2022 Summer Conference on Applied Data Science

Technical Report

Laboratory for Analytic Sciences

NC STATE UNIVERSITY

PUBLISHED
March 2023

Executive Summary

The Laboratory for Analytic Sciences (LAS) at North Carolina State University (NCSU) hosted its inaugural Summer Conference on Applied Data Science (SCADS) in the summer of 2022. The overarching multi-year SCADS "grand challenge" is to generate "tailored daily reports" for knowledge workers (TLDRs). TLDRs would be fairly short reports that contain a summary of information that is specifically relevant to an individual knowledge worker's unique set of interests and objectives. Work done during SCADS 2022 centered around the primary themes of understanding TLDR user needs, identifying content to include in a TLDR, and summarizing content for presentation in a TLDR.

A gap exists around information flow through knowledge work despite a large body of research on various aspects of the analyst workflow. In Section 2 we open with our progress on understanding TLDR users' information needs. We describe a study designed to elicit knowledge workers' information needs by interviewing current and former analysts. We discuss the process for developing a data set based on those interviews and how analysis of that data could be used to inform future design.

Section 3 focuses on novel techniques for identifying content to present in a TLDR. We describe the implementation of two neural network-based recommender models trained on a subset of the Microsoft News Data (MIND) set as a proxy for a TLDR data source. To provide TLDR users additional information along with recommended content, we explore post-hoc explanations of recommendations using Local Interpretable Model-agnostic Explanations (LIMEs) and natural language inference (NLI) predictions regarding whether recommendations support or contradict previously-seen data. Based on the findings from the work presented in Section 2, which found anomalies in a data set should be reported in a TLDR, we also assess the suitability of anomaly detection methods for identifying outliers within data sets.

In Section 4 we describe the development and evaluation of several novel approaches for summarizing information to present in a TLDR. We first document our efforts to improve extractive summarization by modifying how source content is selected for inclusion in a generated summary. We then explore the utility of applying coreference resolution during document preprocessing to more evenly distribute informative content throughout a source document, and describe updates to an existing coreference resolution tool. To influence which sentences are extracted without updating the source document, we examine the effects of multiple different methods which prioritize certain terms or concepts for inclusion in a summary and tailor it toward an area of interest. We assess the suitability of applying text summarization methods to transcribed audio, and present an implementation that generates summaries of knowledge graph data by leveraging the graph's underlying ontology.

We then present empirical evidence of the practical application of this work in Section 5, identifying technologies, models, or approaches that would and would not be effective components of a TLDR system. For example, we find that the performance of Neural News Recommendation with Multi-Head Self Attention (NRMS) models falls only slightly when trained on smaller amounts of user data, which suggests that it would be appropriate for a TLDR use case operating on a smaller data scale than a commercial news provider. Additionally, we demonstrate that NRMS recommendations can be explained using LIME in a way that is interpretable by humans. We also illustrate how summaries of knowledge graphs can be generated by creating sentences based on templates driven by the graph's ontology, and that such summaries are easy to personalize with minimal user input. This approach could offer a straightforward method for incorporating information contained in a knowledge graph into a TLDR. Finally, we document our findings on several mechanisms which ultimately proved unsuitable when looking at other potential methods for tailoring summaries toward new information in a data source, such as excluding previously-seen sentences from extractive summaries or favoring terms in recent documents. In another effort we found that an existing graph-specific anomaly detection algorithm is too resource-intensive to scale effectively for use in a TLDR system.

We conclude this report in Section 6 with recommendations for future efforts to build upon this work. We recommend improvements to models found to be effective for use in a TLDR system, such as further tuning the NRMS model or incorporating more data into background corpora used when generating extractive summaries. We discuss additional metrics to quantify the quality and effectiveness of models and methods employed in this work, which could involve creating a TLDR-specific data set that supports research and evaluation of a full, end-to-end TLDR system. Finally, we lay the groundwork for creating a prototype TLDR system that supports the user needs we distilled through interviews, employs the most promising models and processes we identified, and enables user interaction that can inform future pursuits of the SCADS grand challenge.

Contents

List of Figures

List of Tables

1 Introduction

The Laboratory for Analytic Sciences[1] (LAS) at North Carolina State University (NCSU) hosted its inaugural Summer Conference on Applied Data Science (SCADS) in the summer of 2022. This 8-week, in-person workshop brought together data science expertise from academia, industry, and government to address a "grand challenge" problem of interest to the broader community of knowledge workers in the United States (U.S.) Intelligence Community (IC) and beyond. Roughly 40 university professors, students, industry professionals, and government researchers participated. Beyond the grand challenge, SCADS provided a hands-on learning opportunity for every participant, and served to incubate collaborative partnerships that span affiliations. The inspiration for SCADS stemmed from recommendations in high-level documents from the National Security Commission on Artificial Intelligence (NSCAI) [Com21] and the Center for Strategic and International Studies (CSIS) [TF21], including the need to transform national intelligence by adopting AI-enabled capabilities.

1.1 Challenge Overview

The overarching, 5- to 10-year goal is to achieve the SCADS grand challenge to generate "tailored daily reports" for knowledge workers (TLDRs). These TLDRs would be fairly short reports that contain a summary of information that is highly relevant to an individual knowledge worker's unique set of interests and objectives.

The SCADS grand challenge owes its roots to a document that is produced every day by the IC for the President of the United States, as well as senior staff and cabinet members. Through an intense process spearheaded by the Office of the Director of National Intelligence, which oversees the agencies comprising the bulk of the IC, the sum total of daily intelligence information is considered for inclusion in the President's Daily Brief document (PDB) [otDoNIb] and prioritized based on its priority and relevance to the President's mission. The concept of the SCADS grand challenge of generating TLDRs was born from the notion that all knowledge workers would benefit from a PDB-like document tailored to their own interests and objectives. The PDB draws from the entire intelligence community, which has an $85 billion budget [otDoNIa] and staff of over 100,000, so it is impractical to expend such resources to create similar personalized reports for every knowledge worker. In working to achieve the SCADS grand challenge, participants essentially explore whether modern AI technologies can be leveraged to produce, in some capacity, something akin to a PDB for any knowledge worker.

For the inaugural SCADS conference, we left the details of TLDR composition and design open to enable a broad interpretation of the grand challenge by the participants. To provide some initial guidance on the kinds of research questions that would need to be answered in pursuit of creating a TLDR system, LAS researchers and intelligence analysts developed a structured "problem book" (Appendix B) that outlines several dozen problems of interest. Each of these problems represents a stepping-stone toward achievement of the grand challenge, while simultaneously offering stand-alone value in their own right to professional IC analysts and other knowledge workers. The problem book was organized into five overarching categories:

- Human-focused – problems related to understanding TLDR **user needs**
- Data-focused – problems related to methods for identifying TLDR **content**
- Presentation-focused – problems related to **displaying** TLDR content to users
- Engineering-focused – problems related to **implementing** a TLDR system
- Meta analysis – problems related to TLDR **usage and impact**

Problems in different categories often addressed different aspects of a technical challenge, such as how an algorithm might be applied in a certain scenario (data-focused) and how the results of that algorithm could be presented to a user in that same scenario (presentation-focused). Prior to the SCADS conference, LAS hosted a 5-day pre-event workshop for participants during which LAS staff, plus subject matter experts from the National Security Agency (NSA) and Pacific Northwest National Laboratory (PNNL), provided introductions to some of the technical areas and associated technologies available for use during SCADS.

[1] https://ncsu-las.org/

1.2 Conference Structure

During the first portion of the conference, the participants joined presentations and knowledge sharing sessions facilitated by government analysts, LAS stakeholders, and leaders in data science focus areas. These sessions provided crucial context around the motivation for SCADS, the grand challenge, and the problem set. Some topics that speakers covered included ongoing efforts related to the problem set, the state of the art in technologies needed to create TLDRs, analyst workflows with associated pain points and challenges, and data sets and other resources available during SCADS. As participants learned more about the challenge problems, they also shared their thoughts in formal and informal settings on how they might approach different problems and what skills, data, or other resources were needed for that approach. Through these presentations and discussions, participants connected with other attendees with complementary research interests and self-organized groups of collaborators to work with throughout the remainder of the conference.

Participants spent the remainder of the eight weeks focused on answering their respective research questions. Activities ranged from individual coding sessions to larger group brainstorming and problem solving sessions. Formal presentations and interactive sessions continued throughout this time, and provided participants with opportunities to share progress regarding ongoing activities and formulate additional research questions with colleagues. The final portion of SCADS served as a time to document research activities, findings, and other artifacts generated throughout the conference. SCADS closed with a day-long presentation of work and findings by the participants to an audience consisting of other conference participants, LAS staff, and government stakeholders. SCADS activities were conducted in an unclassified environment.

1.3 Organization and Outline

While the TLDR workflow and design were not defined prior to the conference, a general system workflow quickly emerged which provided a common framework for contextualizing the full suite of research questions under investigation. The workflow consisted of the following components:

1. Multiple information sources exist that contain data in a variety of formats and modalities
2. Some process or processes identify data within those sources that is relevant to a user's interests and information needs
3. Some process or processes condense the identified data into a quickly-consumable format
4. Condensed information is presented to a user via a TLDR
5. User interacts with the TLDR and interaction details are captured

Research groups formed loosely around the following themes reflected in the system workflow. Each group had at least one group member with an established research history in that field who could, in effect, serve as a group mentor.

- Understanding TLDR user needs and optimal experience
- Identifying relevant content for a given user's TLDR
- Summarizing relevant content to include in a TLDR

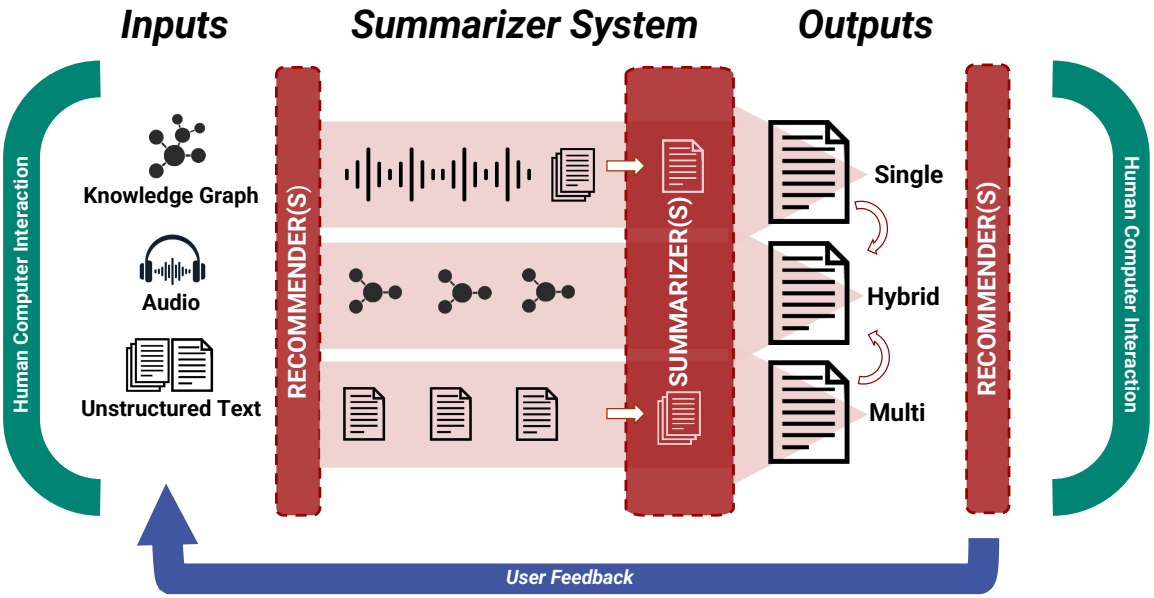


Figure 1: TLDR workflow

These themes encompassed various fields and technologies. Understanding TLDR user needs and optimal experience largely drew from the field of human-computer interaction. Identifying relevant content included experiments utilizing recommender models and knowledge graphs. Summarizing content incorporated text summarization tools, and also considered knowledge graphs as a source of information. Figure 1 shows the high-level diagram of a TLDR system workflow with the research themes or related technologies identified within the workflow.

