIND Test Data)

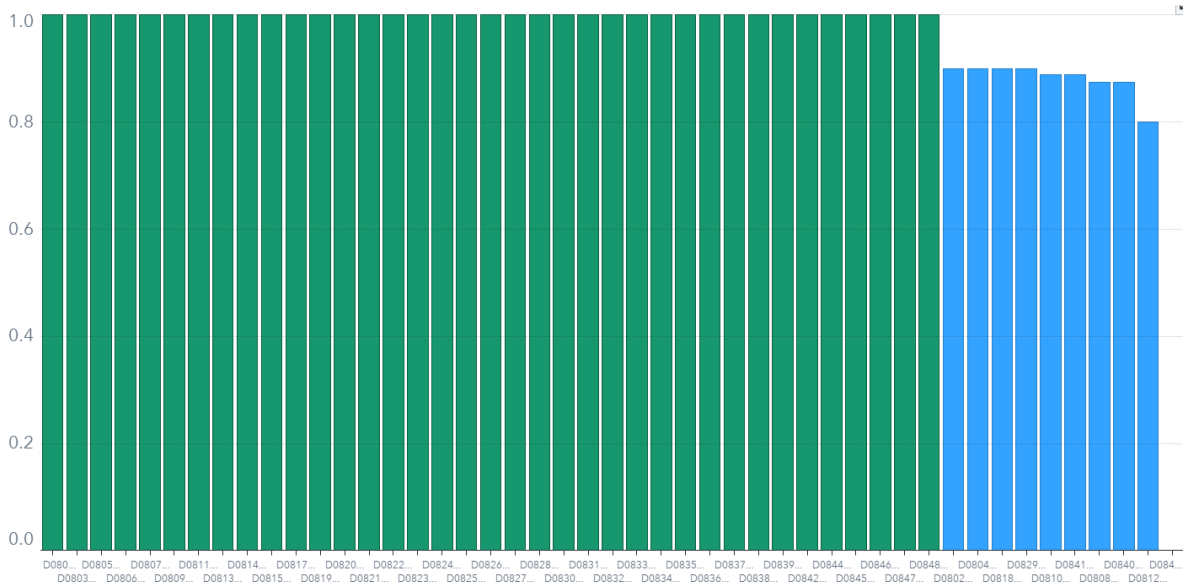

Figure 4 - Proportion of True Positives by Category (TAC 2008 B Data)

3.2.3 Topic Applicable Background for Long Extractive Summary (TABLES)

To further examine the usefulness of these topics generated in Sec 3.2.2, we attempted to create single document summaries of several news articles, leveraging multiple background corpora. It will be helpful to define some terms:

- **General Background Corpus** (GB): a large list of documents that is representative of a language as a whole, usually parsed into n-grams of interest.
- **Topic Applicable Background** (TAB): a background corpus that contains documents of a similar topic to a document that we wish to summarize.

For this experiment, we consider the MIND data for this analysis with major caveats that will be explained in the results of this experiment. We note from the MIND exploratory data analysis (EDA) that most of the human-generated summaries/Abstracts from this data set are 'leading' (Y., Julia, 2022). By leading, we mean that the human-generated summaries mostly just copy the first sentence or sentences from the original article with no abstractive elements. Furthermore, we note that the length of the summaries across the data set was not normally distributed - but actually bimodal in nature as we can see in Figure 5 below.

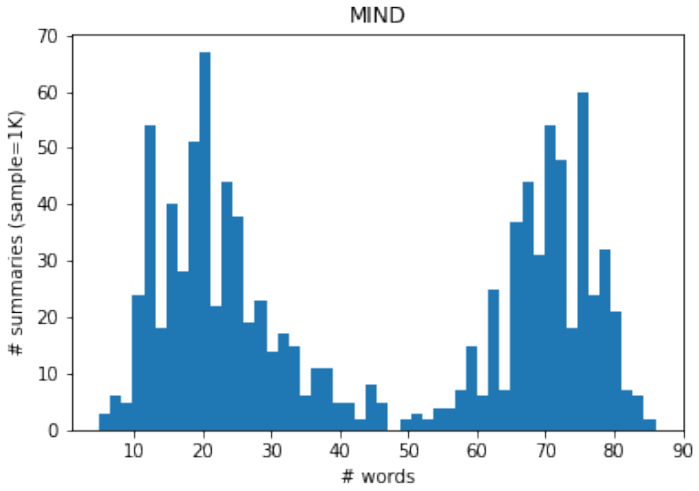

Figure 5 - Bimodal Summary Length in MIND (Julia Y.)

We note that the length of summaries across the MIND data set is bimodal with peaks around 20 and 70, see Fig 5. As for preprocessing of this data - we only looked at documents with a summary that we considered "long" or greater than the mean of about 50. In essence, we take the documents corresponding to the right peak in Figure 5.

We now will quickly introduce the concept of a G^2-test, first used by Dunning (Dunning, 1993), that we will use to extract statistically significant n-grams. This statistical test will be used to help us determine words that are common within a document of interest that we wish to summarize.

Looking at a Document D, and a background corpus B, we define the hypotheses for examining the probability (which we estimate via proportion) of a term t as follows:

$$H_0 : P(t|D) = P(t|B)$$
$$H_\alpha : P(t|D) > P(t|B)$$

Our null assumes the term t occurs with the same frequency in the document as in the background. The alternative states the term t occurs in the document with a significantly greater proportion than that from the background.

Figure 6 - Description of G^2-test Hypothesis

We now want to expand upon how we will leverage the G^2-test for an improved extractive summary. To do this we need to briefly discuss term weights. Term weights are a real number associated with an n-gram that tells us the importance of this n-gram with respect to the document. The simplest approach is to assign the weight to be equal to the number of occurrences of term *t* in document *D*. These can get more complex when considering the position of the term within the document, or the number of times a term occurs across many documents.

We compute the *Fisher Term Weights* of all bigrams as our standard term weight for each document in our sample. For our experiment, we use the bigram counts of the CNN/DM dataset as our General Background Corpus. This is due to the fact CNN/DM is a news dataset with a large breadth of topics discussed and representative of English as a whole. Our TAB is document specific and found based on the topic modeling done in Section 3.2.2. If we wish to summarize a document with the SAS-generated topic of "Law Enforcement", we will create our TAB by counting all bigrams of documents that have the topic "Law Enforcement" excluding the document we wish to summarize. With this information in mind, we now are ready to conduct different types of G^2-tests.

We now need to think about the creation of the GB. At first, we hoped to use MIND text documents that were not categorized with the same topic as our document of interest. This means our GB and TAB would have no intersecting documents. In practice, this would have looked like the term counts for all documents that do not have the SAS-generated topic of "Law Enforcement". However, saying things is sometimes easier than running them. Repeatedly computing word counts for a GB of a specific topic had shown to be cumbersome and took several hours for just one topic (of 14). To alleviate this, we use the CNN/DM data set of news documents to create a GB. We count every bigram throughout all documents in this dataset and this becomes GB for all analyses done in our experiment.

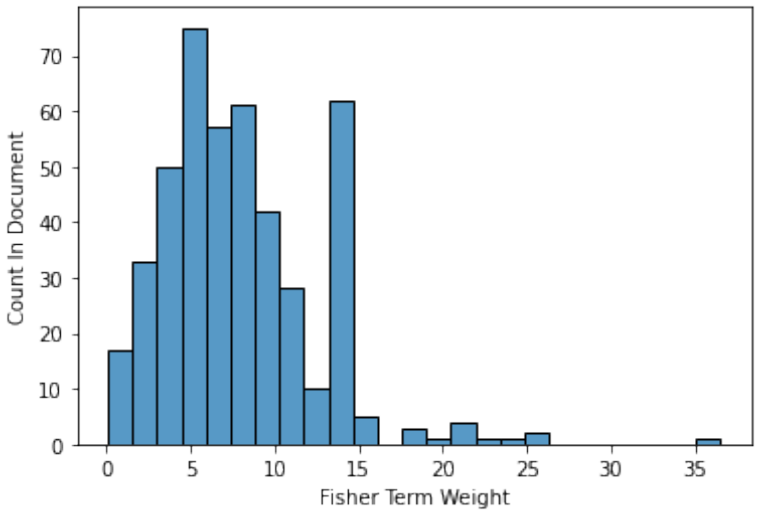

Figure 7 - Fisher Term Weight's Right Skew

In Python, and the associated notebook for this experiment, we create two functions to conduct these tests: *getSigTerms()* and *getSigTerms1()*.

Our 1-test method is just considering the GB corpus. We start with the counts of every bigram in our document of interest and using this GB as our background counts - conduct a G^2-test for every one of the bigrams. The output of this test assigns a p-value to every bigram. This indicates to us the probability that a particular bigram would occur under the assumption this bigram is common throughout the background corpus. We then create a new dictionary of terms that have a p-value below the threshold of 0.0001.

This threshold was used in (Conroy & Davis, 2018) for a similar purpose. These terms are the significant terms of our 1-test method. This first test roughly found 12% of the bigrams in the document of interest to be significant.

We now expand this idea to a 2-test method. The 2-test method takes in the significant terms and their respective counts in the document of interest from the 1-test method. In addition, we supply the bigram counts of the TAB for this document of interest. We then conduct the G^2-Test on this TAB and a subset of terms from the document of interest. After this test, significant bigrams that were output from the 1-test are given a p-value. This p-value is based on the probability this bigram would occur in the document of interest on the assumption it occurs with equal frequency in the TAB. The special idea we use here, is we throw away the significant terms on this second pass. Numerically, we create a dictionary where the p-values from the second G^2-Test are above the threshold 0.01. This gives us a dictionary of terms the general background thought was significant, and filtered out high-frequency bigrams that were stop word heavy and provided meaningful content for a document. Note that once the second test is complete, we observed that approximately 8% of the bigrams from the original document were deemed significant.

Now that we have these significant terms, with respect to each test, we want to be able to update the term weighting to influence the summary creation. As we mentioned previously, we computed the *Fisher Term Weights* for all bigrams in our document of interest. At first, we computed the median value of all term weights across our document of interest and stored this value as a constant. We then added this constant to the term weights of the significant terms produced by each testing procedure. Finally, we set these terms weights in `occams` to compute a summary for our document of interest.

We then created an experiment to avoid the naive use of a median of term weights for our update constant. We wanted to investigate which percentile of the Fisher Terms Weights would give us the ‘best’ update constant for improved ROUGES. So, we do exactly that:

1. We consider each decile of term weights ranging from 0 to 1, inclusive.
2. For each decile, we compute the decile of the Fisher Term Weights for our update scheme.
3. We conduct both the 1-test and 2-test for our random sample of 30 documents
4. We compute the mean ROUGE-2 score of all documents in our sample for each test.

We plot our results of average ROUGE-2 scores below in Figure 8 below. The orange line indicates the 2-test, where we used a General Background followed by SAS-created TAB to find not overly repeated significant terms. The blue line indicated the 1-test, where we used only the General Background to filter for significant terms in our document of interest. We note additionally, the green line is the average ROUGE-2

across occams summaries for the documents in our sample. We note that this does not vary with quantile since we did not do any modifications to the occams scheme.

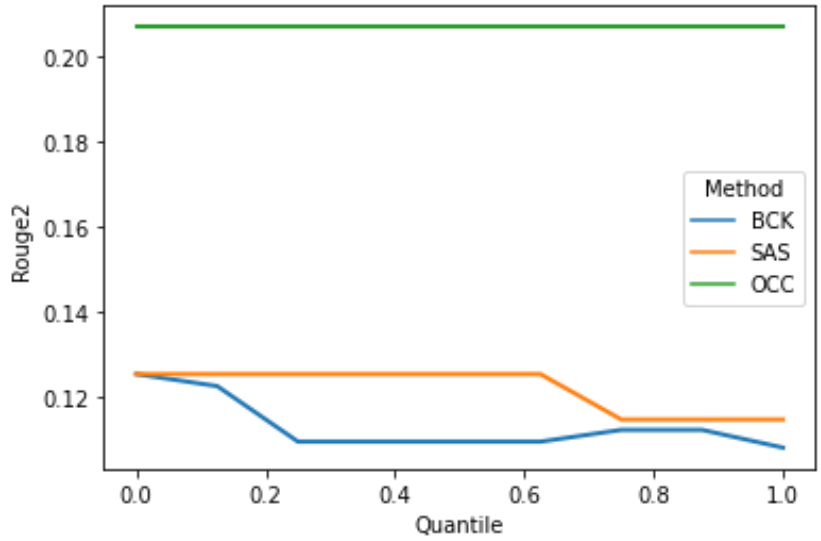

Figure 8 - Average Rouge-2 of a 30-document sample when compared to a 2-test, 1-test summary, & occams summary. We vary over the quantile of base term weights we use as or update constant or significant terms.

As we can see, occams with a positional dense scheme still outperforms the 1-test general background and the 2-test general background and foreground term weight updates. It can be argued that a good reasoning for this is due to the positional weighting occams give to sentences in the MIND documents. As we mentioned before, most of the human generated MIND summaries solely contain the first few sentences of the documents. Since OCCAMs has this positional factor in its term weights- many of its summaries in this sample just contained those first sentences - giving a much higher ROUGE and boosting the average.

We note that the original Fisher Term Weights used in our initial analysis do not have any positional factors involved in their calculations, so this is one potential reason why our summaries, indicated in the orange and blue do not perform better than occams.