This report condenses over 300 pages of content generated by the SCADS participants documenting the work done throughout the conference. In preparing this report, we have made some editorial decisions on which portions of that material to include in this document. In Section 2 we open with our progress on understanding TLDR users' information needs. Section 3 we focus on novel techniques for identifying content to present in a TLDR. In Section 4 we describe the development and evaluation of several novel approaches for summarizing information to present in a TLDR. We then present empirical evidence of the practical application of this work in Section 5, identifying technologies, models, or approaches that would and would not be effective components of a TLDR system. We conclude this report in Section 6 with recommendations for future efforts to build upon this work.

2 Understanding User Needs for a TLDR

The grand challenge of SCADS aims to provide personalized suggestions to individual analysts or information customers to improve information awareness and decision making. To generate useful personalized reports, it is necessary to understand what those individuals need and how different people serving different roles share information or access information within intelligence activities. Discussions with analysts at SCADS revealed that a common theme at the foundation of the grand challenge was communicating information among and between people. Some examples of this communication include:

- an individual receiving a tailored report that depends on information pulled from data stores with contents provided by other analysts,
- an analyst being tasked with creating a new tailored report or specific summarized findings to pass to an information customer, or
- the creation of a tailored report that requires coordination of expertise and informational findings from multiple analysts or subject matter experts (SMEs) in the intelligence community.

Therefore, in order to support the challenge of tailored reporting, it was first necessary to outline the information landscape with attention to information flow, transitions, and types through the broader system of operations. We designed a study to learn more about information types and flows in analysts' work. In this study, we addressed three specific questions.

1. What **kinds** of information do intelligence analysts engage with in their analysis work?
2. How does information **flow** in the analysis work of intelligence analysts?
3. What factors influence **how** intelligence analysts engage with information in their analysis work?

2.1 Background and Literature Review

Intelligence analysts are tasked with a broad set of information finding and sensemaking objectives to generate meaningful knowledge from vast quantities of data. Given the data-centered nature of analysis, computational support is essential to enable efficiency and quality in intelligence work. The vision of human-machine teaming has highlighted the need for new technological solutions that take advantage of artificial intelligence (AI) and machine learning (ML) [WFMD21]. For example, recommender systems can help analysts find relevant information they might not know to look for [GHG+16], and natural language summarization can greatly reduce the time needed for human interpretation of large collections of text [PC20]. Complementary advancements in capabilities for identifying patterns, anomalies, and relationships among entities offer the promise of revealing intelligence findings that might otherwise have been missed.

At the same time, research in human-computer interaction more broadly has repeatedly demonstrated that however impressive the computational capabilities, the practical benefits will depend on whether new tools and technology can be appropriately integrated into existing environments and workflows. Practical integration is not trivial. Intelligence analysts conduct analysis within a complex sociotechnical system [Joh05, Nol13]. Analysts work with numerous tools and process varying types of information, and analysis also requires human judgment and creativity on how to best utilize the available tools. Moreover, analysts also frequently communicate with others both synchronously or asynchronously. Requests for information can come in different forms and be initiated by different types of customers. Aspects of the workflow can be highly collaborative, where communication with colleagues or subject matter experts is essential for filling in knowledge gaps. The high variability and complexity of analysis operations create a major challenge for researchers and software engineers as they work to realize the benefits of their contributions in operationally-relevant settings.

While individual tools are typically designed to support specific activities in the intelligence workflow, awareness of information flow across the broader intelligence ecosystem makes it possible to both improve effectiveness of any individual capability as well as ensure their mutual compatibility. Further, a system-level perspective can uncover new pain points or inefficiencies for further improvements. Accordingly, we conducted an interview study with analysts to work towards a system-level model of the information flow in intelligence environments. In contrast with prior research that has studied

individual sensemaking and decision-making processes, we focus on characterizing information passing among multiple people, systems, and data sources. One of the main goals of this study is understanding the different types and formats of individual units of information generated, processed, and shared through analysis operations.

We began by conducting a literature review to provide insight into the following four areas of researcher interest, or themes:

- **Information:** addresses information or proxy information that analysts deal with, how they deal with it, and how the information flows.
- **Process**: addresses workflows or processes that analysts use in their work.
- **Job**: addresses how analysts perceive their jobs and occupational job scope.
- **Collaboration**: addresses an aspect of collaboration or collaboration as a whole among analysts or intelligence agencies, etc.

We found that there was a significant gap in the literature in identifying information types and the ways that information flows in analysts' work. Additionally, little prior work on the topic of information more broadly was conducted with real analysts rather than proxy subjects. Therefore, we believed that we could add significant value to the field by conducting empirical work with analyst participants on the topic of information types and flows.

2.2 Study Design/Approach

Once we identified information passing as our area of research, we designed a study to learn more about information types and flows in analysts' work. The study design process was highly iterative with continual refinement until we conducted the official study interviews. We started by designing a semi-structured interview guide for researchers to follow, which consisted of an introduction to the interview, a debrief script, and a set of interview questions. We developed interview questions that would encourage interviewees to discuss the information inputs, information outputs, and cooperation aspects of their analysis work. We ordered the questions according to study priority so we could optimize information gleaned from the interviews while still adhering to the allotted interview time.

Because we focused on information flow in analysis scenarios, the interview questions centered around the three key themes of information inputs, information outputs, and cooperation.

- **Inputs:** Analysts' "triggers," or the things that cue them to begin an analysis task, could be thought of as a starting point of their information flows. One key aspect of understanding how information flows in analysts' work is understanding the sources of triggers that lead to an analysis task and the interactions with those sources. In our interviews, we asked questions about where triggers originate, from whom, and in what form they arrive. We also asked about how analysts understand the intentions behind a given trigger and what someone may want from a given trigger that isn't explicitly stated. For example, we asked how much interaction analysts have with those who send triggers, and how much reframing, redirection, and/or discussion is involved to understand what someone really wants to know based on their request. These questions were intended, in part, to help us map out whether information flows are largely linear or are more cyclical (i.e., extending beyond a simple input-process-output model). We also wanted to explore any other information "inputs" that analysts receive that may not necessarily cue them to begin an analysis task, but still influence their workflows.
- **Outputs:** On the other end of the information flow, we wanted to understand the information outputs that analysts produce. This includes formal outputs, such as finished intelligence reporting, as well as informal outputs, such as conversations where information is communicated between co-workers. There are also a number of outputs that fall somewhere in between formal and informal, such as notes that analysts take which are then recorded and serialized in databases. We also asked questions regarding how analysts might tailor their information outputs based on what they know about the intended recipients. For example, if an analyst is preparing the same information for three different stakeholders, how might the presentation of that information vary for each recipient? These questions help us better understand the ways that information is modified throughout an analyst's information flow.

- **Cooperation:** Finally, we wanted to investigate cooperation as its own theme, with particular focus on how it pertains to an analyst's workflow. As we understood from prior work and conversations with analysts, there are often barriers to cooperation among intelligence analysts. Factors such as frequency of disseminated reports and the amount of work completed with your name on it have an impact on individual promotions, which may lead to more hostile and self-regarding work practices. Others commented on how many individuals were involved in their analysis process or the process of preparing raw intelligence for curated products like the PDB. To better understand these anecdotal perspectives, we wanted to hear how analysts experienced working in collaborative ways in a formal setting.

We conducted six pilot interviews as part of the study design process. Pilot interview participants consisted of individuals who had worked adjacent to analysts and were familiar with analysts' work plus an individual who had worked as an analyst but was outside of the study recruitment pool. Four of the pilot participants answered the questions from the perspective of an analyst and the remaining participants responded from their own non-analyst perspective. This allowed us to test the interview questions in the context of analyst work and in realistic scenarios with proxy participants, and enabled the facilitators to become better-versed in analytic language. Another outcome from the pilot interviews was that we discovered the benefits of inviting participants to sketch the components of their analysis process, and included prompting interviewees to do so in the final interview script.

This research study was approved by North Carolina State University's Institutional Review Board (IRB) and passed a review by the Department of Defense. We recruited participants by putting up advertisement flyers around the work space and by word-of-mouth among LAS employees. Further, SCADS participants who had analyst experience were made aware of the study. Interviews started only after interviewees provided explicit verbal consent, and participants had the option of stopping the interview at any time without consequence. Interviews lasted approximately one hour. We used a semi-structured interview style that leaned more toward a structured style as we had a detailed interview guide with prompts for each questions. Conducting our interview as semi-structured allowed us to reword and reorder questions in response to the natural conversational flow of the interview along with providing leeway to explore interesting tangents within the scope of the research question. We created a template[2] for note takers to follow that included the interview questions and broad themes from the interview guide. Immediately after each interview, the facilitators worked together to refine and clarify the notes as necessary to prepare them for qualitative analysis.

We interviewed 16 analysts who had worked in a wide variety of roles, including participants who had experience working domestically as well as those deployed internationally in the field. Descriptions of work roles included *signals intelligence analyst*, *discovery analyst*, *language analyst*, *cyber intelligence analyst*, and *geospatial intelligence analyst*. Participants' experience of working as analysts ranged from 6 months to 26 years. Through multiple iterations of coding interview notes, we developed a set of heuristics that can be applied to subsequent coding efforts (see Appendix D).

2.3 Discussion

Initial analysis of the interview data showed that there are both formal and informal types of information that analysts reference in their work. In a formal sense, analysts work with many different types of data based on their operational role: cyber analysts work with network data; language analysts usually focus on foreign language communication; and geospatial analysts spend time making sense of maps and charts. One repeated theme was that informal conversations occur within the community that directly impact understanding,and that information flow has both formal and informal aspects. For example, requests are sent via formal channels or are based on established priorities,while others arrive as "hair on fire" walk-bys that re-prioritize an analyst's work for the day. These formal and informal activities highlight the types of information that analysts work with and also how information flows in analysis work, as one analyst's output is often used as input to another analyst's workflow.

Factors that influence how intelligence analysts engage with information throughout their analysis work include the size of the system and dissemination challenges. It is not uncommon for analysts to know each other, and to rely on these pre-existing relationships to know what other analysts may be working on. These pre-existing relationships also help with distributing information, because analysts

[2]The notetaking template can be accessed using this link.

will call interested parties to help explain particular sections of the document. Unfortunately, we also found that even though a finished might report be disseminated, people are not guaranteed to discover those reports and learn that new relevant information has been released.

Personalized recommendation is an essential component of the TLDR. It is difficult to build a personalized system without knowing **what** needs to be personalized. The study interviews are part of an essential user research phase that is often overlooked in system design. These findings will help to ground the requirements more directly in needs of analysts as primary users of the envisioned TLDR system. Specifically, we expect to be able to extract dimensions or factors of personalization for a TLDR system from the interview findings that align with the analysts' actual needs. For example, preliminary findings indicate that analysts work requests come in different formats (e.g, email, physical walk-ins), each with its own set of corresponding triggers. One factor for personalization to consider in the TLDR, then, may be the presentation format of work requests based on these varied triggers. This in turn informs the types of automation and processing that need to happen for that TLDR.

This research provides an initial step towards a framework of information flow to inform reporting needs, organize expectations for report content, and identify practical considerations for different ways tailored reports might best be utilized within analysts' workflows. The results, though preliminary at this time, are expected help prioritize algorithmic capabilities for future research and development activities and design of human-interface aspects of systems to allow integration in current work environments. Through future analysis of the study data, we expect to shape and prioritize research questions and define requirements for prototype TLDR systems, and gain more insights how personalization should be incorporated into a TLDR.

3 Identifying Content for a TLDR

A critical component of the TLDR system is identifying which content to include in a TLDR for a given user. The content must be relevant to a user's interests and be significant enough to warrant inclusion in the TLDR. It is not always practical to have the user explicitly state what they find interesting or what kind of information should be deemed significant. Users might not know exactly what kind of content they are looking for, or might not provide clear or extensive information on what is relevant to them. Moreover, user's interests or threshold for significance might change over time or be context-dependent. This suggests that a TLDR system must be able to infer what is relevant and noteworthy, and then explain to the user why certain content is being recommended or not. To better understand possible mechanisms for identifying TLDR content, we investigated the following questions:

1. What is the impact to recommender model performance when training models with less data than in top-performing scenarios?
2. Is natural language inference an effective mechanism for filtering redundant information from recommender model output?
3. What are effective methods for explaining predictions generated by neural recommender models?
4. What are effective methods for detecting anomalies in tabular data and knowledge graphs?

3.1 Generating Recommendations Based on User Feedback

Recommendation systems are filtering techniques that provide personalized item recommendations based on each user's feedback history. The system can learn from "explicit" user feedback (e.g., thumbs up, star ratings) and/or "implicit" user feedback (e.g., URL clicks, dwell time). For the TLDR, we envision a recommendation system that filters the information space according to each user's interests, reducing a large corpus of documents to a small personalized set. This small set of documents will then be summarized and aggregated to produce the TLDR.