One very promising result that we can see from the above figure is that our 2-test generated summary ROUGES bound from above the ROUGES created from using a general background corpus alone. This tells us in all instances when we updated term weights our TAB helped improve the summary.

Now one may be cynical and claim that while the 2-test ROUGES bound the 1-test ROUGES, the difference is not practically significant. While we partially agree with this claim - we believe a central reason for this is the size of each background, not the effectiveness of the term weight updating. Due to the size of the cleaned MIND data - we deal with ~50,000 documents and a topic has at most ~4,000 documents and at least ~40 documents. The rest of the number of documents by topic for large documents is in the table below for reference.

School	4339
Law Enforcement	4070
Family	3651
Basketball	2914
Autos	2141
Fires	2007
Politics	1775
Food and Drink	1228
Baseball	1062
Weather	265
Football	82
Real Estate	40

Table 1 - Counts of "Large MIND Documents" per SAS-generated Topic

This will be expanded upon in the future work section, but we believe if we used a TAB of a similar size to the background, we would see more profound results in the differences between 2-test ROUGES and 1-test ROUGES.

3.2.4 Using Sentence Embedding to Compute Term Weights in `occams`

Pretrained transformer neural language models provide a convenient representation of sentences. These so-called "*sentence embeddings*," take a string of text as input and map it to a vector of floating-point numbers. Depending on the model employed the length of vector 384 to 1024 dimensions, but key to their use is that the length of the vector is model dependent and not a function of the length of the sentence. Sentences whose vector representations are "near" each other are "similar." Such representations can be used to cluster related sentences to identify themes or subtopics in text.

To explore their use on the TAC 2008 – 2011 data, the "Two Days in a Life" problem, the code was written to employ sentence embeddings from the *sentence_transformers* package. The sentences from the TAC 2008 update task were embedded using MiniLM-L6-v2, which embeds the sentences into 384-dimensional space. To illustrate the ability of such a model to cluster sentences we used principal components analysis to further reduce the dimension to two dimensions in the following plot of the sentences from two topics in the A (base "day 1") set for TAC 2008. As the pairwise scatter plot shows, the first principal component (cell (1,1) in the plot) separates the sentence reasonably well. There is some overlap between the embeddings in two dimensions.

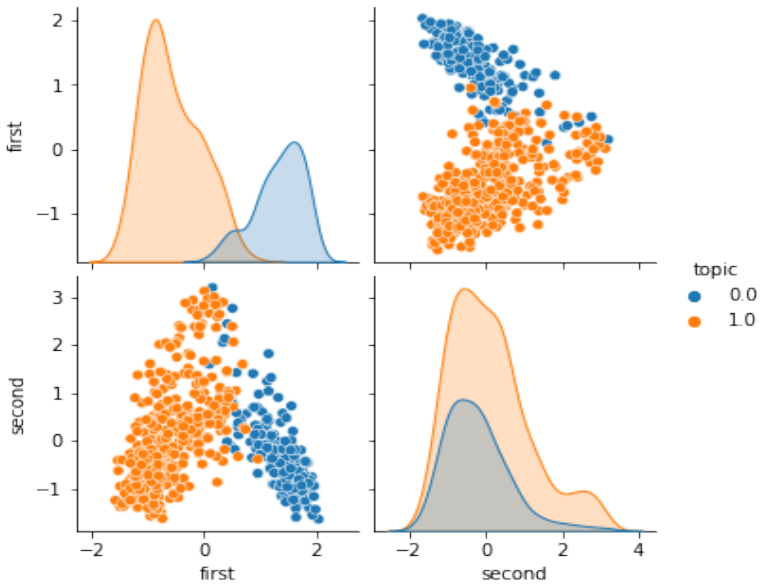

Figure 9 - Sentence Embedding Comparison for Dataset A

The second plot illustrates the sentence embeddings from the A and B sets of one topic. Here we see an all-pairs plot now using 10 dimensions. While there is no separation between the clouds of blue (base topic sentences) and orange (update topic sentences) the individual components show some change in distribution, which could be exploited in a classifier.

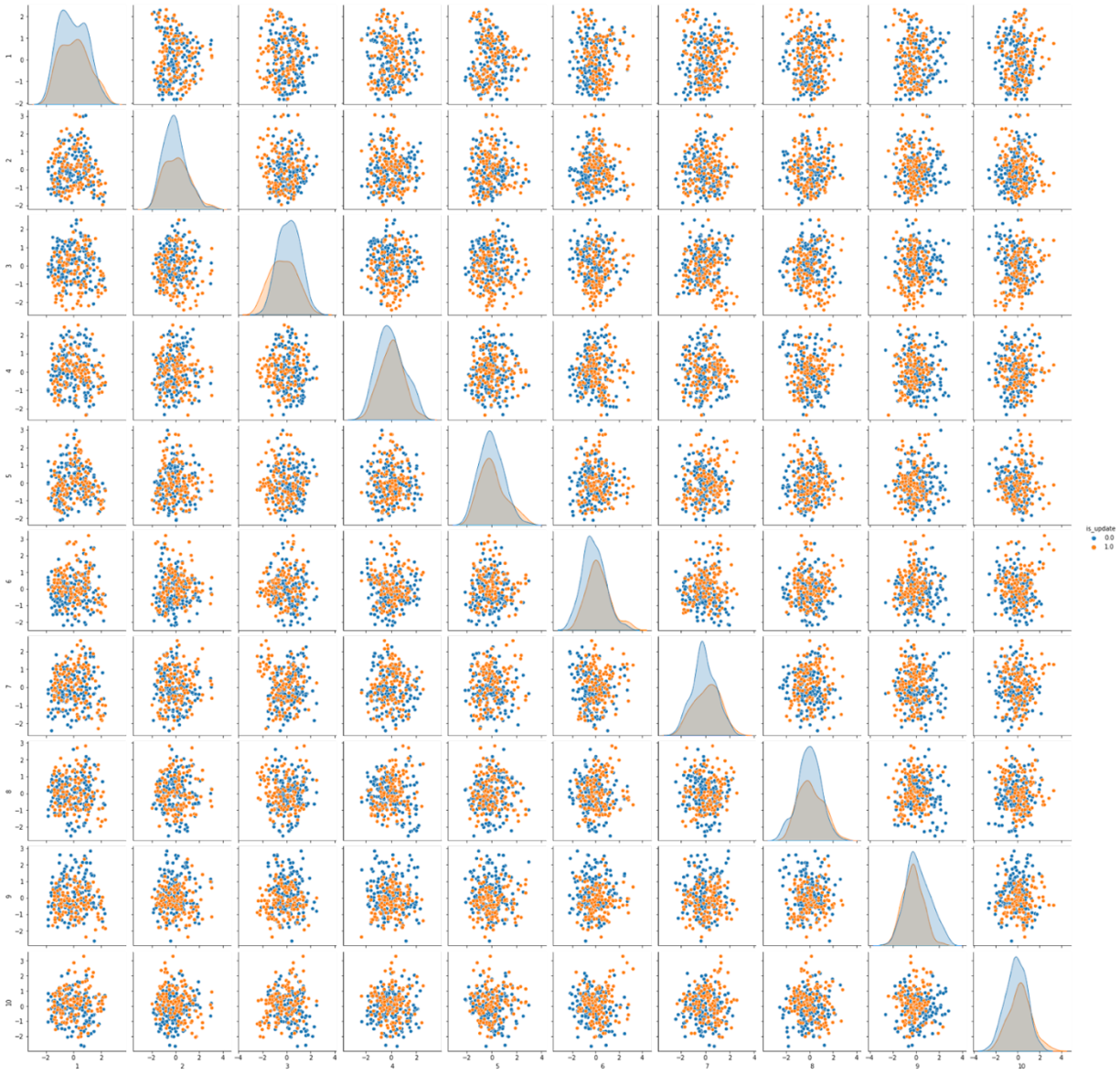

Figure 10 - Sentence Embedding Comparison for Datasets A & B (1 Topic)

The proposal is to train a classifier on the A and B sentence embeddings to assign a probability that a sentence was selected from the A set versus the B set. The aim is to use the classifier to discard sentences in the B set that the classifier determines are "more likely" to be from the A set. Using the TAC 2008 data, for example, about 30% of the sentences in the B set are misclassified as A set sentences. A Python class was written to pass a sentence predicate to the `occams` *SummaryExtractor*, which removes sentences from the optimization process.

The specific classifier that we focused on was Quadratic Discriminant Analysis (QDA). The QDA generalizes Linear Discriminant Analysis, allowing the model to assume that covariance matrices for the two groups are not identical. Additionally, the classifier was trained in such a way that it is possible to tune a threshold where we exclude more or fewer sentences depending on the threshold. This threshold depends on several

variables, and it is expected to be data set specific. After some code refactoring that prevents us from having to embed the same sentences repeatedly, we were able to run more extensive experiments with the text embeddings and the classifier the *t* was computed for them. Because of the nature of the TAC data, the intrinsic parallelism of GPUs is not exploited efficiently. There are 48 topics, and each topic has ten documents for each Update A (aka Day 1) and Update B (aka Day 2). Each document is only around 20-25 sentences on average. So, there are a total of about ten thousand sentences for Update A and another 10k for Update B. The most natural way of doing the embedding is by document. If, however, we group all sentences in a topic together, we can reduce the computation by a factor of ten and leverage much more parallelism. Fortunately, the `occams` *Document* class keeps track of document boundaries and maintains a flat list of sentences. Using this really made hyperparameter searches much more viable for these sentence embedding methods.

In the first example, we tuned this threshold and compared it to the baseline. The baseline here is simply running `occams` on Update B (Day 2) completely ignoring the Day 1 summaries and texts. It showed some potential but not a clear win. The picture below compares the classifier's results to the baseline. It only sampled about ten points.

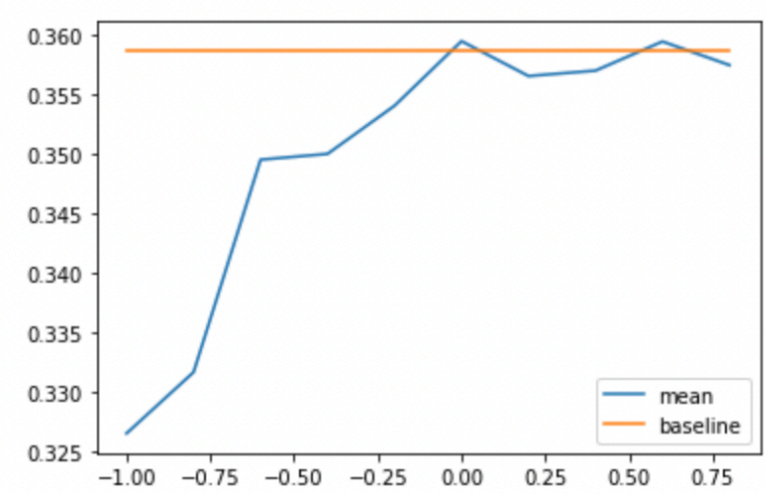

Figure 11 - Initial ROUGE-1 vs Baseline (Update B only)

With the more efficient code, however, it was easy to run a more finely sampled version. The graphs below show the results for ROUGE-1 and ROUGE-2. The mean line is for the classifier based on the sentence embeddings.

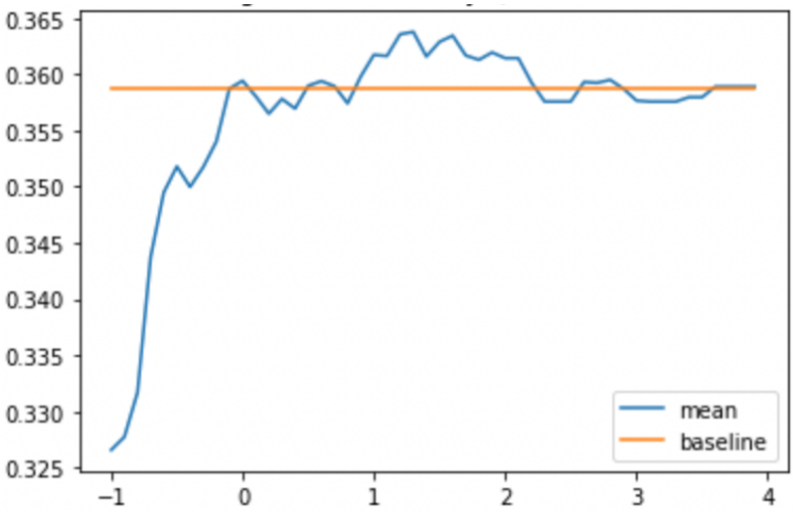

Figure 12 - ROUGE-1 vs Baseline by QDA Threshold

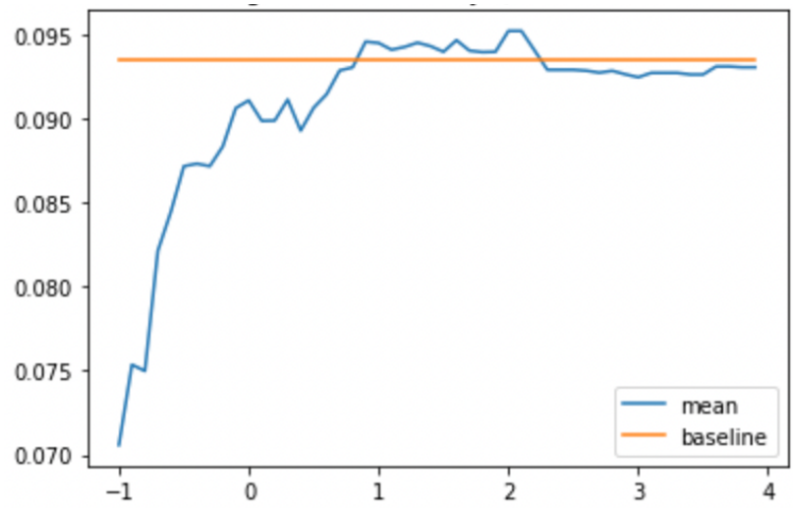

Figure 13 - ROUGE-2 vs Baseline by QDA Threshold

As mentioned before, we use the *sentence_transformers* package to compute the sentence embeddings (Reimers, 2022b). Their default documentation uses a model called 'all-MiniLM-L6-v2'. Per the author's description, this model is designed to be a good tradeoff between performance and speed. Smaller models can embed faster because there are fewer arithmetic operations necessary.

Because we have a relatively small number of sentences and we can embed more of them in parallel, we wanted to compare them to another model. We chose 'all-distilroberta-v1' for our next one to try because we have had a good experience using RoBERTa-based models in previous work (Liu et al., 2019). These results can be seen below. RoBERTa proved to be slightly better in most of the regime where it exceeds the baseline ROUGE-1 results.

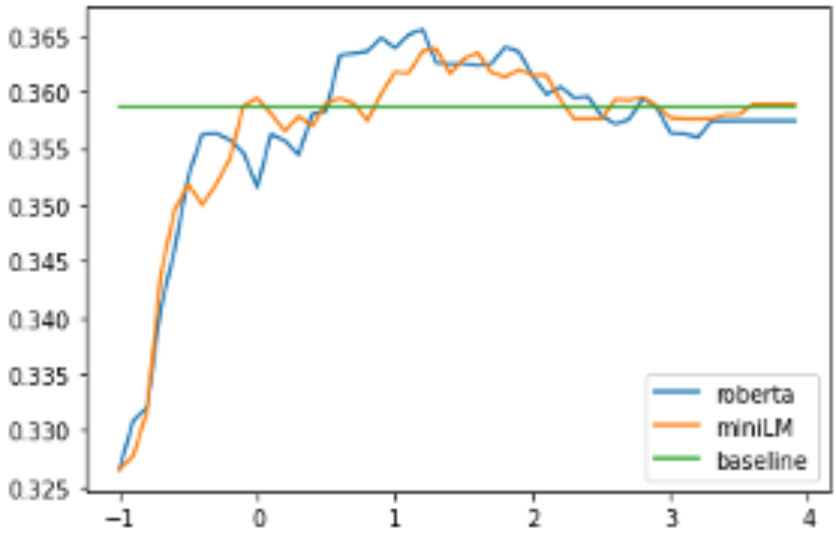

Figure 14 - ROUGE-1 RoBERTa & MiniLM vs Baseline by QDA Threshold

The case for ROUGE-2 is a little less clear:

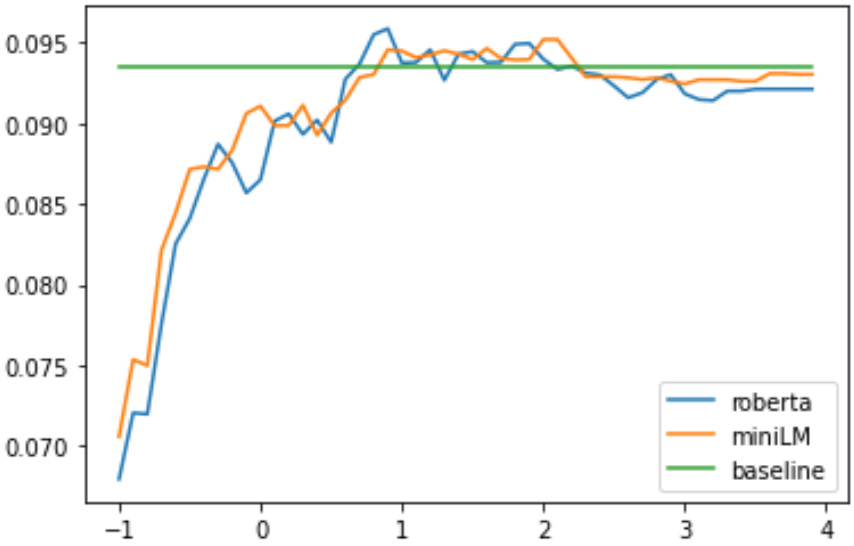

Figure 15 - ROUGE-2 RoBERTa & MiniLM vs Baseline by QDA Threshold

The range where it is best for both ROUGE-1 and ROUGE-2 coincide is when we set our classifiers threshold to about 0.9.

Finally, because of the *sentence_transformers* documentation, we also opted for the 'all-mpnet-base-v2' model. From its documentation, the model is described as follows (Reimers, 2022a):

> "They have been extensively evaluated for their quality to embedded sentences (Performance Sentence Embeddings) and to embedded search queries & paragraphs (Performance Semantic Search). The ***all-*** models were trained on all available training data (more than 1 billion training pairs) and are designed as

general purpose models. The ***all-mpnet-base-v2*** model provides the best quality, while ***all-MiniLM-L6-v2*** is 5 times faster and still offers good quality."

So, understanding those parameters, we tested the 'all-mpnet-base-v2' model as well. Results showed it underperforming both the other two sentence encoders:

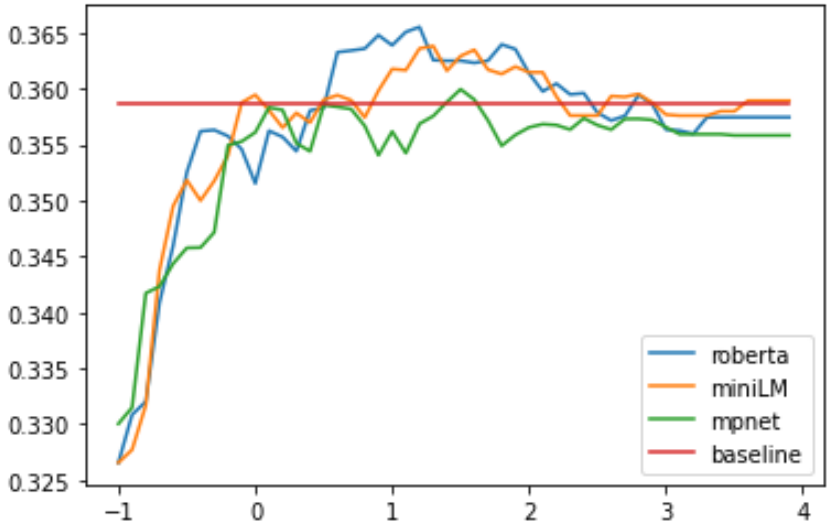

Figure 16 - ROUGE-1 RoBERTa & MiniLM vs Baseline by QDA Threshold

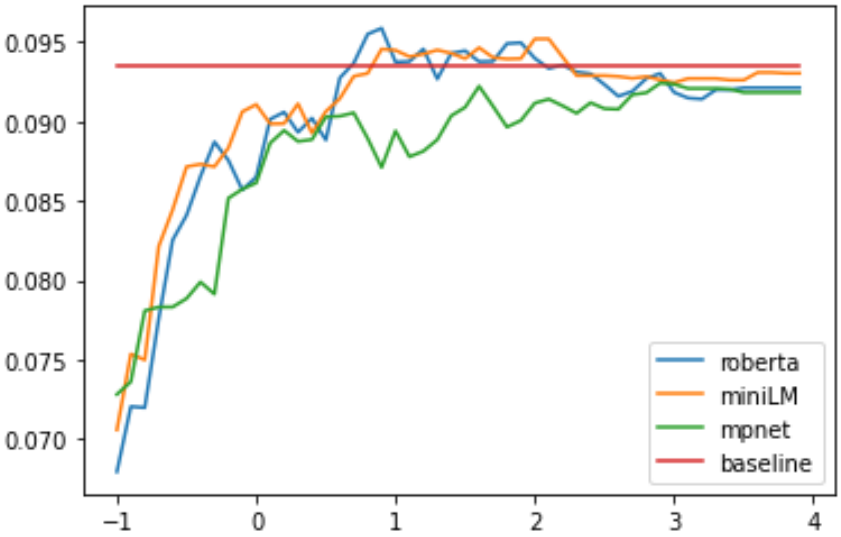

Figure 17 - ROUGE-2 RoBERTa & MiniLM vs Baseline by QDA Threshold

3.2.5 Subtracting Term Weights

Since the term weights are a pivotal input to the software package `occams` that indicates the importance of a concept (instantiated here as a term), we decided to see if we could steer the update summaries (i.e., the summaries from the second day in the

"Two Days in the Life of an Analyst" problem) directly by lowering term weights by those that appeared frequently in the documents from the first day.

This was done by simply using `occams` and its POSITIONAL_DENSE default term weighting out of the box on both the first and second day. I.e., we summarize the second day completely without regard to the first day. This allows us to extract the term weights from both days.

For reference, the POSITIONAL_DENSE term weighting scheme is described in the below excerpt from a forthcoming paper on the `occams` software package:

1. `LOG_COUNTS`: the logarithm of the Laplace smoothed number of occurrences of term i; $w_i = \log(1 + \sum_j a_{ij})$, where a_{ij} is 1 if term i occurs in sentence j and 0 otherwise.

2. `ENTROPY`: the scaled smoothed entropy over the sentences; $w_i = c_i \log(1 + c_i)$, where $c_i = \sum_j a_{ij}$.

3. `POSITIONAL_FIRST`: a variation `LOG_COUNTS` which gives double weight to the first sentence in each document, as inspired by (Gillick and Favre, 2009). Formally, $w_i = \log(1 + \sum_j a_{ij} + \sum_{j \in \alpha} a_{ij})$ where α is the set of first sentences in the documents.

4. `POSITIONAL_DENSE`: a variation of `POSITIONAL_FIRST` which replaces the use of the first sentence with the first sentence above the median score for the document. Formally, $w_i = \log(1 + \sum_j a_{ij} + \sum_{j \in \beta} a_{ij})$ where β is the set of positional dense sentences described below.

Figure 18 - `occams` Term Weighting Scheme Descriptions

Then, we simply reduce the term weights of a term for the second day by some fraction of its weight from the first day. For example, we would reduce the term weight for the word 'pizza' via the below formula:

```
day2_term_weight['pizza'] = day2_term_weight['pizza'] - .3* day1_term_weight['pizza']
```

The hypothesis is that this will steer the Day 2 summary towards new material.

We performed some experiments looking for an optimal value for this fraction (0.3 in the above example). We iterated over possible fractions from 0.0 to 1.0, regenerated summaries with `occams`, and computed ROUGE scores against the human-generated summaries. We then computed the mean ROUGE scores for the dataset. We plot those mean ROUGE scores as a function of the fraction of Update A that was subtracted from

Update B. The resulting mean scores for ROUGE-2 and ROUGE-3 are plotted in Figure 16 for the 2008 TAC data and in Figure 17 for the 2010 TAC data. The baselines in each figure are the mean ROUGE scores for the untouched term weights in Update B. The plots show that the summaries generated from the modified term weights rarely produce improved ROUGE-2 scores. ROUGE-3 scores are often improved over the baseline, but this metric has more variance due to smaller trigram counts. These graphs suggest that this computation is not effective at driving the summaries toward the new material in Update B over Update A.