3.1.1 Data

An aspect of the TLDR system is that the pool of candidate documents for recommendation continuously changes. This differs from many familiar recommender systems which operate on a fairly stable set of items (e.g., movie recommendations), but is similar to news platforms that recommend articles from an ever changing catalog. Thus, we chose to use the MIcrosoft News Dataset (MIND) as a proxy for the TLDR. This data set was collected from the MSN News platform during October and November 2019 and contains information about user behavior when interacting with news articles, metadata about the news articles themselves, and entity and relation embeddings[3] that correspond to the articles [WQC+20]. The first four weeks of data were used to produce user histories (articles that users clicked on during that time period). The next six days were used as training data followed by a day of validation data, and seven days of test data (see Figure 2).

There are three versions of the MIND data: large, small, and demo. The large data set contains 1 million users sampled from the MSN News platform. The small data set contains 50,000 users sampled from the large data set. The demo data set is the smallest version and is provided for quickly testing code. Overall, the MIND data contains observations regarding approximately 160,000 English-language news articles.

For this work, we used the MIND small data set, as we reasoned that the number of users (50,000) in the small data set would best match the scale of users serviced by our future TLDR system compared to the number in the large data set (1 million). Additionally, model training and inference times increase as the number of data points increases, so working with a smaller data set made the task more tractable given compute resource limitations. We only utilized the user behaviors and the article titles in this iteration, and deferred incorporating additional article metadata, text, and related embeddings to future work.

[3] An embedding is a low-dimensional continuous vector representation of discrete data [MCCD13].

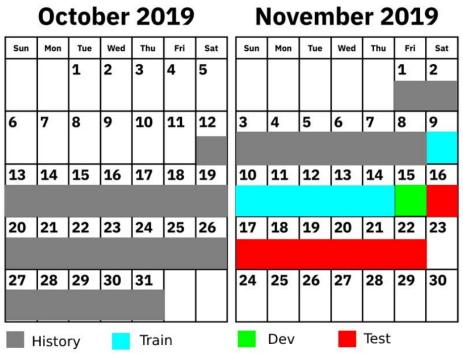


Figure 2: Original MIND Data Splits

Microsoft did not provide a test set for MIND small, so we created our own data split as shown in Table 1. The table shows for each data split: the dates over which each split was collected; the number of days the split spanned; and the percent of total impressions of the MIND small data set represented in the split. The union of our three data splits contains 65,000+ English news articles and 230,000+ impression logs, which contained the user behavior when presented with a recommended article. Comparing the test split to the union of the train and validation: each has 50,000 users, however, only 5,943 users overlap between the two. This differs from MIND large where 78% of the users in the test set were seen in train and/or validation. This could result in lower scores for our model than those trained on the large data set as the model did not have the opportunity to learn preferences for a large portion of the users in the test set. In [WQC+20] models performed better on test data containing users that overlapped with the training than on test data containing unseen users in almost all cases.

Data	Dates	Days	% of Total Impressions
Train	Nov 9-13	5	55
Validation	Nov 14	1	13
Test	Nov 15	1	32

Table 1: Data Splits Used for SCADS 2022

3.1.2 Models

For our proof-of-concept we implemented[4] the Neural News Recommendation with multi-head Self-attention (NRMS) model [WWG+19], which was the top-scoring model from the MIND News Recommendation Competition [WQC+20] based on area under the curve (AUC), mean reciprocal rank (MRR), and normalized discounted cumulative gain (NDCG) accuracy metrics. For building the recommendation engine, we utilized the `recommenders` Python module version 1.1.1 with Python version 3.8.10 and TensorFlow version 2.8.0. We also experimented with the Deep Knowledge-aware Network (DKN) model that can incorporate knowledge graphs into recommendations [WZXG18].

NRMS The NRMS model architecture is shown in Figure 3. Titles associated with user histories are input as "browsed news," and titles associated with articles in the impression list are input as "candidate news." Key components of the model include:

- **News encoder**: Composed of multi-head self attention layers where the input vectors are GloVe embeddings [PSM14] of the first M words of the news title. This component captures the importance of word pair interactions in the title for purposes of predicting click probability. A vector, $\mathbf{r}$, is produced for each news article.

[4] https://github.ncsu.edu/SCADS/RecSystemModels

- **User encoder**: Composed of multi-head self attention layers where the input vectors come from the news encoder for each "browsed" article (i.e., user history). This component captures the importance of article pair interactions. A vector, $\mathbf{u}$, is produced for each user.
- **Click prediction**: The probability that a user will click on a candidate news article is computed using the dot product of $\mathbf{r}$ and $\mathbf{u}$. There is a subtle difference in computation of the click probability for model training versus scoring:
  - **Training**: The problem is re-formulated as a $(K+1)$-way classification task where K is the number of negative (not-clicked) news samples (a hyperparameter) from the impression list. The positive (clicked) news article from the impression list is also considered resulting in $K+1$ classes. Let $\hat{y}_i^+ = \mathbf{u}^T\mathbf{r}_i^+$ where $\mathbf{u}$ is the user representation vector and $\mathbf{r}_i^+$ is the news article representation vector for the positive (news article) in the i^{th} impression list. Similarly, let $\hat{y}_{ij}^- = \mathbf{u}^T\mathbf{r}_{ij}^-$ be the j^{th} negative news article in the i^{th} impression list (for $j = 1..K$). Then the click probability of a positive article is computed via the softmax function:

$$p_i = \frac{e^{\hat{y}_i^+}}{e^{\hat{y}_i^+} + \sum_{j=1}^{K} e^{\hat{y}_{ij}^-}}$$

    As stated in [WWG+19], the authors define the loss function as the negative log-likelihood of all positive samples:

$$\mathcal{L}_{NRMS} = -\sum_{i \in \mathcal{S}} \log(p_i)$$

    where $\mathcal{S}$ is the set of all positive items. However, the software implementation (discussed in Appendix E) uses cross entropy loss, which considers all samples.
  - **Scoring**: Instead of $K+1$ candidate input articles, one candidate news article is scored using the dot product of the associated $\mathbf{r}$ and $\mathbf{u}$ vectors as input to a sigmoid activation function (a special case of the $softmax$ function for two classes). This is shown in Figure 3.

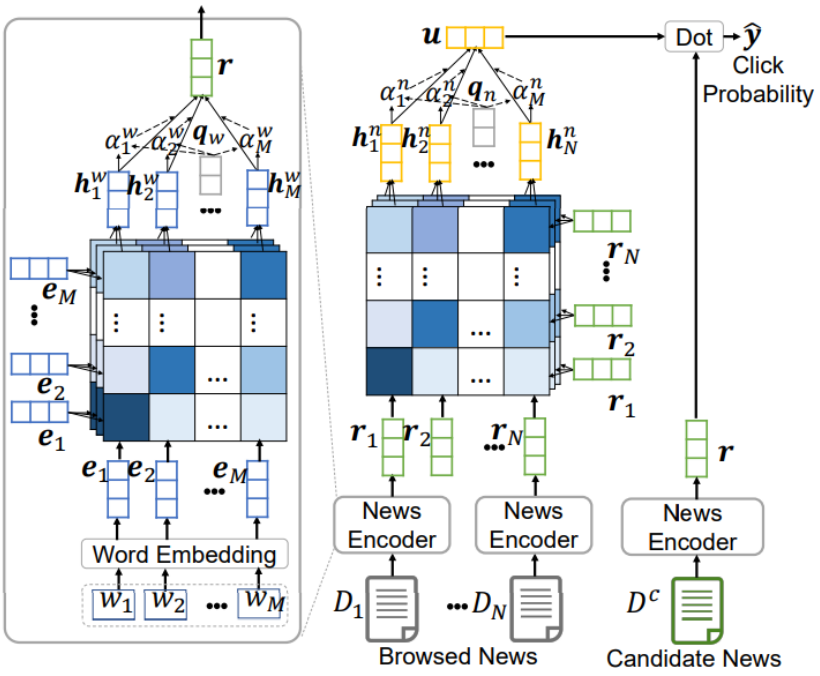


Figure 3: NRMS Model Scoring Architecture. Source: [WWG+19]

DKN The Deep Knowledge-Aware Network (DKN) model nicely demonstrates how a knowledge graph can be incorporated into a recommender system for improved recommendations. The authors of [WZXG18] give the following motivational example:

> Article 1: "Boris Johnson Has Warned Donald Trump To Stick To The Iran Nuclear Deal"
>
> Article 2: "North Korean EMP Attack Would Cause Mass U.S. Starvation, Says Congressional Report"

The authors explain that the two articles share contextual information and that a user interested in the first may be interested in the second with high probability, since both articles address the potential impacts of political actions. However, traditional natural language processing techniques (e.g., topic models, semantic models) have a difficult time identifying the relationship between the two articles because depsite their **semantic** similarity, they do not have much **lexicographic** similarity. Thus, they proposed adding additional knowledge graph information.

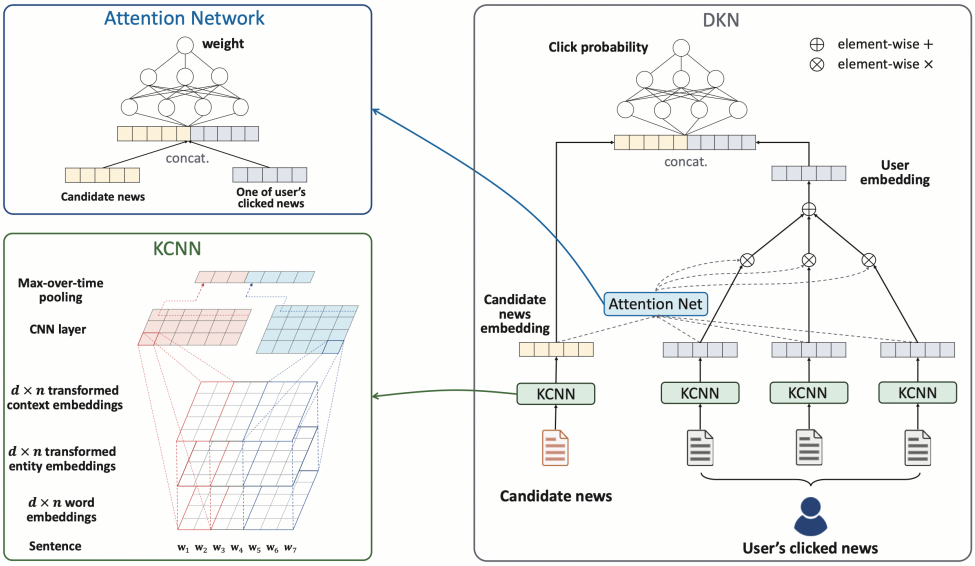


Figure 4: DKN Framework. Source: [WZXG18]

The DKN model framework, shown in Figure 4, takes a candidate article and a user's history as input. Similar to NRMS, [WZXG18] demonstrates the use of article titles; however, the system is generalizable to other text such as abstract or body. Each text is fed into a (multi-channel) knowledge-aware convolutional neural network (KCNN). The stacked channels, shown in the lower left of Figure 4, include:

- **Word embeddings**: Each word of the input text is represented by an embedding vector, which can be pre-trained or randomly initialized.
- **Entity embeddings**: Learned via knowledge graph embedding methods (e.g., TransE, TransH, TransR, TransD) and depicted in Figure 5. The displayed knowledge graph is a subgraph of one much larger such as Google Knowledge Graph or Microsoft Satori. Words in the input vector that are not linked to an entity in the knowledge graph are represented by the zero vector.
- **Contextual entity embeddings**: The context of a target entity, as shown in Figure 6, is composed of the surrounding entities in the knowledge graph. The contextual entity embedding vector of the target entity is the average of the embeddings for the entities in that context.

The news embeddings resulting from the KCNNs are fed into an attention network, displayed in the upper left of Figure 4, to determine the importance of the user's clicked news in regard to the candidate news article. The outcome is a weighted sum of the user's clicked news title embeddings, which is then concatentated with the candidate news embedding and fed into a deep neural net (DNN). Finally, a sigmoid activation function is applied to obtain the probability that the user will click on the candidate news article. Log loss is used to train the model.