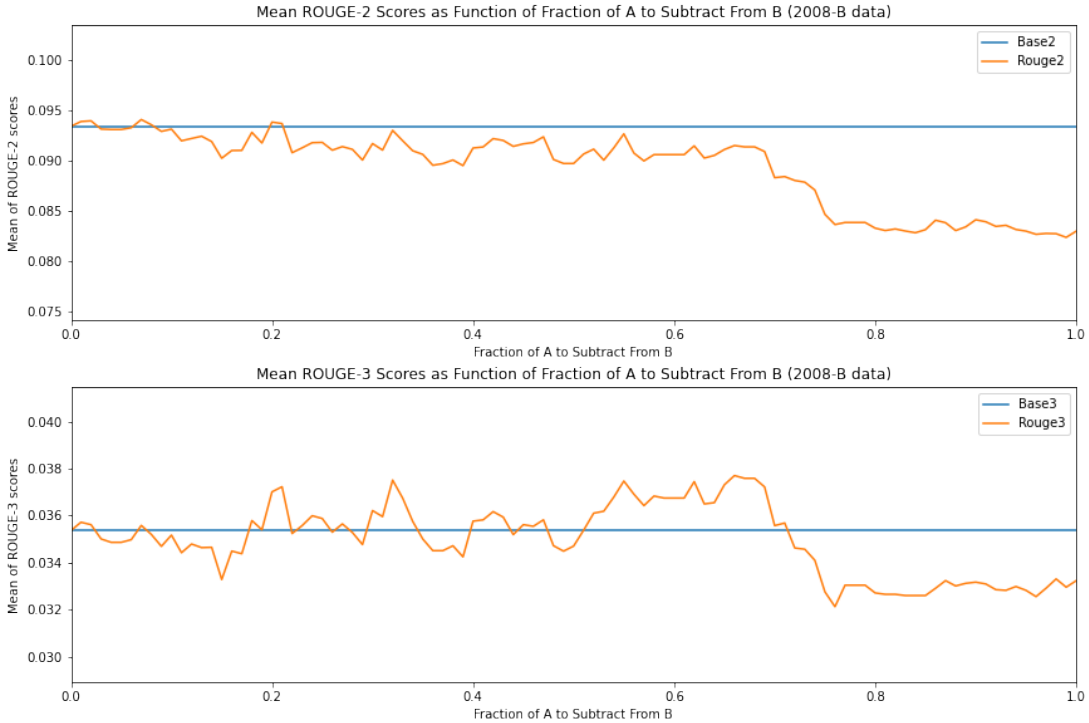

Figure 19 - Mean ROUGE Score as Function of Fraction of A to Subtract from B (2008-B Data)

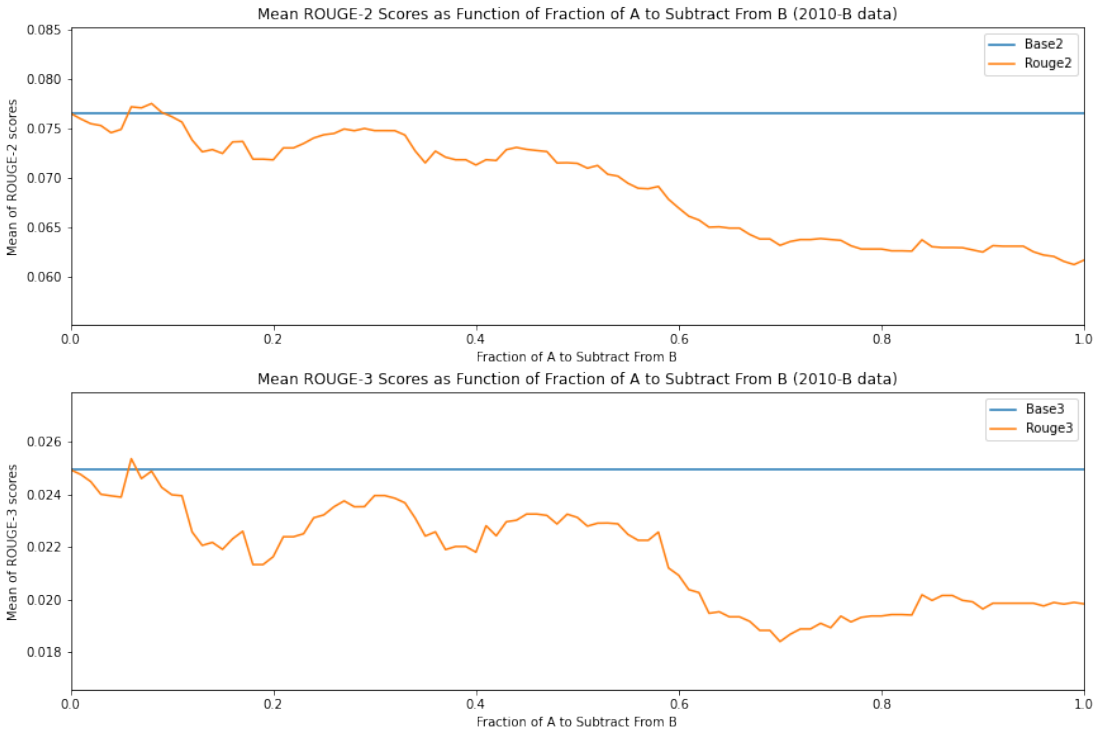

Figure 20 - Mean ROUGE Score as Function of Fraction of A to Subtract from B (2010-B Data)

Upon studying the data, at least some of the effects appear to be due to sentences with unique bigrams that do not really reflect the content of the topics. For example, in one topic, this sentence appeared in many extracted summaries (that is, over a range of fractions): "*That would have been good for the country.*" Every single bigram from this sentence appeared at least as often in the Update B articles compared to the Update A articles (in that topic). However, this sentence contains only generic words and does not contribute to the understanding of its topic. (To make the point more explicit: you cannot discern from that sentence alone what topic it came from or even which country is being referred to!)

To address this, we explored removing or minimizing "background" terms (when available). We obtained a count of bigrams from the CNN/DM data set. Before discussing our next experiments, we pause to provide some statistics of the background counts. The median term count was 1, the mean term count was 13.6, and the standard deviation of this distribution was 624.7. The largest term count was 1,089,117. It was not a surprise to note that many of the highest occurring terms were bigrams involving the traditional stop words.

We performed an experiment in which we modified the term weights from an update by dropping any terms (i.e., setting the corresponding term weight to zero) for which the background corpus count was above a given threshold. Because we did not let Update A interact with Update B in this experiment, we could actually try this separately with each update (A and B) in each year (2008 and 2010). The experiment essentially tests whether removing some of the highest-frequency words from general language would

allow `occams` to better focus on the terms directly related to the topic at hand (compared to how humans would summarize the same topic).

Specifically, we iterated over these “truncation” thresholds (labeled in the graphs below as “max counts" meaning the max counts allowed before we zero’d out the terms), running from the fairly aggressive truncation threshold of 1,000 (which removes 14,237 terms) to a rather mild truncation of only the 45 terms appearing at least 100,000 times in the background corpus. As before, we generated `occams` summaries and compared these to our human-generated summaries. We computed the mean ROUGE scores for the data. The resulting mean ROUGE-2 and ROUGE-3 scores are plotted in Figures 21-24 as a function of the truncation threshold. The baseline in these figures corresponds to there being no truncation at all, that is, all terms are included with their original term weights. These showed more promise. See Figures 21-24.

For the 2008 data (both updates), we see that the ROUGE-2 scores are often above the baseline, and the ROUGE-3 scores are almost uniformly above the baseline. In fact, we were able to find peak values of ROUGE-2 that are likely statistically significant deviations above the baseline. For Update A (2008), the baseline was 0.102134 and we found that a truncation threshold of 28,000 led to a ROUGE-2 of 0.109390. For Update B (2008), the baseline was 0.093462 and we found that a truncation threshold of 3,750 led to a ROUGE-2 of 0.100123. It is curious that these maximums are found in substantially different regions.

Figure 21 - Mean ROUGE Score as Function of Max Count from Background Corpus (2008-A Data)

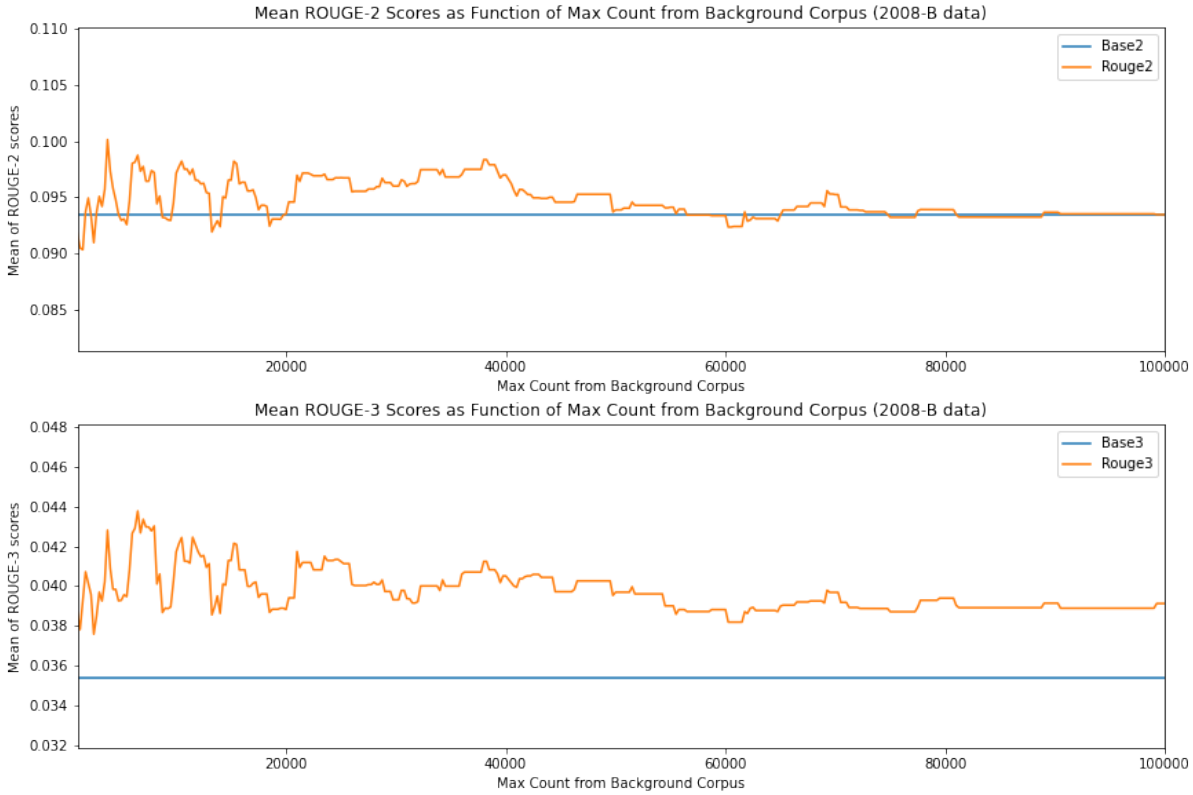

Figure 22 - Mean ROUGE Score as Function of Max Count from Background Corpus (2008-B Data)

The results are different for the 2010 data (both updates). There, we see that the ROUGE-2 scores are rarely above the baseline, and the ROUGE-3 scores are only sometimes above the baseline. For Update A (2010), the baseline was 0.094362 and we found that a truncation threshold of 81,000 led to a ROUGE-2 of 0.094989. For Update B (2010), the baseline was 0.076561 and we found that a truncation threshold of 52,000 led to a ROUGE-2 of 0.0.076823. Again, we note that these maximums are found in substantially different regions.

It is hard to draw a conclusion about the efficacy of this particular experiment. There seems to be a difference in the data between 2008 and 2010. Things just seem harder in the 2010 data. Note, for example, that the baseline ROUGE-2 scores are lower in 2010 than in 2008. Additional EDA of these data sets might shed light on the differences. Also, data from the years 2009 and 2011 were held out until the end of SCADS, and we did not have enough time to investigate them. Perhaps more insights could be gained from including those articles in these experiments. It is also possible that this global truncation might be too blunt an instrument and that a more nuanced use of the background corpus might yield better results. However, this seemed to work for the 2008 data, suggesting that there might be identifiable properties that could indicate when this kind of technique might succeed. More study would be needed to draw any such conclusion.

Figure 23 - Mean ROUGE Score as Function of Max Count from Background Corpus (2010-A Data)

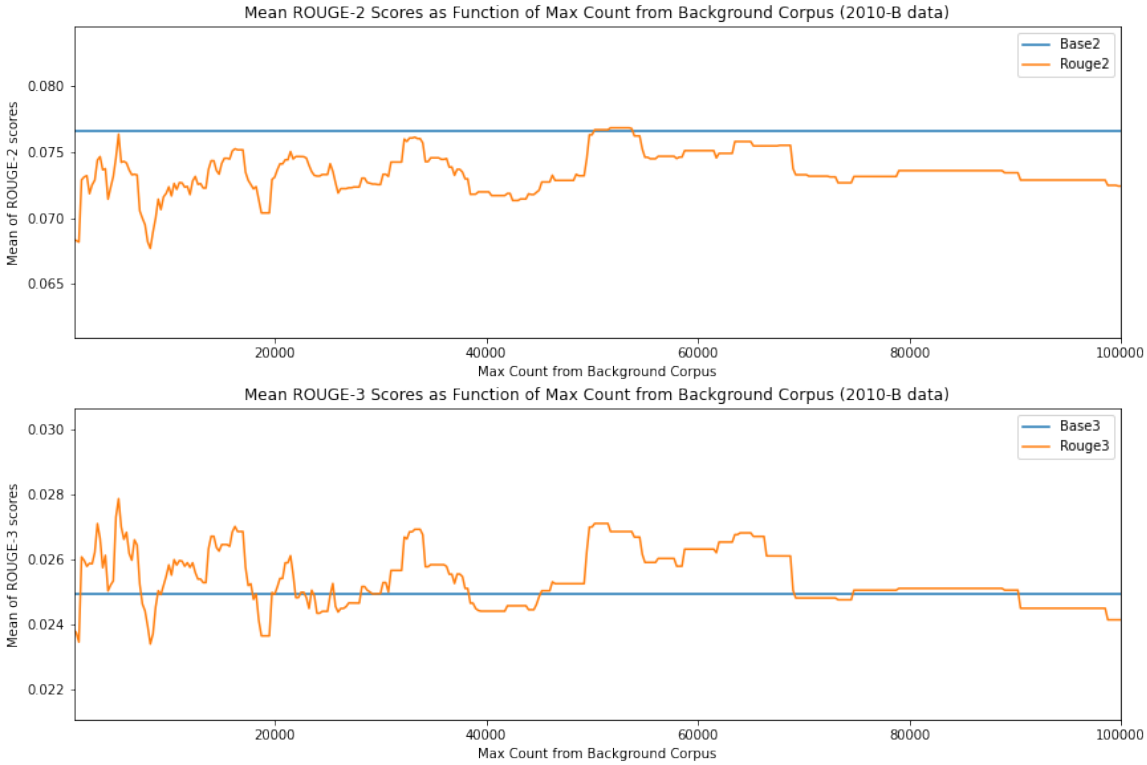

Figure 24 - Mean ROUGE Score as Function of Max Count from Background Corpus (2010-B Data)

There is another observation that piqued our interest: the way the curves in Figures 21 – 24 rise and fall. At the left of each graph, the method is quite aggressive at truncating terms. As we progress to the right, it removes fewer and fewer terms. It is curious to note that the graph oscillates, meaning that as some terms are re-introduced to consideration, some of the extracted summaries change enough that the overall mean of the ROUGE scores can change dramatically. Looking at a peak of ROUGE-2, one can move to the left and see the ROUGE scores drop in average, suggesting that an important term was just excluded. There may be value in identifying such terms. Perhaps even more interesting, moving to the right from that same peak, the average ROUGE scores also drop, suggesting that the terms just included somehow alter the summaries enough to pull down the overall average. What is the significance of terms that would do this? We suspect that there is value to the overall text summarization effort in understanding which terms cause these increases and decreases (and why). This might lead to a useful new metric on groupings of documents. We did not have time during SCADS to explore this question further.

We also conducted a few brief experiments reducing the available term weights using the background corpus counts rather than simply zeroing them out. We tried a small number of very simple things, such as dividing the term weights by the background counts (and also by the log of those counts), but we did not see any encouraging signs of improvement. In the interest of time, we moved on from this line of investigation. It is plausible that with a more principled (or more surgical) comparison of the objects here, one might find a good way to combine them. Indeed, the background corpus counts and the `occams` term weights are not the same kinds of objects, so some care must be applied in combining them. Moreover, any individual term of value in a summary is likely to have low probability, so we might need to use statistical reasoning for rare events compared to the background corpus. Other thoughts on ways forward here include:

- using the `ThresholdLogFrequencySummaryExtractor` class in `occams`, which enables a background test against a background corpus
- normalizing all of the terms so that they can be used directly with one another
- using a different term-frequency scheme other than `POSITIONAL_DENSE`; options include `POSITIONAL_FIRST`, `COUNTS`, and `LOG_COUNTS`, among others
- using a different class of term weights, such as `fisher_term_weights`

We did not explore these, given the time constraints.

3.2.6 Temporal Experiments with `occams`

The documents in the “Two Days in the Life of an Analyst” problem include timestamps. We considered different variants of `occams` LOG_COUNTS which gave additional weight to documents with more recent timestamps within a topic. The motivation was the TAC 2008 baseline (first few sentences of the most recent document for a topic, not exceeding 100 words) and observations from manual inspection that more recent documents appear to have a greater influence on update summaries. The temporal

schemes did not outperform the default scheme, POSITIONAL_DENSE. Some variants were comparable.

For each topic, we grouped set A (10 documents) and set B (10 documents) together and computed the term weights over the AB set. Let {D[1], D[2],…, D[16], D[17], D[18], D[19], D[20]} denote the 20 documents with D[20] being the newest. We considered the following variants of LOG_COUNTS for n=1,2,3,4,5:

- POSITIONAL_NEW_n: Most recent 5 documents added with an *additional* n x weight (e.g., n=1 means double weight, n=2 means triple weight, etc.)
- POSITIONAL_NEW_SCALED_n:
  - n=1: add additional D[20].
  - n=2: add additional 1xD[19] + 2xD[20].
  - n=3: add additional 1xD[18] + 2xD[19] + 3xD[20].
  - n=4: add additional 1xD[17] + 2xD[18] + 3xD[19] + 4xD[20].
  - n=5: add additional 1xD[16] + 2xD[17] + 3xD[18] + 4xD[19] +5xD[20].
- POSITIONAL_1_NEW_SCALED_n: add additional n x D[1] to NEW_SCALED for the oldest document, D[1] (to provide a boost for background material).

A few scalar multipliers were also applied to NEW_SCALED and 1_NEW_SCALED.

The best results are provided below:

2010: updateAB: POSITIONAL_DENSE
Rouge1 0.329033
Rouge2 0.067247
Rouge3 0.017845
Rouge4 0.006565

2010: updateAB: POSITIONAL_NEW_SCALED_2
Rouge1 0.330665*
Rouge2 0.066621
Rouge3 0.018610*
Rouge4 0.006662*

2008: updateAB: POSITIONAL_DENSE
Rouge1 0.346600
Rouge2 0.077187
Rouge3 0.025014
Rouge4 0.011258

2008: updateAB: POSITIONAL_NEW_SCALED_2
Rouge1 0.345079
Rouge2 0.078062*
Rouge3 0.026177*
Rouge4 0.012029*

*The POSITIONAL_NEW_SCALED_2 ROUGE scores are higher than the corresponding baseline POSITIONAL_DENSE scores.

While the POSITIONAL_NEW_SCALED_2 ROUGE-2 scores are higher, they are most likely not significantly different. The 95% confidence intervals for ROUGE-2 scores are roughly +- 0.01 for TAC data, at least for the base (A) set. The paper (Dang & Owczarzak, 2008, p. 14) shows the top performing group of systems as having ROUGE scores in the range of 0.094 to 0.130. This group includes the 8 human summarizers, with the lowest scoring human summarizer having a ROUGE-2 score of 0.108.

A literature search revealed that while the 2008 TAC update baseline performed well in human assessments such as “readability,” the results with automatic metrics were poor (Dang & Owczarzak, 2008). Our hypothesis is that while there is information in timestamps, it is insufficient to leverage using simple methods.

3.2.7 Audio File Text Summary

Audio recordings can be tedious to analyze, especially when hours long. Even when transcribed well, the resulting transcriptions are not easy to read due to the disfluencies in speech and disorganized flow of conversations. Additionally, conversations can cover many different topics.

In order to add audio files into any multimodal TLDR workflow, the system must be able to segment the relevant pieces and summarize them accurately. We hoped that text summarization methodologies might be leveraged to facilitate how analysts navigate audio (e.g. find the relevant portion of the conversation that they want to focus on), and facilitate identification of relevant audio segments for further processing/multimodal intelligence gathering.

We investigated different ways to segment conversations to isolate portions of conversations to summarize:

- No segmentation
    - Utilize the full conversation transcript with no smaller segments
    - Ignore speaker identification and timestamp information
- Speaker diarization/speaker turn
    - Create segments where each segment is a speaker turn in the conversation
    - Speaker turns are identified by the AWS Transcribe service used to generate the automatic text transcript
    - Ignore timestamp information
    - Apply transcript length segments, if needed, if speaker turn segment exceeds maximum input length accepted by summarization method
- Transcript length

    - Create segments up to length X tokens[8], where X is determined as the maximum token length a summarization method accepts as input
    - Segments consist of speaker turns concatenated until the next speaker turn would cause the overall segment length to exceed X tokens
    - Ignore timestamp information
- Semantic clusters
    - Compute sentence embeddings for each sentence in the transcript
    - Utilize HDBScan to identify semantically similar clusters of the sentences

We used the transcript segments as input to automatic summarization tools. We generated extractive summaries using the `occams` package, as previously described. For all extractive summaries generated using `occams`, we used the options `-t LOG_COUNTS --words -2` and either `-b 50` or `-b 100` to generate 50- or 100-token summaries, respectively. For abstractive summaries we used the summarization pipeline from Huggingface[9] with all default parameter values.

The goal of semantic clustering was to identify topics within a conversation that might then allow for segmentation of the audio during the time that topic was discussed. Semantic clustering was done with the Sentence-Transformers model[10]. As such, it required cautious cleaning of the data. Diarization was removed. The transcript was then split by sentence.

While stop words were not removed as it would impact the accuracy of the sentence embeddings, many sentences were replaced when their occurrence in the dataset was too high, and their meaning was not impactful. For example, ‘Yes, sir.’ and ‘Is that correct?’ Additionally, speech fillers like ‘Uh huh’ and ‘Hmm’ were removed. Duplicate rows were also removed from the dataset prior to processing as the embeddings would be the same regardless.

Once sentence embeddings were calculated, several clustering methods were attempted to develop meaningful topical clusters. Because of the poor quality of the speech-to-text transcripts, agglomerative clustering returned more meaningless clusters than meaningful ones. HDBScan was best for creating topically meaningful clusters for single transcripts.

These clusters were not significant enough to allow for meaningful summarization. For example, the cluster below repeatedly mentioned profits, but the sentences are either poorly transcribed or filled with noise, meaning the summaries included noise:

> The point you raised about the profits. After considerable discussion. I think this decision was to And if we can tow work profits as a kind of. Small profits, and

[8] For this work, transcripts were tokenized on white space, as the text transcripts were automatically generated and did not contain complex punctuation, digits, or other constructs that can require more advanced tokenization methods.

[9] Default model for summarization pipeline is https://huggingface.co/sshleifer/distilbart-cnn-12-6

[10] Used all-MiniLM-L6-v2 model for computing sentence embeddings.

> that's something else again are put in there. Control of profits. But there have been talking about profits, windfall profits. And that is a briefing, say profits or control. No one only control profits in the sense that we make promises to high as we can.

So, while a reader might be able to discern they were likely talking about windfall profits and controlling them, the summary does not pull that out. A TF-IDF calculation of the sentences lists the sentences with control in them in 6th and 7th out of 8 places.

Attempts were made to process the entire dataset. There were 1,207,496 sentences in the full dataset before cleaning. There were a total of 259,319 removals. Ultimately, there were 973,845 rows in the cleaned dataset. That is a 233,644 row difference. However, embedding and clustering required more time than was available by the end of the conference.

So two thousand file slices were run through preprocessing, create embeddings, and cluster the sentences. For larger slices of the dataset, it still returned meaningless clusters initially. However, this did identify phrases and errors that could be cleaned from the dataset, which was described above. It also identified artifacts of the speech-to-text transcription, like with the cluster below:

> Well, I'll plan toe. Well, then we'll plan. He wants it toe to be a little bit later, though. You have to go toe hand. Just try toe. For example, I'm going toe. The point is toe. I'll have toe. think it's just well, toe. you have toe to put it this way. Toe what. I don't want to toe. Toe sort of put the spotlight of attention out there on if something could come out of it. Well, I'm just going toe. Well, anyway, they will try toe. I'm going toe. I'm going toe. I mean, I don't want toe.

Clustering was able to gather meaningful insights and gather together sentences that might not themselves have shown up in an analysts search. The below snippet of a cluster includes sentences with Laos, Cambodia, and Saigon in addition to all the sentences with the word Vietnam in them. It was able to discern the similarities in region and pull these sentences together making a cluster that appears to contain all discussions of the Vietnam war:

> It's a national issue and will be here long after Vietnam. Hurt the North Vietnamese for us. Get Vietnam and everything. So that z good a Vietnam in on another, Vietnam in on we made that'll be routine. What do you think about the, incidentally on the E guess Nothing more we could do on the Vietnam side, except they're not going to get the way, way, way. North Vietnam, this is your credibility that we wanna be. North Vietnamese. I hope you don't disapprove of the fact that I'm kicking the hell out of North Vietnam at the moment out of North Vietnam. Saigon forces take. In Laos, we had a settlement. Mr President, Vietnam just knocked that right off off of people's minds. That's a Vietnam thing. For on that the people of South Vietnam's will determine their future without having a Communist or coalition government imposed upon them. And

> then point out that the timing of this was all determined by the North Vietnamese,.At what point do we inform Saigon that we're going to proceed in that way. Well, I think it will wind up with Saigon. Saigon, I suppose. Laos and Cambodia. What the response in Saigon is. You know, the same thing on Laos, that kind of argument, see.

These more meaningful clusters still proved difficult to summarize, as the sentences are jumbled sentences from different sections of audio. The clusters can be used to recompile segments of the audio that can then be fed into the summarizer, but this will face the issue other segment summarization methods face. Below is a TF-IDF generated summary from the full cluster clipped above:

> Yes, Vietnam, that. Vietnam started there for us. This is altogether Vietnam. Oh, you in Vietnam. North Vietnam, this is your credibility that we wanna be. North Vietnam is, Ah, just as a dragon. assuming no action in Vietnam. A few problems with Vietnam and others. So I talked mostly about Vietnam. And the North Vietnamese are bantry. Its trusted with the North Vietnamese. Hurt the North Vietnamese for us. It is North Vietnam and North Vietnam alone. Announce up to South Vietnam.

Audio File Use Case Evaluation

We manually reviewed a selection of segments and summaries generated during this work. The initial plan for evaluating this work included calculating ROUGE scores for summaries on a subset of the transcripts for which human-created gists exist; however, the summaries that were generated were largely nonsensical due to the poor quality of the speech-to-text results and nature of the conversations. Additionally, the gists available for use as reference summaries were quite short and were not reasonable proxies for reference summaries typically used to compute ROUGE. Given the low quality of the summaries and short length of the gists, we anticipated very low or meaningless ROUGE scores and chose to focus on qualitative assessment of the output. As such, the current findings are anecdotal.