Model Training We used the optimal hyperparameters for NRMS and DKN given in [WWG+19] and [WZXG18] and selected the optimal number of epochs using our validation set. The validation loss increased after just one epoch for both models, which may indicate that the models would benefit from further tuning. The specific hyperparameters used can be found in our GitHub repository. We used an AWS g5.2x large EC2 instance for training the models. Training times are shown in Table 2. NRMS model inferencetook 23.5 hours to score 5,369 candidate articles for each of the 50,000 users using a Jupyter notebook on a p3.2xlarge AWS EC2 instance. The candidate article scoring did not finish in time to allow for calculation of all DKN model performance metrics to be included in this report. We utilized a notebook to run the models to generate output with which we could compute metrics, as the model storage effort described in Section 3.1.2 was conducted in parallel with model training and article scoring.

Model Storage and Serving For our proof of concept pipeline, we deployed the model to Bailo, a model storage and serving capability being developed by Government Communications Headquarters (GCHQ) and in early stages of testing at LAS. Bailo provides a model repository to store and share models, and a mechanism for deploying a Docker container that enables model inference via API endpoints. We coded the Bailo `Model predict` function and associated test functions to be compatible with a TensorFlow `SavedModel` and multidimensional array data, as required by the recommender model described above. In developing our proof of concept pipeline we identified and supported improvements to the recommender model and Bailo implementation and documentation. See Appendix F for additional details.

3.1.3 Performance

To understand how the models performed, we computed multiple metrics using the test data split (see Section 3.1.1). The metrics fell into two categories, referred to as *Accuracy* and *Beyond Accuracy*. Accuracy metrics provide insight into how well the models perform on the task of identifying content a user might be interested in based on that user's history, while Beyond Accuracy metrics provide insight into aspects of model performance that might relate to other goals, like minimizing redundancy in recommendations and identifying recommendations from a variety of sources [RD22].

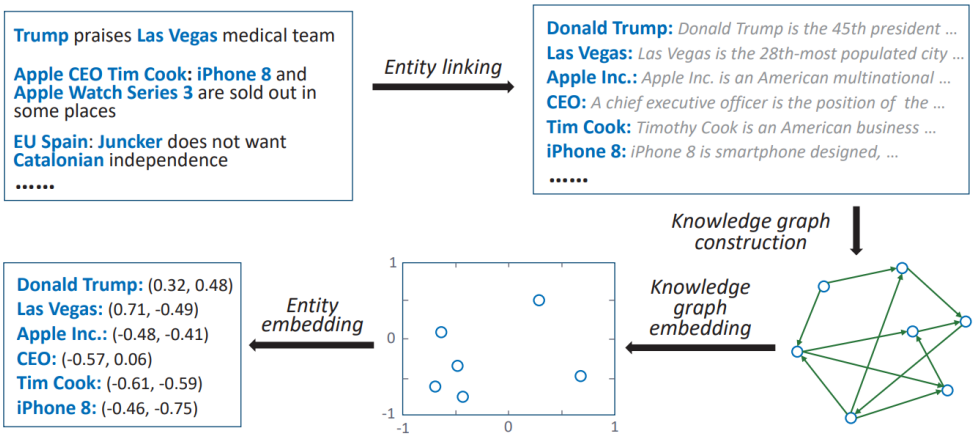


Figure 5: Knowledge Extraction. Source: [WZXG18]

Model	NDCG@5	NDCG@10	AUC	MRR	Train Time
NRMS [SCADS]	0.3081	0.3745	0.6211	0.2822	10 min
DKN [SCADS]	0.2721	0.3408	0.6299	0.2522	12 min
NRMS [WQC+20]	0.3594	0.4163	0.6776	0.3305	*n.r.*
DKN [WQC+20]	0.3384	0.3948	0.6460	0.3132	*n.r.*

Table 2: Performance metrics of NRMS and DKN models generated during SCADS. Scores for comparable models reported in [WQC+20] included for comparison.

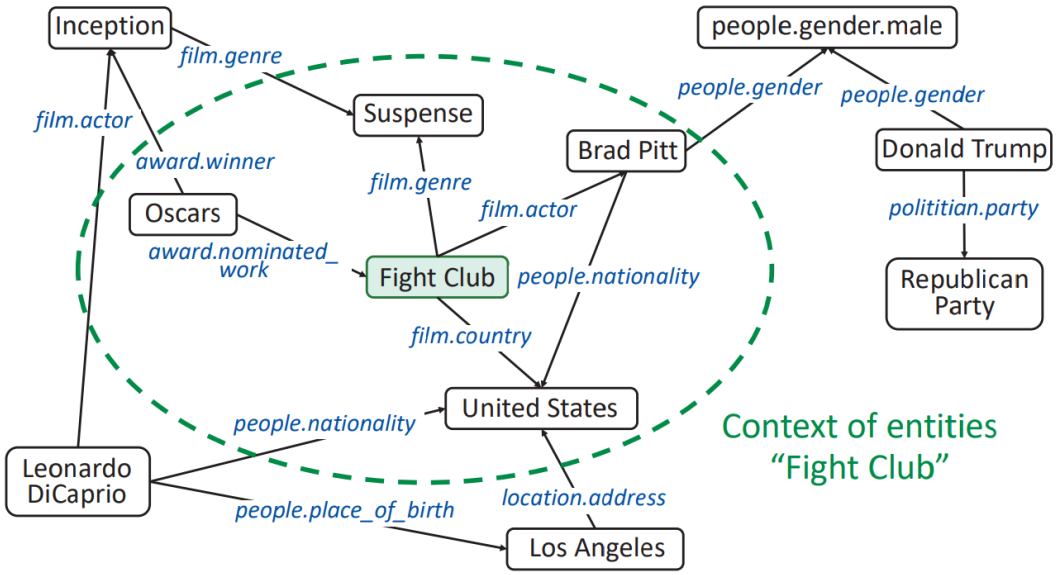


Figure 6: Entity Context in a Knowledge Graph. Source: [WZXG18]

Accuracy We evaluated the NRMS and DKN models by calculating the same metrics displayed on the MIND News Recommendation Competition leaderboard: AUC, MRR, and NDCG. The NDCG metric considered only the top-k (5 or 10) scoring articles in the test set for each user, averaged over all users.

We compared the NRMS accuracy metrics here to those reported in [WQC+20], our implementation scored slightly lower than the competition-winning model, which was expected since we used the smaller data set. The lower scores may also be due to less user overlap between the train and test sets compared to that of the large data set. In [WZXG18], DKN was trained on Bing News, so we could not directly compare our implementation to that of the authors'. Our DKN implementation scored slightly lower than a DKN model trained on the MIND large training set in an experiment by [WQC+20]. While our NRMS implementation had slightly higher NDCG and MRR scores than our DKN implementation, the performance difference is likely so small that a TLDR user would not notice.

Beyond Accuracy For the NRMS model, we also calculated beyond accuracy metrics of diversity, novelty, and catalog coverage, which have been used to mitigate known issues with solely considering accuracy metrics when selecting a recommendation system [RD22]. For example, beyond accuracy metrics could provide additional insight into a system with high test accuracy but low user satisfaction ratings if the system presents the user with a non-diverse set of products. The beyond accuracy metrics also consider only the top-k recommendations for each user and average over all users. To produce the top-k recommendations for users we scored a set of candidate articles, which we derived from the test data by compiling the list of unique articles seen in all user impressions on this day. This produced a list of 5,369 articles that we scored for each user to determine the top-k articles. See Appendix G for further explanation of beyond accuracy metrics.

Model	Diversity@10	Novelty@10	Catalog Coverage@10
NRMS	0.8763	0.8009	0.1144
DKN	NC	NC	NC

Table 3: Beyond Accuracy Metrics (NC = Not Calculated)

The NRMS model diversity was 0.8763, meaning that almost 88% of the top 10 recommendations for users were considered to be dissimilar to other recommendations. Novelty was 0.8009, or about 80% of the top 10 recommendations for users were unique to the user. Catalog coverage was 0.1144, so approximately 11% of the article catalog was included in the top 10 recommendations.

3.1.4 Natural Language Inference

We explored the use of natural language inference (NLI) as way to determine if titles recommended by the NRMS model provide the user repetitive information compared to titles the user has already viewed. NLI is the "task of determining whether a 'hypothesis' is true (entailment), false (contradiction), or undetermined (neutral) given a 'premise' [Rud22]." If the hypothesis can be *inferred* from the premise, then the premise *entails* the hypothesis. In the context of a recommending content for a TLDR, we could leverage NLI to assess whether a previously-viewed title (premise) entails a recommended title (hypothesis), in which case the recommended title could be considered repetitive information to the user. Direction matters for entailment. We cannot assume that phrase 1 (premise) entails phrase 2 (hypothesis) if phrase 2 entails phrase 1. This is why we considered prior knowledge as the premise and new knowledge as the hypothesis. We might tailor recommendations in a TLDR based on whether NLI indicates that recommended content supports (entails) or contradicts content that the user has previously viewed. Table 4 shows example phrases and their entailment relationship.

Premise (Phrase 1)	Hypothesis (Phrase 2)	Relationship
The woman eats a salad containing apples for lunch	The woman has some fruit for lunch.	Entailment
The woman eats a salad containing apples for lunch	The woman skips lunch.	Contradiction
The woman eats a salad containing apples for lunch	The woman quickly eats a salad containing apples for lunch	Neutral
The woman eats a salad containing apples for lunch	The woman is fluent in French and English.	Neutral

Table 4: Entailment Examples

We used a pre-trained NLI model from Keras [Mer20] to predict the entailment relationship between two article titles. The Keras model was based on Bidirectional Encoder Representations from Transformers (BERT) [Mul22] and was trained on the Stanford Natural Language Inference (SNLI) Corpus [BAPM15]. The NLI model required a premise phrase and hypothesis phrase as input, for which we used each of the article titles in a user's interaction history as a premise and each top k candidate article titles as recommended by the NRMS model for that user as a hypothesis. To minimize the number of pairs to be scored and reduce compute time, we used a subset of the data that included only those premise-hypothesis pairs for which the articles have the same category and subcategory, or approximately 908,000 pairs of titles. There is some duplication in the premise-hypothesis pairs in cases where users had overlapping viewing history and recommendations. The NLI model was case insensitive, so we converted all applicable title pairs to lower case and encoded them using a class from [Mer20]. The NLI model output a confidence score between 0 and 1 for each possible label (entailment, contradiction, neutral) for a premise-hypothesis pair, and we used the label with the highest score as the prediction for that pair.

Table 5 shows the breakdown of entailment predictions on the subset. Less than 1% of the title pairs were labeled as entailment, which was expected based on manual inspection of the titles in the MIND data.

Label	Percent of Pairs
Contradiction	51.4595
Entailment	00.6660
Neutral	47.8745

Table 5: NLI Results

We did not have ground truth labels for NLI on the NRMS recommendations, so we were not able to calculate accuracy metrics for the NLI model during SCADS. We inspected the results manually to understand how the model performed and determine next steps for incorporating NLI into TLDR creation. Table 6 shows a sample of premise-hypothesis pairs from the data subset with the NLI model output for that pair.

Ex.	Viewed Title	Recommended Title	Prediction	Confidence
1	NFL Cheerleaders	NFL Cheerleaders	Entailment	0.9392
2	Every outfit Duchess Kate has worn in 2019	See every outfit Duchess Meghan wore in 2019	Entailment	0.7318
3	85 Easy Christmas Appetizers That Will Delight All of Your Holiday Party Guests	55 Christmas Treats and Sweets Perfect for All Your Holiday Parties	Entailment	0.5917
4	Rosie O'Donnell: Barbara Walters Isn't "Up to Speaking to People" Right Now	Whoopi Goldberg Addresses "The View" Tension: "If We Were Fighting, You'd Actually Know It"	Contradiction	0.9383
5	Sex offenders sue sheriff who put "no trick-or-treat" signs in their front yards	Missouri woman charged after husband's body found in freezer	Contradiction	0.7322
6	Deer fatally attacks hunter who shot him	Police: Man Arrested After Allegedly Admitting To Tying Up, Killing Missing Bethel Park Woman	Neutral	0.6526
7	Police: Off-Duty Dallas Officer Mistakenly Shoots, Injures Adult Son Thinking He Was An Intruder	Police: Man Arrested After Allegedly Admitting To Tying Up, Killing Missing Bethel Park Woman	Neutral	0.8968

Table 6: NLI Predictions

We observed that quite a few of the pairs predicted as contradiction should have been labeled as neutral, as in Example 5. There were also some instances of pairs predicted as entailment that should also have been neutral, as in Example 2. Those examples have similar prediction scores, so in future work we would like to investigate the distribution of scores to determine a threshold with which we could categorize high- and low-confidence predictions. The examples show interesting behavior around scores, entities, and overlap between the tokens in the two titles. During our manual inspection, we noted that the prediction scores tend to be higher with more overlapping tokens in the viewed and recommended titles, as in Examples 1 and 7. It also seems that the entities mentioned in the titles influence the NLI prediction, as in Examples 2 and 4. Further investigation is needed, however, to better understand the NLI model behavior and implications for the TLDR.