Extractive summaries on transcripts with no segmentation and diarization-based segmentation often produced output that would be of limited value to analysts. For example, in one summary for which a human gist exists, there is relevant information missing from the extractive summary that was noted as imported in the gist.

- *Extractive summary:* And then then are after the way, Get him a soon as he could get over. You know, after this program, we'll catch the Apollo guys at 11. We just keep on whenever it could be after him. Four o'clock tomorrow after the time about it Has convenience, right?
- *Human gist:* Nixon and Haldeman are discussing the President's schedule. Haldeman mentions catching the Apollo guys at 11:30.
- *Automated transcript:* You got to go back up. Hill testified. Yep. Okay, fine. You get great. And then then are after the way, Get him a soon as he could get over. You know, after this program, we'll catch the Apollo guys at 11. 30. The Conley

> thing. We just keep on whenever it could be after him. Four o'clock tomorrow after the time about it Has convenience, right? I just want to be sure that No, no, no. We've shifted that already. Okay. Okay. Yeah. Yes. What's please? Oh, fine. Okay

In this case, the extractive summary misses a portion of the time noted in the human gist. The full-time information is included in the automated transcript, but the information is transcribed in such a way that it does not appear to be a time (*11. 30.*) and gets tokenized into multiple tokens rather than one single token (*11.* and *30.*) Additionally, the human is able to provide more context in the gist than the transcript or summary provides, such as who exactly the speakers are and the overall topic of conversation.

Abstractive summaries on transcripts with no segmentation or length-based segmentation were prone to “hallucinations”. The abstractive summary generated for the example transcript used above reflects all of the time information, but also generates additional numeric information (highlighted in bold) that does not exist in the transcript and could confuse an analyst.

- *Abstractive summary:* After this program, we'll catch the Apollo guys at 11. 30.**30.** We just keep on whenever it could be after him . I just want to be sure that No, no, no. We've shifted that already, we've changed that already. The Conley thing.
- *Automated transcript:* You got to go back up. Hill testified. Yep. Okay, fine. You get great. And then then are after the way, Get him a soon as he could get over. You know, after this program, we'll catch the Apollo guys at 11. 30. The Conley thing. We just keep on whenever it could be after him. Four o'clock tomorrow after the time about it Has convenience, right? I just want to be sure that No, no, no. We've shifted that already. Okay. Okay. Yeah. Yes. What's please? Oh, fine. Okay

Other hallucinations in the abstractive summaries were obvious anachronisms, for example in the following sample abstractive summary:

> President Barack Obama calls President Obama on Saturday morning to thank Connecticut friends for their support . President Obama: "I'm sorry I couldn't get there personally, but I know that everybody is working hard. Connecticut will be the first state to come in, and we're just counting on that, uh, that great organization of yours, Tom Thio pull us through"

Given that the recordings were created during President NIxon’s presidency, which occurred decades before President Obama was in office, we would not expect to see any mentions of President Obama in any of the summaries of these tapes. The beginning of the abstractive summary also does not make much sense, as it is unexpected for the president to call himself to thank a third party. While these hallucinations are somewhat amusing given their obvious falsity, it would be necessary to be able to identify hallucinated information and flag it as such in order to provide analysts with abstractive summaries.

Though the clusters created on the full dataset appear meaningful to the human eye, it proved difficult to create extractive summaries that contained the valuable pieces of those clusters. For example, a 15 sentence tfidf summary of the cluster below does not provide any meaningful insights about what was discussed. The only potentially useful sentence is ‘Hurt the North Vietnamese for us’ and only for the hostile sentiment in phrase:

> Yes, Vietnam, that. Vietnam started there for us. This is altogether Vietnam. Oh, you in Vietnam. North Vietnam, this is your credibility that we wanna be. North Vietnam is, Ah, just as a dragon. assuming no action in Vietnam. A few problems with Vietnam and others. So I talked mostly about Vietnam. And the North Vietnamese are bantry. Its trusted with the North Vietnamese. Hurt the North Vietnamese for us. It is North Vietnam and North Vietnam alone. Announce up to South Vietnam.

One effort to apply topic modeling to the cluster using SaS was able to take it as relating to Vietnam. For another cluster, the same topic model produced grand jury, grand. So, there is still work to be done to better tune the topic modeling results.

3.2.8 Sentiment Scoring for Filtering Extracts

For the most part, `occams` determines the best sentences to extract based on analysis of term coverage. While powerful, this approach still leaves areas in which the model can be complemented by sentence level or document level analysis. One possible method is demonstrated here based on the hypothesis that, while term statistics will capture some sentiment associated with a set of documents, using sentiment analysis as a means of constraining the possible output of `occams` to better match the overall sentiment of a set of documents will lead to quantifiable improvements in multi-document summarization. To test the hypothesis, the TAC 2008 dataset was explored to determine if there was any merit in sentiment analysis for the multi-document summarization and update tasks described in 3.1.1. The general approach was to score the sentiment of each sentence in each document individually and examine the distribution of sentiments on a topic-by-topic basis to perform sentence filtering in `occams`.

To score sentiments of sentences in the TAC dataset, the TweetEval model (Barbieri et al., 2020) was selected from a number of pretrained candidate models that can be found in common natural language processing libraries. This model had two main advantages over other possible candidates. First, since it was trained on tweets (140 characters or fewer), it seemed especially adept for scoring sentiment on a sentence-by-sentence basis. Second, most models in this area tend to be trained on reviews and focused on making strict binary classifications of whether the reviews were positive or negative. Sentiment scores for other models tended to be extremely confident about predictions (e.g. 0.99), while TweetEval tended to produce nuanced output that matched with a human scoring a document.

An example distribution of sentiment scores for a set of documents from a single topic for update tasks A and B from the TAC 2008 dataset is shown below. Because the dataset is constructed from news articles about a particular topic, sentiments of sentences tend to be neutral and tinged with negativity; one would hope fair reporting to be mostly neutral while topics that tend to make the news are often negative (disasters, crises, etc.). The example figure is from a topic concerning disasters in unsafe coal mines in China. For the most part, aside from a couple exceptions, the TAC 2008 dataset almost always skews negatively.

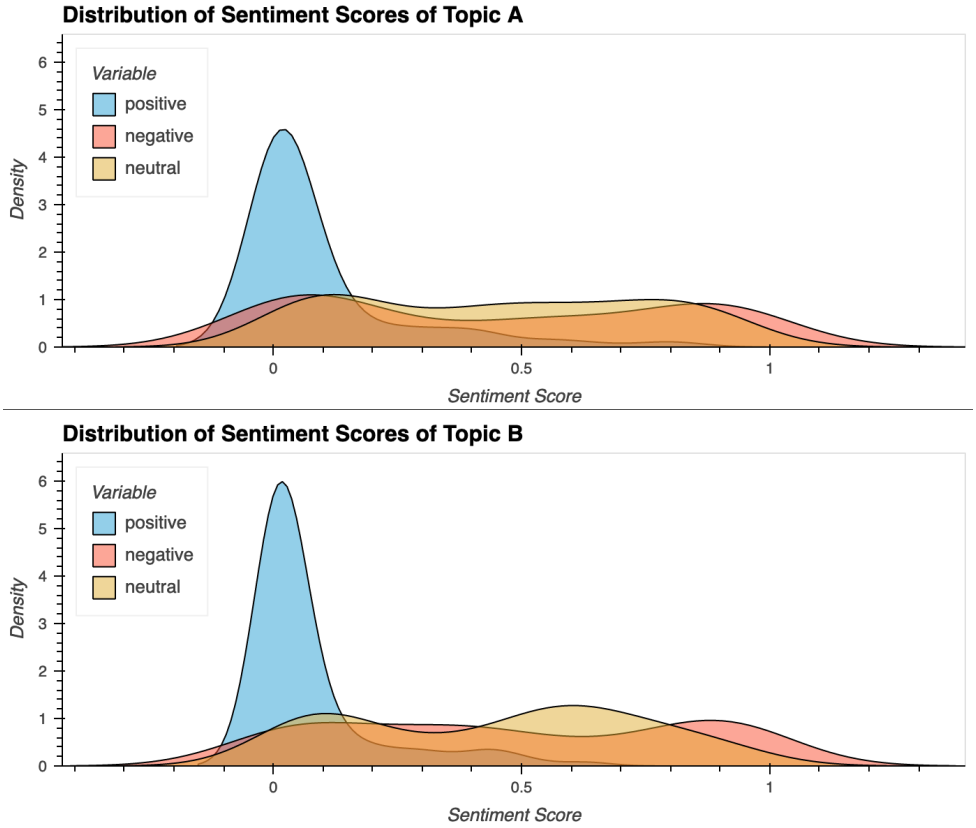

Figure 25 - Sample Distribution of Sentiment Scores

As a first test, an extremely simple set of experiments were performed. Depending on which way the sentiments skewed from neutral (mostly negative), a threshold metric of the ratio of the negative score to positive score was calculated for each sentence. For a given quantile of this metric over the set of documents of a topic and update task, if the metric was lower than the quantile, it was simply filtered out of possible sentences `occams` could use to construct the extractive summaries. Rouge scores were tallied and averaged over the topics and summarized in the figures below, against the baseline score of the document averages.