3.1.5 Explaining Recommendations

With any implementation of new-to-the-user automation techniques comes a substantial hurdle – skepticism. Augmenting black box models with interpretable reasoning helps assess how well or how poorly the model is making decisions. These explanations help to develop trust between effective models and end users over time. This section describes our work using Local Interpretable Model-agnostic Explanations (LIME) to aid in model explainability.

Our recommendation system was designed to assign a score $s \in [0, 1]$ to an article based on its similarity to articles a user has viewed in the past – we get a more confident (i.e., "better") recommendation the farther s is from 0.5. We employed LIME [RSG16] for post-hoc explainability of our recommendations. Further exploration in future SCADS cohorts could prove fruitful, if implementation is not too computationally expensive. The ultimate goal of LIME is to estimate how much each word in the article contributes to or takes away from the recommendation. LIME treats the NRMS model as a black box and provides a faithful and interpretable local linear approximation of the model's behavior. LIME reaches an explanation by minimizing unfaithfulness and complexity:

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \, \mathcal{L}(f, g, \pi_x) + \Omega(g). \tag{1}$$

where

- G is the set of interpretable models over the absence/presence of interpretable components,
- $\mathcal{L}$ is a function measuring unfaithfulness,
- f is our black box model,
- π_x is how close of an approximation g is to f, and
- Ω measures complexity of the approximation.

In the context of our model, the interpretable components were the words in an article title and ξ represented how much positive, negative, or no impact each word had on user interest. For higher values of s, more words in the title indicated that a user would be interested in reading the article while titles with few words indicating user interest would have lower values of s. It is important to note, however, that for low values of s, two potential causes exist for failing to recommend:

- **Disinterest:** A subset of the title's words deal with topics the user is not interested in.
- **Irrelevance:** None of the words in our article title exhibit qualities that the user is either interested or disinterested in.

If s was very close to 0.5 we were likely to see a mixture of words, some weighted significantly in the positive direction and others in the negative. These contradictions yielded inconclusivity and, thus, failed to give a confident recommendation one way or the other.

We implemented the LIME explanation function to take a single article title as input, tokenize the title, and score each token against our NRMS model. This gave us an idea of the relative importance of each word to the title, given a user's history and the overall space of article titles in our data set. We looked at some examples from the MIND data set in which the NRMS model recommended or did not recommend the title with high confidence to the user. In these examples, the user had a full, 50-article click history where all of the viewed articles pertained to the National Football League (NFL). Table 7 contains the article titles we examined with LIME and the corresponding NRMS output.

Ex.	Article Title	Prediction	Prediction Score
1	Packers wide receiver Davante Adams to miss Monday's game with toe injury.	Recommend	0.88
2	LA Measles Patient Visited Disneyland, Starbucks While Infectious.	Don't Recommend	0.96

Table 7: NRMS Output Used in LIME Examples

We visualized the recommendation explanations with the `LimeTextExplainer.explain_instance` function found in the `lime_text` module. Example 1 shows a title that was recommended to our user with a score of 0.88, indicating with high confidence the user would be interested in reading the article. Figure 7 (top) provides a breakdown of the six most impactful words in the title. It made sense to see *Packers* (an NFL team in Wisconsin), *receiver*, *injury*, and *game* contributing to the recommendation because of their relevance to the game of football. On the other hand, it was interesting to see that *Adams* had a positive impact on this recommendation. We also saw that *toe* took away from the overall recommendation score.

At face-value, we can see the title in Example 2 likely has nothing to do with professional sports. Figure 7 (bottom) shows a component-wise breakdown of why this article was not recommended to the user. None of the words contributed significantly for or against the recommendation, especially when compared to the impact of *Packers* in the previous example. It made sense to see that *Measles* and *Infectious* steered the user away from this article given the expected lack of overlap in epidemiology and sports terminology. The token *LA* likely contributed positively toward recommend because there are two NFL teams in Los Angeles, CA.

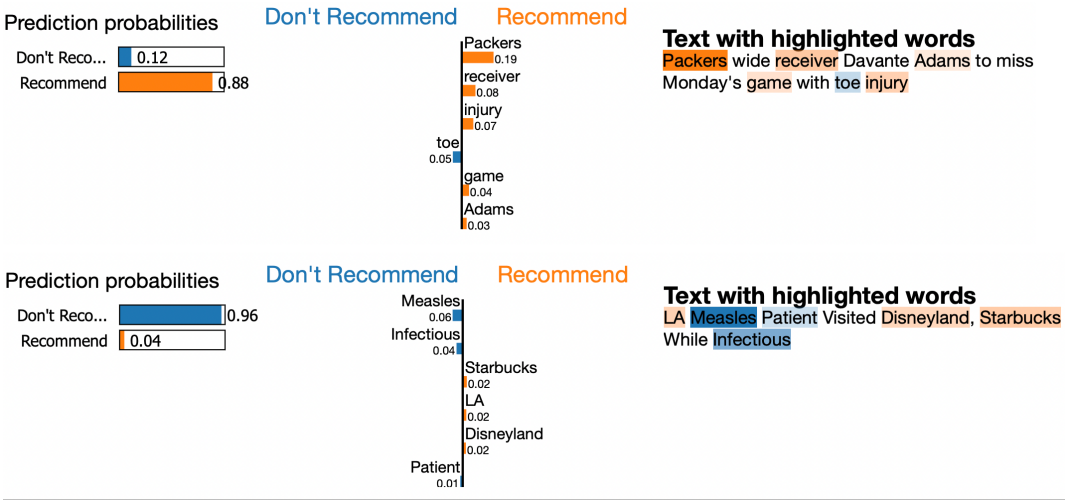


Figure 7: LIME Explanation for a Recommended (top) and Not Recommended (bottom) Article

3.1.6 Discussion

The recommender models we implemented were trained on article titles and user impressions, which may produce "click bait" (i.e., the titles may be tantalizing, but the content may or may not be interesting to the user). However, both of these models generalize to any type of text input, such as the article body, and can serve as building blocks to be adapted to specific TLDR system needs. In future work, we would recommend re-running these experiments on full text articles.

We computed various metrics to assess recommender model performance. We were not able to compare our models' performance on any metric to other model performance in comparable use cases, as the limited amount of literature on similar models' performance focused primarily on retail use cases. Our models' performance on common metrics for comparing recommender model effectiveness, such as NDCG and AUC, was slightly lower than the original models' performance in the Microsoft MIND competition. This was expected given the smaller training data set used here. We computed beyond accuracy metrics, such as diversity and catalog coverage, and found that overall the recommendations were distinct and users were shown a small portion of the available data. Further analysis is required, however, to understand the implications of this finding in the context of a TLDR or how this information could be leveraged to improve model performance. Similarly, we implemented an NLI model and produced NLI predictions for recommendations from the NRMS model, but need to further assess how best to incorporate NLI information into a TLDR. For example, a TLDR interface might incorporate beyond accuracy and NLI information to provide the user with additional details that supplement the model's recommendations.

LIME proved to be a powerful tool in both utility and visualization. It could be valuable to incorporate LIME into a TLDR system, such as to explain to a user why the TLDR included certain content that was identified by some underlying recommender model. LIME could also be applied to other types of models to provide explanations of their output, however it would be important to validate LIME's utility for explaining different models.

The recommender system implemented during SCADS requires data about what types of information the user is interacting with in order to predict new content that user might be interested in. So, a recommender system would need to be applied in a use case in which the user interacts with a set of information that's presented to the user, such as a news feed. One potential TLDR use case discussed throughout SCADS incorporated a manager who is regularly inundated with emails trying to prioritize which emails to read. A possible near-term application of a NRMS-style recommendation system could be to recommend which messages the user should read, in effect establishing a prioritized queue. The user would also have access to other metadata associated with emails, or "side information," such as sender, priority flags, and so on, to incorporate into their decision making. NLI information and LIME explanations could also be provided as side information as one method to help users combat the potential creation of feedback loops as they interact with the TLDR system.

3.2 Identifying Content Based on Changes in Source Data

In conversations with knowledge workers, one frequently-requested feature of a TLDR was to alert users to changes in data that the user interacts with on a regular basis. We envision a mechanism for identifying changes in a data source, then subsequent processes that summarize the change or generate a visualization to include in a user's TLDR. We began by considering "changes to the data" as anomalies in the underlying data, and explored different approaches to anomaly detection to assess which might be effective at identifying changes that should be presented in a TLDR.

3.2.1 Background and Related Work

Anomaly detection is a widely studied area and many techniques exist for detecting anomalies in both tabular and graphical data sets. Typically, anomaly detection models are trained to flag as anomalous any aspect of the input data that lies outside of the norm [MWX+21]. One shortcoming of conventional anomaly detection methods is that they can often misidentify complex relationships, which is especially true in temporal graphs where nodes and edges change over time [MWX+21, XCL+20]. A further issue is that graph databases are often very large, and in many cases lack ground truth for evaluation purposes [MWX+21]. Lastly, the high-dimensionality of real-world data acts as a constraint on many anomaly detection methods, which has given rise to the use of deep learning anomaly detection models [RVG+18].

Many anomaly detection models for both dynamic graphs and temporal knowledge graphs are supervised or semi-supervised learning models [ZLL+19, CCL+20]. Supervised learning requires labeled data, which can be very labor-intensive to curate, especially for anomaly detection applications where anomalies are such a small proportion of the data. One technique to address the lack of labeled anomalous data is to take an existing dataset and inject anomalies [MWX+21]. In this approach, existing data is labeled as "normal" and the injected data is considered "anomalous." However, validating that the background data contains no anomalies is also resource-intensive, and so is often overlooked. Moreover, contradictory information is also a valuable type of anomaly to detect, but such contradictions would not have an associated label in a data set with solely injected anomalies. Therefore, conventional supervised learning models may learn how to find human injected anomalies, but not anomalies inherent to the data itself. For these reasons, we decided to focus our attention on unsupervised learning models, which is at present a more promising area of research.

We reviewed existing work applying anomaly detection methods in the cyber domain to identify malicious behavior or actors within computer network data. In [AIH21], researchers explored the DARPA OpTC data set in the context of intrusion detection. This work is further extended in [Vea21], in which the OpTC data set is converted from the original PDF data into a machine-readable tabular format. The tabular version of the OpTC data was then used to explore the accuracy of classification models in identifying malicious actors. Such work could be applicable in the context of a cyber analyst whose TLDR alerts them to potential network intrusions that need to be investigated further.

Some anomaly detection methods specifically deal with data that evolves over time, such as temporal knowledge graphs. For example, Adversarial Multiscale Anomaly Detection (AMAD) leverages large differences in residual values between timestamps within a graph to identify anomalies with some success [XCL+20, GGM+19]. This approach only applies to individual nodes, however, which might limit its utility in contexts for which community or group level anomalies are of interest. DeltaCon compares the edge sets for each node in a graph across different versions of the graph over time [KVF13] by computing a similarity score between different versions, which is then used to predict where and when anomalies appear within the graph. DeltaCon-ATTR is an extended version of DeltaCon that allows for identification of which nodes and edges changed within the graph [KSV+16]. However, both DeltaCon versions utilize node and edge counts to identify anomalies, which unfortunately does not account for changes to semantic information over time.

Many anomaly detection algorithms focus on either finding a time that is anomalous (considered "case" or "warning" anomalies) or finding an anomalous individual node/edge or sub-graph occurring within a system (nested anomalies), but not both. Additionally, many methods used to detect anomalies within graphs were not specifically developed for knowledge graphs, and therefore do not explicitly take advantage of the complex, relational data they contain. Although some anomaly detection methods operate specifically on knowledge graphs, most do not scale well or require an intensive amount of human involvement, making them unsuitable for real-world application [SOWC21].