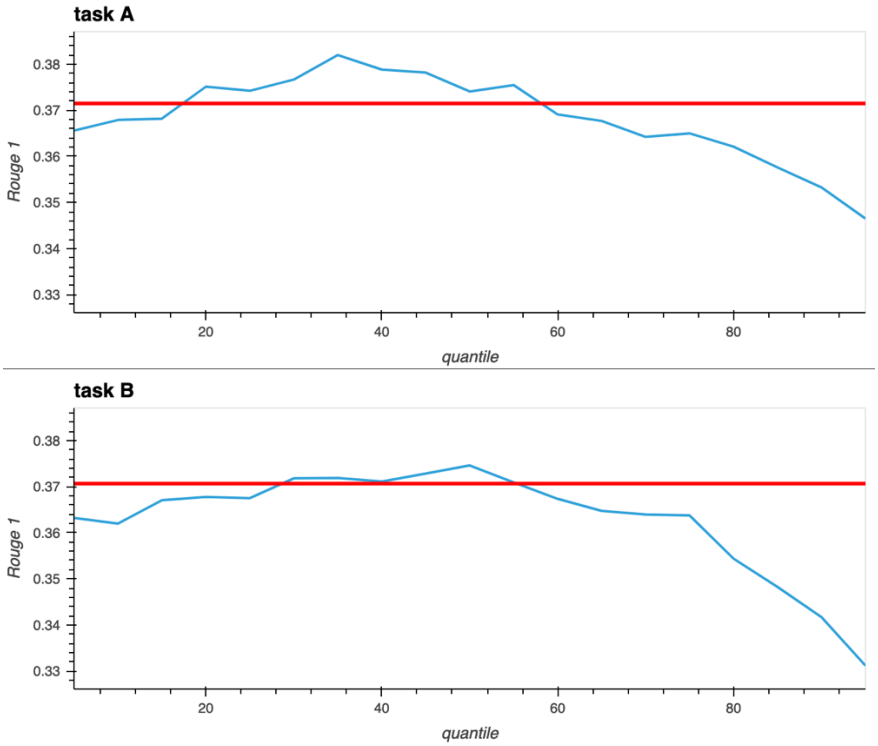

Figure 26 - ROUGE-1 vs Baseline Topic Summaries

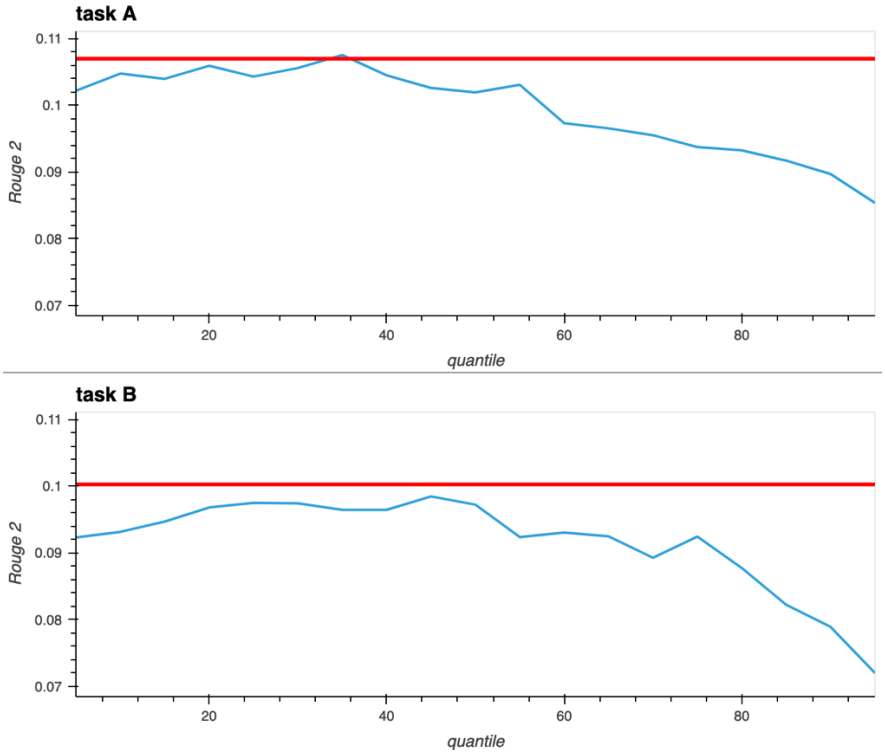

Figure 27 - ROUGE-2 vs Baseline Topic Summaries

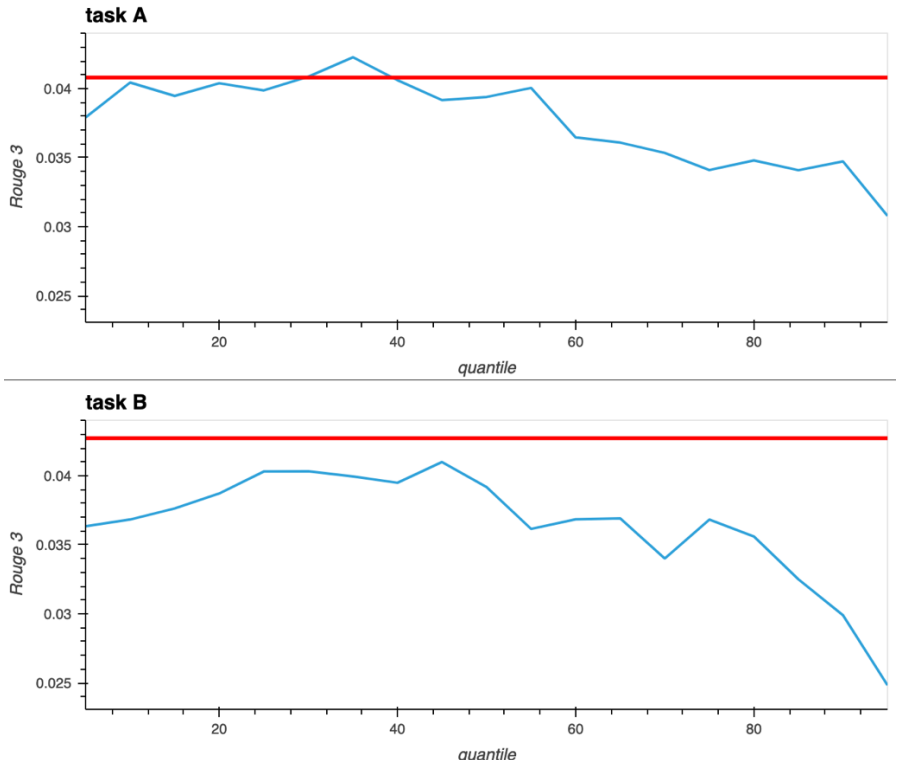

Figure 28 - ROUGE-3 vs Baseline Topic Summaries

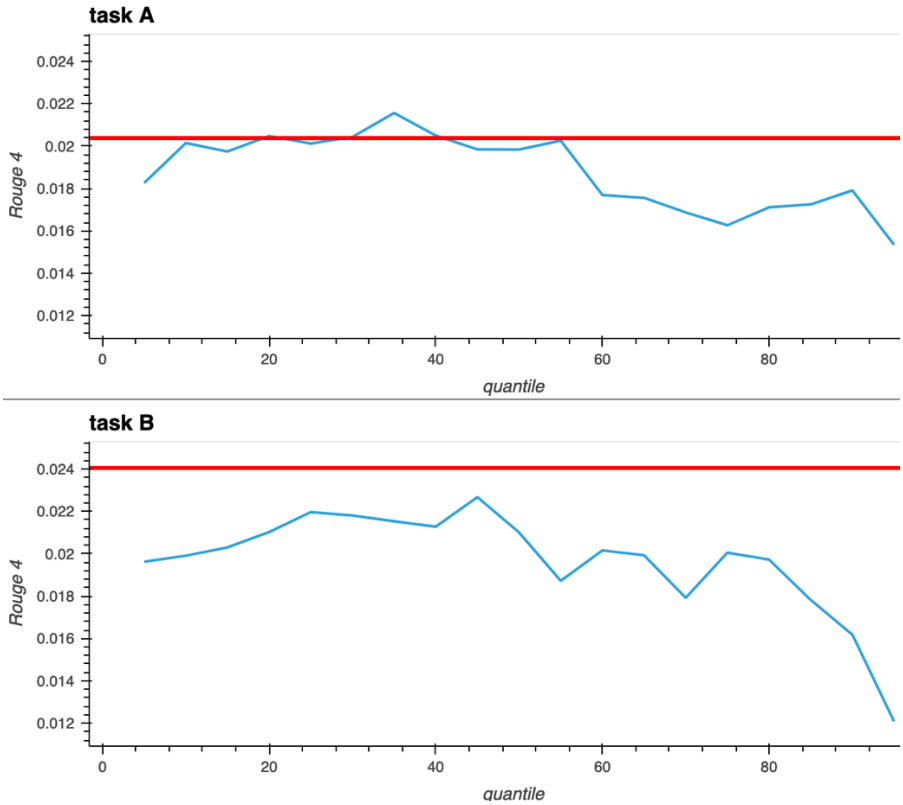

Figure 29 - ROUGE-4 vs Baseline Topic Summaries

As can be seen from the figures, only marginal improvements were achieved, if any. Further analysis shows only 55 out of the possible 96 topic/update sets changed their summaries when using the best performing quantile threshold, with more topics changing to worse Rouge-1 scores than better, in fact.

There are a number of issues with this approach, but further refinement still holds some promise. The filtering scheme demonstrated is painfully simple and cannot account for certain nuances. For example, an ideal approach would attempt to ensure that the distribution of scores produced by `occams` would match that of the set of documents as closely as possible. By simply filtering at a threshold negative to positive sentiment ratio, the few positive, but information-rich sentences that might exist in a series of mostly negative articles would be ignored.

Further efforts might also attempt to introduce schemes to weight or penalize summaries that stray from the underlying sentiment distribution. Rather than filtering out sentences beforehand, configurations of extracted sentences might be regularized by these penalties. Finally, it should be noted that Rouge scores might not be the best metric to evaluate results in this area. Presumably the gold standard human summaries should also reflect the general sentiment of the documents, and some similarity measure of the sentiment of the extracted summary should be developed to compare to the gold standard.

3.2.9 QA Evaluation Metrics

BARTScore, F1, and many other metrics are commonly used within QA models to measure factual accuracy of statements and the overlap of statements. Generation of questions from the source document or from a filtered subset of the source document varies from method to method. The benefit of questions generated from a document filtered down to higher priority sentences is the reduction of noise. Generating questions straight from source can lead to many odd and randomly generated questions that the answer portion of the model may either not be able to answer or skews evaluation metrics.

QAFactEval is an evaluation of text summary that uses a set of many evaluation metrics available at the time of QAFactEval's publication (Fabbri et al., 2022). QAFactEval combines all these metrics and evaluations into its own pipeline. The pipeline is composed of the best combinations of the current metrics that the authors have found. The result of using the pipeline is a 15% increase in evaluation scores over other methods. QAFactEval also works with more updated versions of python than FEQA and QuestEval. QAFactEval can also be used to evaluate the factual accuracy of either abstractive or extractive text summaries.

QAFactEval pipeline filters the source document down to a set of answers through extractive summarization, that are key to the topics of the documents. Questions are

generated from the set of answers. The QAFactEval then evaluates how factually correct the text summary answers the generated questions.

4. Conclusion

4.1 Discussion

4.1.1 Insights

Exploratory data analysis reinforced the importance of using a dataset designed to test how our models would perform in user-specific contexts. In the case of this experiment, we confirmed that the MIND data is great for recommending news articles. However, this dataset is poor for longer document summarization. The “Two days in the life of an analyst” data required sophisticated techniques which leveraged subtle aspects of the data (see sentence embeddings section) when more light-weight techniques (see temporal experiments with `occams` sections) were insufficient.

Pre-processing the data prior to model implementation highlighted methods for reducing the confusion caused by inaccurate part of speech (POS) tagging. While coreference resolution focused on the relationship between proper nouns and their pronoun substitutes, the lessons learned could be applied to broader term-tagging areas to reduce the work required to condense datasets prior to model employment. Similarly, topic modeling acted as a proxy for analyst interests revealing insights into how the model may be trained contextually prior to implementation.

For the unstructured text use case, multiple methods were applied that returned positive results:

- Using a second G^2-test on Topic Applicable Backgrounds, in addition to a general background corpus G^2-test, to filter out terms gives an improvement in summary quality over just using a general background.

- Term weight manipulation showed an interesting relationship between summary quality and which terms were truncated or removed. Further exploration on this method can give potentially powerful insights into the value of specific terms in the context of an entire corpus.

- Sentiment scoring remains inextricably linked to reinforcing context and provides a weighting alternative to term statistics. This variable could become an additional factor in determining sentence value once a method is identified to reduce the chances of negating “negative” documents.

In processing individual audio transcripts, HDBScan proved effective at creating clusters without noise and disfluencies. However, when applied to larger slices of the full transcript dataset, there were increasing numbers of clusters that were made up of

speech disfluencies and noise. The workflow will need additional data engineering to handle these clusters in order to limit results to only the semantically valuable clusters. Additionally, alternative methods will need to be explored to integrate the resulting clusters into any summarization workflow. We suggest attempting topic modeling on these clusters, though it will require engineering to ensure the topic results meaningfully.

4.1.2 Limitations

Topic Applicable Backgrounds (TABs) and General Background term weight updates will not be able to outperform `occams` unless we potentially start with a baseline of positional term weights. Furthermore, TABs of greatly unequal size to the General Background have the potential to give a set of 0 significant terms for a 2-test. This would then make the 2-test equivalent to the 1-test.

4.2 Recommendations for Further Work

4.2.1 Data

One task that we hoped to take on during this conference was to do a topic-based single document summarization of government-related documents with sequential background corpora. We were hoping to use the documents an RP cited to create a topic-specific background corpus, that we define as a foreground for the document. We looked at one dataset, SciSumm, and initially found it very promising until we saw some fatal flaws.

We focused our search on summarization of scientific documents due to how their internal structure and level of field specific jargon mirrors that of a government report: SciSumm was a data set that aimed to collect 1,000 scientific documents (RPs) related to computational linguistics and natural language processing to generate *gold* summaries for each RP. We note *gold* summaries are human-written summaries that are the *gold standard* that all model-generated summaries are compared. In previous datasets (CL-SciSumm only containing about ~30 papers), gold summaries were prepared by humans, namely academic research and faculty with domain expertise in NLP and linguistics. To build up the human summarization of 1,000 documents in SciSumm, the work was outsourced to five graduate students in this particular field. At this point, we thought this was a fantastic dataset.

The issue arose in what was given to these graduate students/experts to create these gold summaries. To start they were given an RP to summarize, the annotators then read all the text of the RP and the sentences that cited the RP in future documents to grasp both the content and scientific applications and wrote a comprehensive summary. This is where our issue arose: these *gold summaries* contain information that is both about the content of the RP and information that is not even included in the RP. This means if we were to create a summary based on information from the single RP alone -

we will not be able to capture all of the information in the gold summary - no matter how good our model performed.

Our issues with this data grew when we realized what was actually given to us to summarize. Each document in this data set had a file containing not the RP citation documents as a whole, but just the sentences that cite the RP. So not only are these citations in the 'wrong direction' (the RP's future citations rather than the works the RP cited) but they do not even contain the full text to allow us to create an RP foreground. With these thoughts in mind - we eliminated the SciSumm dataset from consideration in our project.

To move forward with the summarization task, we have three points we would like to meet with a new potential data set. It would be beneficial to have: documents with expert-generated gold summaries that are based on an RP, a larger collection of RPs (>100), and the full text of a representative number of (at most 30 since some RPs tend to be dense in citations) documents that the RP cited. We believe this dataset, if applied to sequential background corpora, would give us greater insight into this method's effectiveness in boosting the weighting of significant terms.

As for achieving this goal, it would be imperative to have government-related white papers scraped to be our collection of RPs. Even if we are unable to utilize annotators for gold summary writing we are hopeful that abstracts in these reports could suffice. One caveat of this is that the abstracts would have to truly be an abstractive summary of the report, in the sense that they do not contain direct sentences from the body of the report itself.

4.2.2 Methods

Audio Transcripts. Based on the results of generating summaries on automatically generated audio transcripts, more work is needed to produce useful summaries of audio data that would be beneficial to analysts. In future work, it could be useful to incorporate audio and/or transcript metadata into the summary to provide context for the audio in addition to a summary of what is in the transcript. For example, the Nixon tapes provided information about the location where the conversation occurred, and often include information about the day and time the conversation happened plus who participated in the conversation. This conversation metadata could be incorporated into a template that is provided separately from or prepended to a summary of the transcript. Other metadata that might be possible to generate using audio processing techniques, such as descriptions of non-speech noises on the tape and speaker sentiment or tone, could also be included in the summary if deemed to be useful and reliable. Current or previous analysts could provide input on the utility of including this contextual information in the audio summary and effective ways of presenting that information to analysts.

Other work regarding summaries of audio data would be to run question-answering evaluations on the audio summaries to help assess the utility of the summary and

possibly help flag hallucinations in abstractive summaries. This would enable the comparison of different summary generation techniques without having to create numerous human-generated summaries to compute metrics like ROUGE. The questions used in the QA evaluation could be entity-focused, such as those generated by Eyal, et al., or could be tailored to analysts' interests, provided a sufficient QA system exists to automatically answer tailored questions.