Our investigation into existing anomaly detection methods uncovered two notable exceptions to the aforementioned limitations. The DeepSphere algorithm from [TYEL18] attempts to do both case anomaly detection and nested anomaly detection, while [SOWC21] attempts to incorporate semantic information from a knowledge graph in an unsupervised anomaly detection approach. We explore both efforts further in several mission-relevant contexts: cybersecurity, fraud detection, and communication networks.

3.2.2 Data

OpTC The Operationally Transparent Cyber data set is the publicly-available result of a red-blue team exercise on a 1000-host network equipped with various sensors conducted by the Defense Advanced Research Projects Agency (DARPA) as part of the Cyber Hunting at Scale (CHASE) program [AIH21]. The exercise took place over seven days, during which there were three days of benign traffic events and three days of traffic events which contain documented red team activities.

The OpTC data set consists of event and sensor logs that amount to approximately 1,100 gigabytes of data in compressed JSON format [AWHK21]. The data is divided into three folders, `ecar`, `bro`, and `ecar-bro`. The `ecar` and `ecar-bro` folders are divided further into `benign`, `evaluation`, and `short` folders, while the `bro` folder is divided into subfolders for each day of the exercise. The `ecar` data uses extended Cyber Analytics Repository (eCAR) format, which was developed for the OpTC data set by the company FiveDirections and was based on MITRE's Cyber Analytics Repository (CAR) format. The `bro` data is the data collected from *bro* sensors, now known as Zeek sensors. The `ecar-bro` data consists of specific events in the `ecar` data annotated with information from the bro sensors.

The CAR format consists of objects, actions, and fields, which eCAR extends to include principal and actor attributes. Object values refer to a component in the system, such as a host, file, or connection. Action values refer to something that happens to the object. Field values refer to some attribute of the object. Including actor information enables linking the red-team diary and the **[head, relation, tail]** triple provided by [actor, action, object]. The data set includes timestamp information for each data point and a unique identifier for each attribute.

One technical challenge surrounding the OpTC data set is that it is extremely imbalanced, with less than 0.01% of the data representing malicious activity. While this characteristic is good in that it is representative of network activity in the real world, it is a challenge to implement classification models on data with a dramatic class imbalance. Because the data set is so large it is helpful to work with data samples, but uniformly-drawn samples might not contain sufficient examples of malicious events to train machine learning models to detect such behavior.

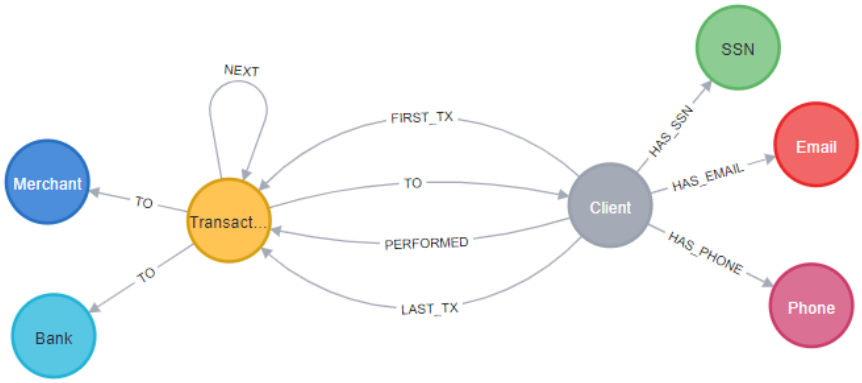


Figure 8: The PAYSIM Data Model. Source: [RHB]

PAYSIM The Neo4j fraud detection knowledge graph based on the PAYSIM data set [LREA16] is a synthetic knowledge graph describing transactional and fraudulent behavior, which is available through the Neo4j sandbox. This data set has an established ground truth that can be used in evaluation, and a defined ontology to support feature engineering. A version of this data set is also available on Kaggle, a publicly-available platform for hosting data science competitions by providing a challenge,

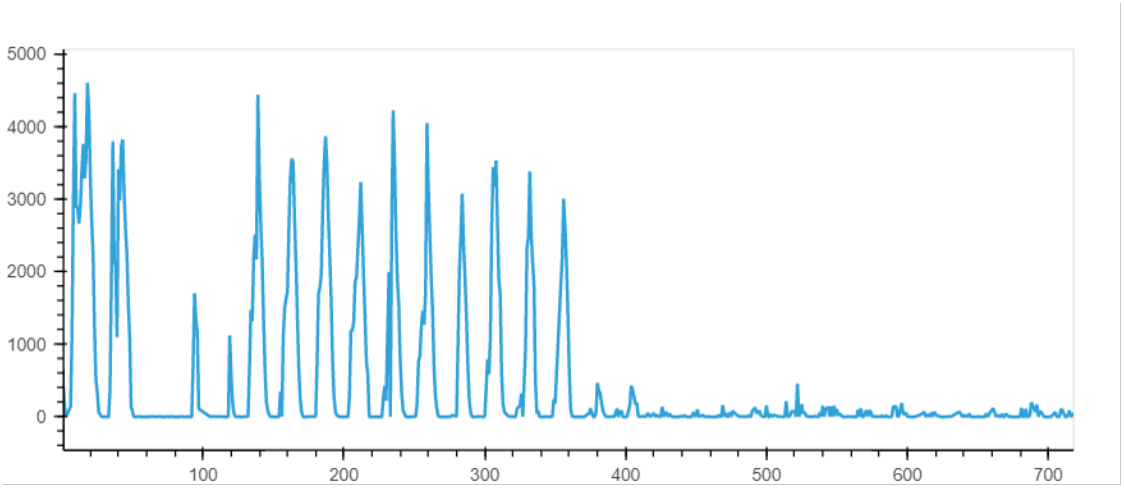


Figure 9: PAYSIM transaction count by hour elapsed

data set, and related code. Users build upon existing resources and submit results to be displayed on a leaderboard, so there is an existing body of results to which we can compare our efforts.

Figure 8 shows the underlying data model. The data set contains the node types *Transaction* (e.g. Payment, Transfer), *Agent* (e.g. Banks, Merchants, and Clients), and *Identifier* (e.g. Phone, Social Security Number). Each node may have additional *Property* information, such as a score indicating the potential that a *Identifier* is fraudulent. The edges between nodes denote the *Relationships* between those nodes, such as '*has_phone*' or '*first_tx*.'

For compatibility with the DeepSphere model, we created adjacency matrices using the amount of data transferred per hour between entities as the matrix values. We used only the client, banks, and merchant nodes and dropped the "middle man" transaction nodes that appeared between any two entities involved in a money transfer activity. Dropping some node types resulted in smaller adjacency matrices that were easier to use in experiments. We grouped the data by day for a total of 15 days covered by the data set. Figure 9 shows the distribution of transaction counts within the data set. Time is represented as hours elapsed from the initial timestamp. We only worked with data from timestamp 1 to timestamp 361, as it was clear that transaction patterns changed drastically after time 361. For this work we limited the adjacency matrices to include only the first 1000 nodes within the data set, as we noted that this portion of the data set contained more connections between nodes. This also helped to reduce the complexity and resource requirements needed for experiments. We standardized the data with `scikit-learn`'s `StandardScaler` before passing it to DeepSphere.

UCI Message Data The University of California Irvine (UCI) message data set is a dynamic graph data set that represents messages sent between members of an online student community at the University of California, Irvine [OP09]. This data is anonymized, using numerical identifiers to label graph components, and contains no information regarding the content of the messages that were sent. The data set creator applied this data when studying communication within social networks, which is similar to potential use cases in which a TLDR could be helpful. The data represents 27 weeks of messages, and consists of nodes corresponding to 1,899 users and directed edges corresponding to 13,838 messages. Figure 10 shows the distribution of messages sent per day over two date ranges within the data, which varies widely over the course of the collection period.

We created an adjacency matrix for each day, with values corresponding to the number of messages sent from one person to another on that day. We then grouped the data by week. Because the message counts are not evenly distributed through the data set, we created two different train-test splits. The first training set consisted of data from weeks 1 through 18 and the corresponding test set consisted of data from the remaining weeks 19 through 27. For the second data split, we only included data from the last 17 weeks of the set (the portion represented on the lefthand side of Figure 10). The second training set consisted of data from weeks 11 through 22, and the test set consisted of data from weeks 23 through 27. Additionally, we created a small sample of only 200 nodes from the full data set for testing purposes.

3.2.3 Detecting Anomalies in Tabular Data

As an example of identifying anomalous behavior in tabular data using unsupervised approaches, we leveraged the Isolation Forest algorithm from `scikit-learn` and applied it to a sample from the DARPA's OpTC data set in tabular format. Isolation Forests attempt to identify outliers in the data by randomly splitting on feature values to produce a decision tree. Decisions reached with shorter paths are considered to be anomalous [K.20]. Isolation Forests can be tuned according to a contamination parameter, which is a measure of expected anomalous behavior within the data set.

We used the OpTC data set in these experiments as it contains anomalous network behavior and has associated open source code repositories. We worked with a subset of the OpTC data set that corresponded to a single day of network data during the original red-team event, and selected the *actor*, *action*, *object*, *timestamp* fields as features. For these experiments we converted the timestamp information to be time in seconds elapsed from a previous event rather than date-time format. We ran the algorithm using the features encoded two different ways: as-is and one-hot encoded as categorical data. In both cases, we set the parameters to use a contamination rate of 0.1 and maximum sample rate of 10,000. We considered all data by malicious actors to be anomalous in our accuracy measures, though this interpretation of the data might vary from that of cybersecurity practitioners. Table 8 shows the results of running the Isolation Forest algorithm on the single-day OpTC data.

Encoding Method	True Positives	False Positives	False Negatives
As-is	279	12,299	26,402
One-hot	1,348	11,230	25,333

Table 8: Isolation Forest Results for Single Day of OpTC Data

The algorithm falsely identified numerous instances as anomalous in both encoding schemes, which would require more manual review of the potentially anomalous activity than is realistic for operational applications. Additionally, a large number of malicious events were not predicted to be anomalous. In the context of identifying information to include in a TLDR, the high volume of false positive and negative predictions would overwhelm a user with incorrect information and would be counterproductive in an operational setting.

3.2.4 Detecting Anomalies in Graph Data

Knowledge graphs incorporate more semantic information than graphs that do not specifically capture knowledge, so we sought to better understand the influence of the knowledge component of knowledge graphs on graph-specific anomaly detection. In this work, explored the suitability of DeepSphere [DMGP20] (a graph-specific anomaly detection algorithm) for identifying changes in a data source that should be presented to a user in a TLDR. DeepSphere analyzes a time series from a graph and predicts which the times and node pairs are anomalous within that data using Long Short Term Memory (LSTM) autoencoders with hypersphere learning. The hypersphere learning occurs between the encoding and decoding layer of the autoencoder, which helps discover and "remove" the anomalies

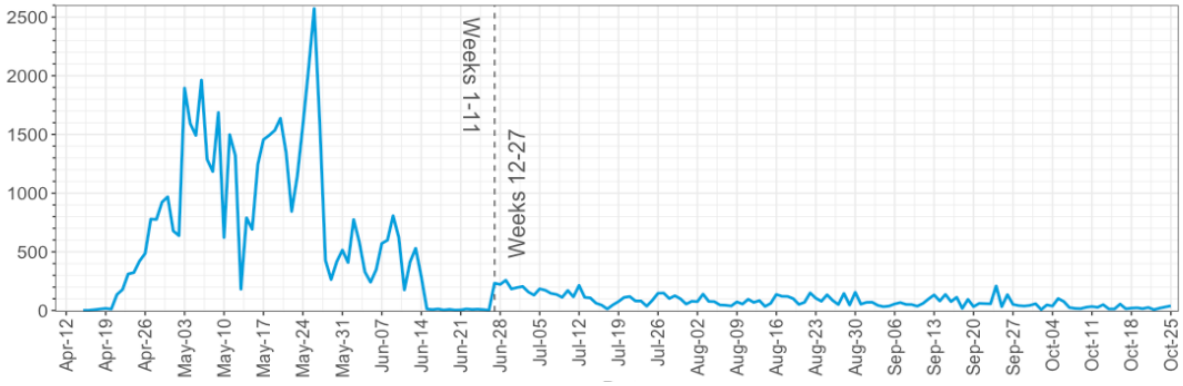


Figure 10: UCI total message count by day

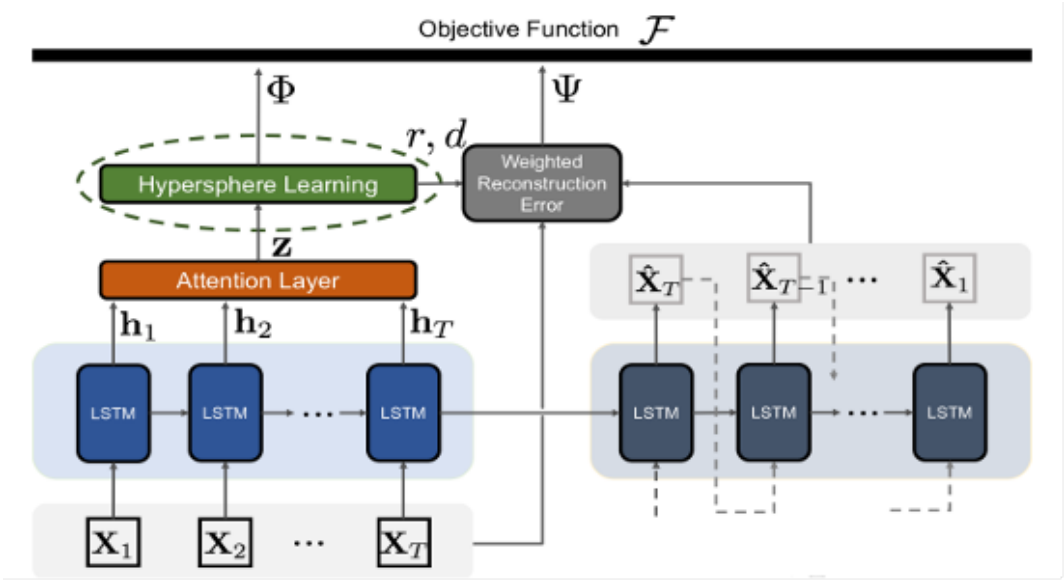


Figure 11: DeepSphere architecture

so that the LSTM autoencoder only learns the normal behavior of the system. In this architecture, the hypersphere learning mechanism gives rise to the case level anomalies, while the LSTM autoencoder will be able to alert the user to a nested anomaly, as the difference between the input and the output will be higher for anomalous behavior.

The time series input to the LSTM autoencoder is formatted as an adjacency matrix for each time step within the data, and the output is a reconstructured version of these adjacency matrices. Tunable LSTM hyperparameters include hidden layer size, dropout probability, and number of epochs. Hypersphere learning occurs between the LSTM encoder and decoder, and is visualized in Figure 11 [TYEL18]. In that step, the model learns a center and radius for the data and anything outside of this radius from the center is considered anomalous. The DeepSphere loss function weighs the points predicted as anomalous during hypersphere less than points predicted as normal, meaning the autoencoder effectively learns to ignore anomalous data in the decoder section. *Gamma* is the main tunable parameter for hypersphere learning, which serves as a penalty term for anomalies.

We extended the DeepSphere code from [TYEL18] for our experiements[5] and employed a g5.xlarge AWS instance for model training. When we trained the model using the first UCI training data set from weeks 1 through 18 of the data, we determined the ideal hyperparameters for balancing loss and penalty distribution to be as follows:

- 20 hidden layers
- 150 epochs
- *keep probability* = 0.95
- *lambda* = 2
- *gamma* = 0.1
- *tolerance* = 0.01
- *learning rate* = 0.001

We ran each of the three samples of UCI test data through the trained DeepSphere model, with the smallest sample that contained 200 nodes serving as a validation step. Our manual inspection of the model output for the smallest sample the model indicated the results were reasonable, with anomalies predicted in most weeks at the case-level and some variety in the predicted values for

[5]https://github.ncsu.edu/SCADS/DeepSphere extends the code found in https://github.com/picsolab/DeepSphere

nested-level anomalies. The model output for the two larger samples, however, did not align with our expectations. For both of the larger UCI data samples, the model predicted all test data to be anomalous at both the case- and nested-level.

We proceeded to identify and isolate potential problem areas in our implementation by running a test set through a regular LSTM autoencoder using parameters nearly identical to those of the DeepSphere LSTM autoencoder used in these experiments. The output from this effort varied from the input, as we would expect, which suggests the problem occured within the hypersphere learning portion of the model. Results from running the DeepSphere implementation on the PAYSIM test data were similar, in that the number of case-level anomaly predictions was much higher than expected. For comparison, one analysis of the PAYSIM data showed that less than 0.1% of the data set represented anomalous behavior, which is much less than the amount predicted by our model.

3.2.5 Discussion

When attempting to detect anomalies in the OpTC data set using the `scikit-learn` Isolation Forest algorithm, there were too many false predictions to consider this a viable approach to identifying anomalous behavior in network traffic to present in a TLDR. It is possible that using a larger portion of the OpTC data or selecting different features would result in more accurate predictions and a reasonable volume of potentially anomalous behavior for a TLDR user to review. Our interpretation of the events by malicious actors as anomalies to be detected likely also contributed to a large number of predictions being considered false, as some events by malicious actors were identical to typical events by any network user.

Our implementation of the DeepSphere model predicted all data as anomalous, which was clearly incorrect. We suspected this was due in large part to data sparsity issues that rendered the DeepSphere model inappropriate for this task, which could have been exacerbated by the manner in which we sampled data for this work. Additionally, DeepSphere was highly memory intensive due to its use of adjacency matrices, which suggests that the model might not scale well enough to feasibly apply in a TLDR use case. Given the low accuracy issues in this experiment and computing resources required for running DeepSphere operationally, we do not recommend pursuing DeepSphere as a mechanism for detecting anomalies in knowledge graphs to be reported in a TLDR.

4 Summarizing Content for a TLDR

An essential component of the TLDR system will be the ability to condense information into a format that users can quickly consume and decide whether to act on that information. Formats for condensed information might include graphics or other visuals, or could be short text-based summaries of source data. Content to be summarized might originate from different sources and exist in structured or unstructured formats. The TLDR system should be able to automatically generate these summaries and customize them for each user. As we learned through interaction with knowledge workers, each TLDR user might be interested in different aspects of the same information source, so it might be appropriate to generate several different summaries of that information. For example, one user might want to see only new developments in an area they are familiar with, while another user might want to see all content relevant to their topic area of interest. Given the existence and availability of automatic summarization tools and techniques, we explored how to extend those for the TLDR context.

4.1 Background and Related Work

Automated text summarization is not a new challenge [Luh58]. 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, and includes these sentences word for word in the resulting summary. In contrast, abstractive summarization generates 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 [MPMS19]. Data sources consisting of audio or video data might contain text-based transcripts that could potentially be summarized using extractive or abstractive techniques. Other data sources might not contain any corresponding text-based portion, for example network traffic data, but may have some other mechanism for providing context to that data which could then be incorporated into a text-based summary.

4.1.1 Extractive Summarization

Extractive summarization expresses sentence value in terms of key word frequencies, prompts, or location 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 instead of individual words. In either case, the final summary is constructed by pulling the highest valued sentences together according to some length-limiting factor established *a priori*. 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.

One major challenge with extractive summaries is defining how the model identifies the word (term) values. The work of [DCS12] aimed to develop an algorithm that improves term selection for extractive summaries by maximizing information value of sentences to include in a summary and minimizing redundancy within a summary. The software package `occams` was 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 algorithmproposed by [GF09]. The approximation algorithm has been shown to give over 90% of the optimal coverage with over an order of magnitude performance improvement [CD18]. The current release of `occams` has a new highly optimized Rust implementation of the covering algorithm, which breaks the extractive summarization task down into three subtasks:

1. Segment input document or documents (the input text) into sentences and terms.
2. Compute term weights to indicate the relative importance of the terms in the input.
3. Given a target summary length in characters or words, select a subset of sentences that maximizes the weighted coverage of terms by selecting a subset of sentences using an optimal approximate algorithm, such that the sum of the sentence lengths does not exceed the budgeted target length.

The resulting summary can be influenced by modifying the length of summary to generate or which terms over to favor over others, but will be constrained to content that appears in the source document or documents.

4.1.2 Abstractive Summarization

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 sentences. The encoding on the front end is usually accomplished through either structural or semantic approaches, where terms are classified according to where they appear in context or how they appear relative to either domain or language-specific characteristics [DS18]. These NLP techniques are generally much more complex than the models used to extract whole sentences in extractive summarization. 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, inference, and natural language generation to produce summaries consistent with the informational value and intent of the original documents [GG18]. The tradeoff is that abstractive summarization can produce a shorter version of a given sentence or combine multiple sentences while preserving the intended meaning [CAR18].

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 later fine-tuned for other tasks. The models have two parts: 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, and the *decoder* takes this representation and generate an abstract. The models are trained with hundreds of thousands of text-human abstract pairs, and training typically takes days of computation on multiple GPUs. Huggingface.co has several pre-trained abstractive summarization models available. While OpenAI's ChatGPT was not publicly available until after SCADS 2022 concluded, this technology also appears promising for future investigation.

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 "hallucinations" or "fake facts" generated by the model will result in the abstractive summary which do not appear in the source text. A study designed to capture the faithfulness of abstractive summaries estimated that nearly 30% of the output results of these systems suffer from hallucinations [CWLL18]. That said, abstractive summaries can perform better at informational value as they are not reliant on existing sentence structure [MG14].

4.1.3 Knowledge Graph Summarization

Knowledge graphs are rapidly gaining traction as an important tool for concisely and intuitively providing a "shared substrate of knowledge within an organization" [NGJ+19]. They have been shown to be an effective data structure for recording intelligence information, as they can capture evolving factual information [Hea21] in a structured manner that allows for querying "at a high level of abstraction" [AG08]. This flexible, queryable technique is especially valuable in the context of the grand challenge posed this summer, to "generate tailored daily reports for knowledge workers that capture information relevant to their individual objectives and interests."

However, most approaches for communicating the contents of a knowledge graph focus on explicitly visualizing the knowledge graph itself [NKI20], rather than treating it as a data structure. As their level of detail increases, the techniques adapted from the graph visualization research community (e.g. node-link diagrams, adjacency matrices) quickly become intractable for direct use in operational settings. Moreover, traditional approaches to visualizing networks are largely predicated on Shneiderman's Visual Information-Seeking Mantra "overview first, zoom-and-filter, details on demand" [Shn96], which assumes that the most important information to convey is related to the overall structure of the network. While useful in many information-foraging contexts, there is a mismatch between this approach and the way people tend to use knowledge graphs, in which they place less emphasis on general characterizations of the connectivity and shape of the graph and more on the relationships between a specific subset of nodes, certain types of relationships, and other local features.

4.1.4 Audio Summarization

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 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 [TDM11]. It is common to divide audio data, whether in audio or transcribed text format, into smaller portions, or segments, prior to summarization. Common segmentation techniques leverage duration and content similarity to create meaningful subsections within the data.

For example, when summarizing broadcast news videos, [HBCB21] 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. [YS22] 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. 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. Their approach performed better than traditional HMM approaches but not as well as neural network-based approaches on the same data.

4.1.5 Measuring Summary Quality

What is considered a "good" summary can be highly subjective, which can make it challenging to compare the quality of automatically-generated summaries. ROUGE, or Recall-Oriented Understudy for Gisting Evaluation, is a widely accepted suite of metrics for evaluating summaries. ROUGE quantifies the amount of overlapping n-grams between a generated summary and a reference summary written by a human, typically notated as ROUGE-n [Lin04]. ROUGE scores closer to 0 imply minimal overlap between the reference and generated summaries, while scores closer to 1 indicate high similarity between the summaries. We note that there are limitations of using ROUGE scores for measuring summary quality. As ROUGE compares automatically-generated summaries to existing human-generated summaries, variation due to tailoring in the automatically-generated summaries might result in less overlap with the human summaries and lower ROUGE scores. For example, [DO08] noted an example of a summarization system that performed well in human assessments in areas, such as "readability," and poorly with automatic metrics, such as ROUGE.

Alternate methods for evaluating summaries attempt to address some of the drawbacks of ROUGE, such as relying on a reference summary and considering only surface forms of the data and ignoring meaning [EBE19]. One approach called BERTScore uses contextual embeddings to compute the similarity between the tokens in two summaries [ZKW+19]. Another approach leverages question-answering (QA) systems to quantify the factual consistency of a summary by identifying the number of questions based on the source document that can be answered using only information from the summary. Both approaches attempt to quantify overlapping *meaning* between the source and summary documents rather quantifying overlapping *words*. The QA approach can be applied to both extractive and abstractive summaries, and can also assist with identifying hallucinations in abstractive summaries [FWLX22]. Because QA-based tools rely on a pipeline which includes automated components, their evaluation results can vary depending on the effectiveness of individual components [FWLX22].