Regardless of the modality of the original information being summarized, various methods explored throughout SCADS could be applied to improve summary quality or tailor summaries to an analyst's interests.

Background Corpora. Based on the results from Topic Applicable Backgrounds, more work is needed on the calculation of initial term weights that are updated by the 1-test and 2-test G^2 likelihood tests. We recommend that a future group should examine how our 1 and 2-test systems would work when we input positional term weights rather than positional independent fisher term weights.

If a new team were to start with a baseline of positional term weights (say from `occams` Positional Dense Algorithm) Topic Applicable Backgrounds and General Background term weight updates could possibly outperform `occams` in terms of ROUGE.

Our work on TAB only examined the significance of bigrams in a corpus. It may be beneficial to look at unigrams or other n-grams, when conducting the G^2 likelihood tests to see if this improves ROUGES.

Our work only focuses on creating TABs from SAS-generated Topic Modeling. While this is useful - could human-generated TABs be better or worse in terms of ROUGEs? We have seen human-generated topics in the MIND data -where each article is already classified as "news", "politics", "travels" and etc. While we do not recommend doing further analysis on MIND - we wonder if the TABs created when looking at these broader human topics for an article would be better or worse than the SAS topics. We hypothesize that these human topics will perform worse than the SAS topics on a 2-test because the SAS topics cluster documents on more specific bigrams than do the human topics that cluster on more general terms.

5. References

Barbieri, F., Camacho-Collados, J., Espinosa Anke, L., & Neves, L. (2020). *TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification* Association for Computational Linguistics. https://10.18653/v1/2020.findings-emnlp.148

Cao, Z., Wei, F., Li, W., & Li, S. (2018). Faithful to the original: Fact aware neural abstractive summarization. *Proceedings of the ... AAAI Conference on Artificial Intelligence, 32*(1)https://10.1609/aaai.v32i1.11912

Chopra, S., Auli, M., & Rush, A. M. (2018). *Abstractive Sentence Summarization with Attentive Recurrent Neural Networks* Association for Computational Linguistics. https://10.18653/v1/n16-1012

Conroy, J. M., & Davis, S. T. (2018). Section mixture models for scientific document summarization. *International Journal on Digital Libraries, 19*(2-3), 305-322. https://10.1007/s00799-017-0218-6

Dang, H. T., & Owczarzak, K. (2008). *Overview of the TAC 2008 Update Summarization Task*. Gaithersburg, MD: NIST.

Davis, S., Conroy, J., & Schlesinger, J. OCCAMS -- An Optimal Combinatorial Covering Algorithm for Multi-document Summarization. Paper presented at the *12th International Conference on Data Mining Workshops,* 454-463. https://10.1109/ICDMW.2012.50 https://ieeexplore.ieee.org/document/6406475

Dineshnath, G., & Saraswathi, S. (2018). Comprehensive survey on abstractive text summarization. *International Journal of Innovations & Advancement in Computer Science, 7*(1)https://10.17577/IJERTV9IS090466

Divoli, A., Nakov, P., & Hearst, M. A. (2012). Do peers see more in a paper than its authors? *Advances in Bioinformatics, 2012*, 750214-15. https://10.1155/2012/750214

Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics, 19*(1), 61-74. https://dl.acm.org/doi/10.5555/972450.972454

Elkiss, A., Shen, S., Fader, A., Erkan, G., States, D., & Radev, D. (2008). Blind men and elephants: What do citation summaries
tell us about a research article? *Journal of the American Society for Information Science and Technology, 59*(1), 51-62.

Eyal, M., Baumel, T., & Elhadad, M. (2019). *Question Answering as an Automatic Evaluation Metric for News Article Summarization* Association for Computational Linguistics. https://10.18653/v1/n19-1395

Fabbri, A. R., Wu, C., Liu, W., & Xiong, C. (2022). *QAFactEval: Improved QA-Based Factual Consistency Evaluation for Summarization* Association for Computational Linguistics. https://10.18653/v1/2022.naacl-main.187

Gillick, D., & Favre, B.A scalable global model for summarization. Paper presented at the *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing,* 10-18. https://10.3115/1611638.1611640 https://aclanthology.org/W09-1802/

Gupta, S., & Gupta, S. K. (2018). Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications, 121*, 49-65. https://10.1016/j.eswa.2018.12.011

Haloi, P., Bhuyan, M. K., Chatterjee, D., & Borah, P. R. (2021). Unsupervised story segmentation and indexing of broadcast news video. *Multimedia Tools and Applications,* https://10.1007/s11042-021-11490-y

Hermann, K. M., Kočiský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., & Blunsom, P. (2015). Teaching machines to read and comprehend. *Advances in Neural Information Processing Systems (NIPS),* (28), 1684-1692. https://arxiv.org/pdf/1506.03340.pdf

Huggingface.co. (2020). *CNN / Daily Mail - Datasets at Hugging Face.* https://huggingface.co/datasets/cnn_dailymail

Kumar, C., Pingali, P., & Varma, V.Generating Personalized Summaries Using Publicly Available Web Documents. Paper presented at the *, 3* 103-106. https://10.1109/WIIAT.2008.332 http://dl.acm.org/citation.cfm?id=1487270

Liu, Z., Lin, W., Shi, Y., & Zhao, J. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *Chinese Computational Linguistics,* , 471-484. https://doi.org/10.48550/arXiv.1907.11692

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development, 2*(2), 159-165. https://10.1147/rd.22.0159

Mahajani, A., Pandya, V., Maria, I., & Sharma, D. (2019). A comprehensive survey on extractive and abstractive techniques for text summarization. *Ambient Communications and Computer Systems* (pp. 339-351). Springer Nature Singapore.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

MSNews. (2020). *MIND: Microsoft News Dataset.* https://msnews.github.io/

Munot, N., & S. Govilkar, S. (2014). Comparative study of text summarization methods. *International Journal of Computer Applications, 102*(12), 33-37. https://10.5120/17870-8810

Reimers, N. (2022a). *Pretrained Models.* https://www.sbert.net/docs/pretrained_models.html

Reimers, N. (2022b). *SentenceTransformers Documentation.* https://www.sbert.net/

Sonawane, S., & Kulkarni, P.The Role of Coreference Resolution in Extractive Summarization. Paper presented at the *2016 International Conference on Computing, Analytics and Security Trends (CAST),* https://doi.org/10.1109/cast.2016.7914993

Veres, C. (2022). *Large Language Models are not Models of Natural Language: they are Corpus Models* https://doi.org/10.48550/arXiv.2112.07055

Wu, F., Qiao, Y., Chen, J., Wu, C., Qi, T., Lian, J., Liu, D., Xie, X., Gao, J., Wu, W., & Zhou, M. (2020). *MIND: A Large-scale Dataset for News Recommendation* Association for Computational Linguistics. https://10.18653/v1/2020.acl-main.331

Yang, G., Wen, D., Kinshuk, K., Chen, N., & Sutinen, E. (2012). Personalized Text Content Summarizer for Mobile Learning: An Automatic Text Summarization System with Relevance Based Language Model. Paper presented at the 90-97. https://10.1109/T4E.2012.23 https://ieeexplore.ieee.org/document/6305948

Y., Julia (2022). An EDA of the MIND dataset. *SCADS 2022 Technical Report,*

Yu, J., & Shao, H. (2022). Broadcast news story segmentation using sticky hierarchical dirichlet process. *Applied Intelligence (Dordrecht, Netherlands), 1*https://10.1007/s10489-021-03098-4

Appendix A: Machine Abstract Examples

Below are samples of how the two types of summarization models create an abstract using this paper as the reference document.

A.1 Extractive Summarizer (`occams`)

Extractive Summary of 329 words budget (same as given abstract) via `occams` of the entire document:

Additionally, analyst queries were approximated through topic modeling on the background corpus. The MIND dataset is designed for news recommendation research. If there are no meaningful coreference spans in a cluster, that cluster is removed. However, the natural referent of a coreference cluster is not always the first referential phrase. b) If a span contains a noun, it is considered more likely to be the referent than a span that does not contain a noun. A day after President Ranil Wickremesinghe was sworn, hundreds of armed troops raided a protest camp outside the president's office in the early hours of Friday, attacking demonstrators with batons. While the rules identified appear to improve coreference resolution for the SCADS use case, we note that there were no metrics tested for this implementation. In the Data node, variables are assigned to identify the text within the documents, as well as a unique ID. These terms are the significant terms of our 1-test method. As we mentioned previously, we computed the Fisher Term Weights for all bigrams in our document of interest. Using the TAC 2008 data, for example, about 30% of the sentences in the B set are misclassified as A set sentences. The picture below compares the classifier's results to the baseline. We can then compute the mean ROUGE scores for the dataset. o n=2: add additional 1xD[19] + 2xD[20]. Semantic clustering was done with the Sentence-Transformers model8. * Extractive summary: And then then are after the way, Get him a soon as he could get over. You know, after this program, we'll catch the Apollo guys at 11. We just keep on whenever it could be after him. Four o'clock tomorrow after the time about it Has convenience, right? I just want to be sure that No, no, no. North Vietnam, this is your credibility that we wanna be. A few problems with Vietnam and others. Hurt the North Vietnamese for us. Questions are generated from the set of answers.

A.2 Abstractive Summarizer (GPT-3 Open AI)

Abstractive Summary of Document Abstract on PG 4:

The Text Summarization Working Group conducted a study to explore the feasibility of using machine learning algorithms to generate summaries of text documents. The group experimented with text transcripts generated by historians, journalists, and scientific researchers. Texts originated as audio files (Nixon tapes), news articles (MSN, CNN, Daily Mail), and technical research papers. While some documents were accompanied by human- generated summaries or abstracts, others remained in their original form without any processed summarizations.

The team experimented with the occams software suite to build summaries, testing various parameters to adjust how the model weighted specific ideas in the reference texts. The results were evaluated by either comparing them to human-generated summaries of the original text or through a question answering model that queried the resultant text.

From this work, the team gained valuable insights into how to structure reference and background data samples, represent analyst interests through parameter adjustment, and measure the results to gauge model efficiency.

Appendix B: Repetitive Phrases in Nixon Dataset

As part of cleaning the transcripts for processing with the sentence-transformers model, repetitive sentences that appeared in experimental iterations of clustering were removed in the final dataset. These are those sentences and the corresponding occurrences in the Nixon logs.

- Yeah, right. occurred 1,667 times.
- Yeah, Yeah. occurred 979 times.
- You understand occurred 297 times.
- No way occurred 1,303 times.
- Wait a minute. occurred 141 times.
- Yes, way. occurred 19 times.
- Right, right. occurred 90 times.
- Thank you. occurred 7,264 times.
- That's good. occurred 444 times.
- Don't you think so? occurred 94 times.
- You see. occurred 10 times.
- I see occurred 1,913 times.
- That's that. occurred 49 times.
- Of course. occurred 1,205 times.
- You see what I mean? occurred 99 times.
- See what I mean. occurred 1 times.
- You agree? occurred 53 times.
- I agree occurred 1,285 times.
- my point occurred 743 times.
- your point occurred 253 times.
- Hello occurred 3,635 times.
- Mm hmm. occurred 273 times.
- Uh huh. occurred 5,722 times.
- You have. occurred 30 times.
- I have. occurred 339 times.
- That's great. occurred 281 times.
- Yeah, I know. occurred 148 times.
- Is that right? occurred 124 times.
- Is that correct? occurred 35 times.
- Why not? occurred 196 times.
- It is. occurred 221 times.
- Okay, Okay. occurred 235 times.
- no, no occurred 1,163 times.
- Don't you agree occurred 199 times.
- Well, okay. occurred 234 times.
- Okay, well, occurred 625 times.
- That's right. occurred 3,746 times.
- I mean. occurred 481 times.
- That's true. occurred 265 times.

- But what? occurred 169 times.
- You know what? occurred 553 times.
- I know. occurred 1,144 times.
- Oh, God. occurred 260 times.
- Oh, my God. occurred 237 times.
- So what? occurred 390 times.
- All right, fine. occurred 158 times.
- I understand. occurred 804 times.
- Yes, yes occurred 275 times.
- No, sir. occurred 266 times.
- What happened? occurred 259 times.
- That happened occurred 24 times.
- I think so occurred 746 times.
- Oh, yeah occurred 3,557 times.
- Uh, yeah occurred 1,155 times.
- Come on. occurred 595 times.
- Okay, fine. occurred 269 times.
- That's correct occurred 181 times.
- Yeah, that's right. occurred 279 times.
- Yeah, sure. occurred 841 times.
- I don't know. occurred 2,142 times.
- That's what I mean occurred 63 times.
- I think. occurred 1,203 times.
- Yeah, yeah, yeah. occurred 924 times.
- Uh oh. occurred 373 times.
- Oh, Okay. occurred 63 times.
- Uh, no. occurred 151 times.
- Is it? occurred 99 times.
- I did. occurred 385 times.
- huh occurred 1,977 times.
- uhh occurred 1,471 times.
- uh occurred 56,133 times.
- um occurred 22,973 times.
- mhm occurred 1,310 times.
- hmm occurred 84 times.

Regex was used to remove salutations and standalone letters from the transcript.

- (Hello, \w+\s*\w*\.) occurred 1,052 times.
- ((?<!\w)[EeOoUu](?!\w)) occurred 121,037 times.