4.2 Data

We identified multiple different data sets with which we could investigate various approaches of condensing or summarizing that data to present in a TLDR. In addition to the data sets described below, we also used the MIND data set introduced in Section 3.1.1 in some summarization experiments.

CNN/Daily Mail

The CNN/Daily Mail data set is an English-language corpus containing just over 300,000 unique news articles as written by journalists at CNN and the United Kingdom's Daily Mail [HKG+15]. 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.

Nixon White House Tapes

The Nixon White House Tapes were 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. For this work, we used a version of the data that had previously been augmented by LAS. 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 on AWS, and stored the results in a consistent JSON format. Additionally, LAS further annotated portions of the data set to include gists of certain portions of the recordings that pertain to various conversation topics.

Text Analysis Conference (TAC) 2008-2011

The Text Analysis Conference (TAC) started in 2008 and grew out of the National Institute of Standards and Technology (NIST) Document Understanding Conference (DUC) for text summarization [DO08]. TAC data sets include 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. 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.

4.3 Pre-processing for Automatic Summarization

One way to potentially improve automated summaries without modifying the summarizer itself is by pre-processing data prior to running it through the summarization process. We explored using coreference resolution as pre-processing step to disambiguate pronouns in a text prior to summarization. We focused on coreference resolution because automatic text summarizers, particularly via extractive text summarization, tend to devalue sentences that contain pronoun references compared to sentences that contain proper noun references. This can lead to potentially excluding sentences from the summary that are meaningful to a reader, or conversely, including ambiguous sentences that lack the appropriate referential context [SK16]. Coreference resolution has also been shown to improve performance of abstractive summarization, as it reduces ambiguity in the training corpus [LSC21].

We began by evaluating the suitability of existing open source coreference resolution tools to incorporate into our TLDR research and found that there are few available. We chose to use the AllenNLP coreference resolution model during SCADS, as it was one of the most commonly used open source coreference resolution tools we identified. The AllenNLP model uses a combination of predictive model to identify coreference clusters, where a coreference cluster is a group of tokens within a document that are identified as having the same referent. It then applies rules to replace coreference tokens with the associated referent. A Huggingface model (Neural Coref) is also commonly used, but this implementation is not compatible with Python3.8+ or spacy 3.x+. AllenNLP is generally described as finding more potential coreference "clusters" and having higher false positive rates, while Neural Coref is desribed as finding fewer clusters and having higher false negative rates, comparatively.

We supplemented the AllenNLP model with custom rules defined for the TLDR use case, which was an approach motivated by an NLP case study we reviewed from Neurosys [MM21]. We developed the rules with the goal of updating texts to consist of sentences that were weighed similarly by an automatic summarizer, maintaining efficiency in the coreference resolution process, and creating sentences that are both readable and grammatically-sound. In practice, this meant minimizing the number of replacements made in an attempt to minimize potential errors due to false coreference cluster predictions and/or improper sentence syntax after replacement. For this work, we defined a coreference span as *meaningful* if it contained a noun or pronoun phrase, or conversely a span was *non-meaningful* if it did not contain a noun or pronoun phrase. We identified a cluster as *redundant* if all spans in that cluster are identical.

The rules we implemented are:

1. Only one coreference is replaced per sentence.
2. If there are no meaningful coreference spans in a cluster, that cluster is removed. For example, in the sentence "It was raining last Thursday." 'It' does not refer to anything else in the sentence, so the cluster containing 'It' would be removed.
3. If there is a non-meaningful span in a cluster, that span is removed.
4. If a cluster is redundant, that cluster is removed.
5. If a span encompasses a span from another cluster, only the inner span of a nested cluster is replaced. For example, in the sentence "The dog keeps moving his bowl." 'his' is nested within the longer span 'his bowl.'
6. If a span contains two clauses, those clauses are split into separate spans to improve readability.
7. Each cluster's referent is selected based on part of speech, entity, and placement within the text:
   (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.
8. If a cluster consists of exactly one meaningful span, that cluster is removed

Table 9 shows a portion of an Associated Press news article before and after applying AllenNLP and SCADS coreference resolution. The text formatting highlights the differences that result from using AllenNLP replacement rules versus the custom SCADS rules. We were not able to formally quantify the differences between the AllenNLP and SCADS implementations, but manual inspection suggests that the SCADS implementation improves coreference resolution results for the TLDR use case.

Original Sample	AllenNLP coreference	SCADS coreference
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.	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.	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.

Table 9: Coreference Resolution Example

We have identified several areas for improvement to be undertaken as future work. First, the code used to implement the coreference resolution can increase in efficiency and speed to support generalization to larger data sets. Second, the choice of rules could be modified or expanded. Some of the rules depend on the accuracy of spaCy's Named Entity Recognition (NER) and Part of Speech (POS) tagging models, and could be updated to incorporate those accuracy measures into the rules. Third, we suggest that the AllenNLP model itself could be modified to improve the accuracy of the generated coreference clusters, which was not measured. If another coreference resolution model becomes available, we suggest revisiting this work to see if a new model improves performance.

4.4 Tailoring Summaries Using Multiple Approaches

The information needs for each TLDR user vary depending on a variety of factors. For example, two users might be interested in the same source information for different reasons, and might therefore benefit from different summaries of the same information. We conducted multiple experiments looking at different approaches to produce an automated summary of unstructured text data that takes background information into account. We focused on finding sentences to include in a summary by using term weights to influence what information is considered relevant, and also on finding sentences to exclude from a summary because they contain irrelevant information. We used the `occams` extractive text summarizer for generating extractive summaries.

4.4.1 Weighing Topic-Applicable Terms

We first evaluated the impact of using topic-specific term weights when generating extractive summaries by comparing those summaries to ones generated using generic term weights. Term weights are real numbers associated with an n-gram that reflect the importance of that n-gram with respect to the document. The simplest approach to calculating term weights is to have term weight be the number of occurrences of term t in document D. We can increase the complexity of the term weight by incorporating information like position of term within document, number of times term occurs across multiple documents, and so on. For this work, we first created two background corpora to simulate differences in how frequently various terms occur in different domains. We created a general background (GB) corpus from the CNN/Daily Mail data set to represent generic English and a topic-applicable background (TAB) corpus from MIND data set labeled as pertaining to a specific topic by the SAS topic modeling function. For each corpus, we counted bigram occurrences for all bigrams in the corpus.

We then identified two sets of significant terms, roughly corresponding to a set of generally significant terms and topically significant terms. We started with the set of bigrams in the TAB corpus, conducted a G^2-test that compared each bigram to the bigrams in the GB corpus following the methodology established in [Dun93], and selected those bigrams for which the G^2 test had a p-value less than 0.0001 [CD18]. We referred to that set as Term Set 1. We then ran the G^2-test again to compare Term Set 1 to the bigrams in the topic-applicable background corpus, and selected those bigrams for which the p-value was greater than 0.01 as Term Set 2. By selecting terms above p of 0.01 we isolated terms with meaningful content by filtering out high-frequency terms, many of which contained stop words. When compared to Term Set 1, approximately 12% of the bigrams in the original document are considered significant, versus approximately 8% when compared to Term Set 2.

We calculated Fisher's term weights for the two sets of significant terms by computing the median, M, of all the term weights in the document of interest. We then added M as a constant to the term weights for each term in Term Sets 1 and 2, and calculated Fisher's term weights for each quantile of term weights in the document of interest. We used each set of term weights as input to `occams` to generate summaries for each MIND article in our sample, resulting in 10 sets of summaries, one per quantile per term set. We computed the ROUGE-2 score for all sets of summaries by using the MIND abstract as the reference summary, and also generated summaries using `occams` default positional-dense term weights for comparison (see Figure 12). The ROUGE-2 scores for summaries generated using the custom term weights results were lower than those created using `occams` positional dense term weights. This may be due to the reference summaries in the MIND data consisting of lead (the first few sentences of an article). Positional dense term weights incorporate position within the document, whereas the term weights in our experiment ignored term location in the document. Summaries generated using topic-specific term weights outperformed those generated using generic term weights, which suggests that this approach is promising for creating topic-specific summaries.

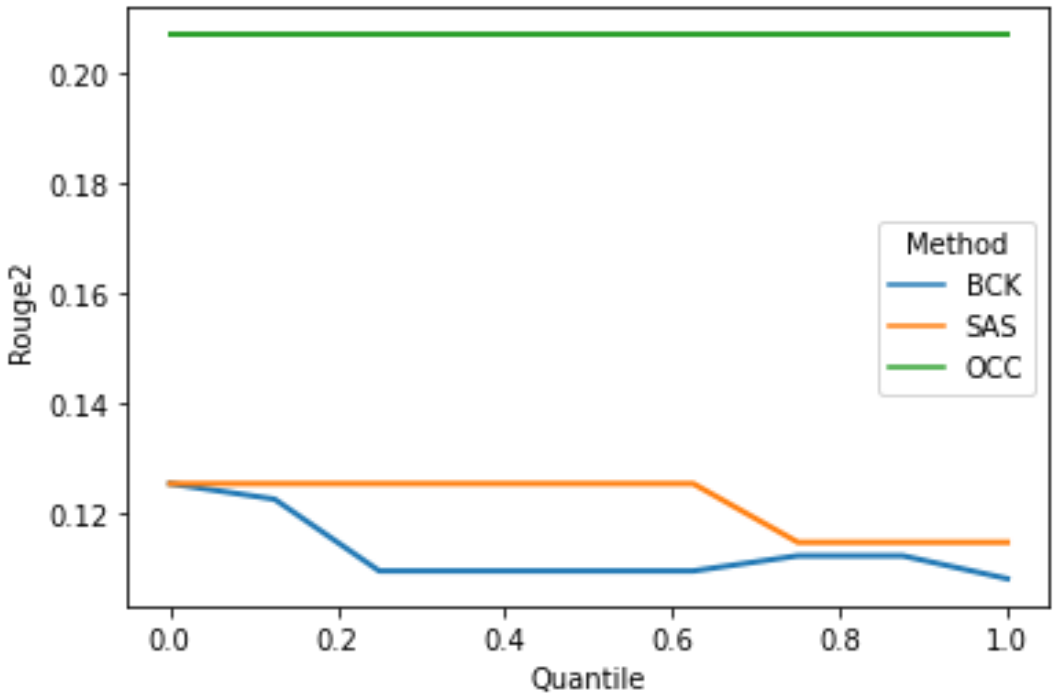


Figure 12: Average ROUGE-2 of a 30-document sample when compared to a 2-test summary, 1-test summary, & `occams` summary

4.4.2 Subtracting Term Weights

We considered the TLDR use case in which the TLDR presents to the user a summary of information that is new since the last time the user viewed the TLDR. We refer to this as the "Two Days in the Life of an Analyst" use case. Because term weights are a critical input to `occams` to indicate the importance of a concept (instantiated here as a term), we decided to test whether lowering term weights for terms that appeared frequently in the documents from the previous TLDR view would be effective at prioritizing recent changes. Throughout this effort we used the NIST TAC data, which presented a similar "Two Days in the Life of an Analyst" use case for context when generating the provided data.

To test the impact of adjusting term weights based on prior appearance in the TLDR, we used `occams` and its default term weighting [GF09] on Set A and Set B independently. This allowed us to extract the term weights from both days to serve as "base" term weights for each set. 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.

We then reduced the term weights of a term for the second day by some fraction, f, of its weight from the first day. Our hypothesis is that this would steer the Set B, or Day 2, summary towards new material. We tried to try to identify an optimal value for the fraction, f, by which to reduce Set A term weights. We iterated over possible values of f from 0.0 to 1.0 and computed the new Set B term weights using the base Set A and Set B term weights. We generated summaries for Set B using `occams` with the new Set B term weights, and computed ROUGE scores using the human-generated summaries as the references.

We then computed the mean ROUGE scores for the data set. We plotted the mean ROUGE scores as a function of the fraction of Set A term weight that was subtracted from Set B. The resulting mean scores for ROUGE-2 and ROUGE-3 are plotted in Figure 13 for the 2008 TAC data and in Figure 14 for the 2010 TAC data. The baseline in each figure is the mean ROUGE scores for summaries generated using the base Set B term weights. These 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 approach is not effective at driving the summaries toward new material.

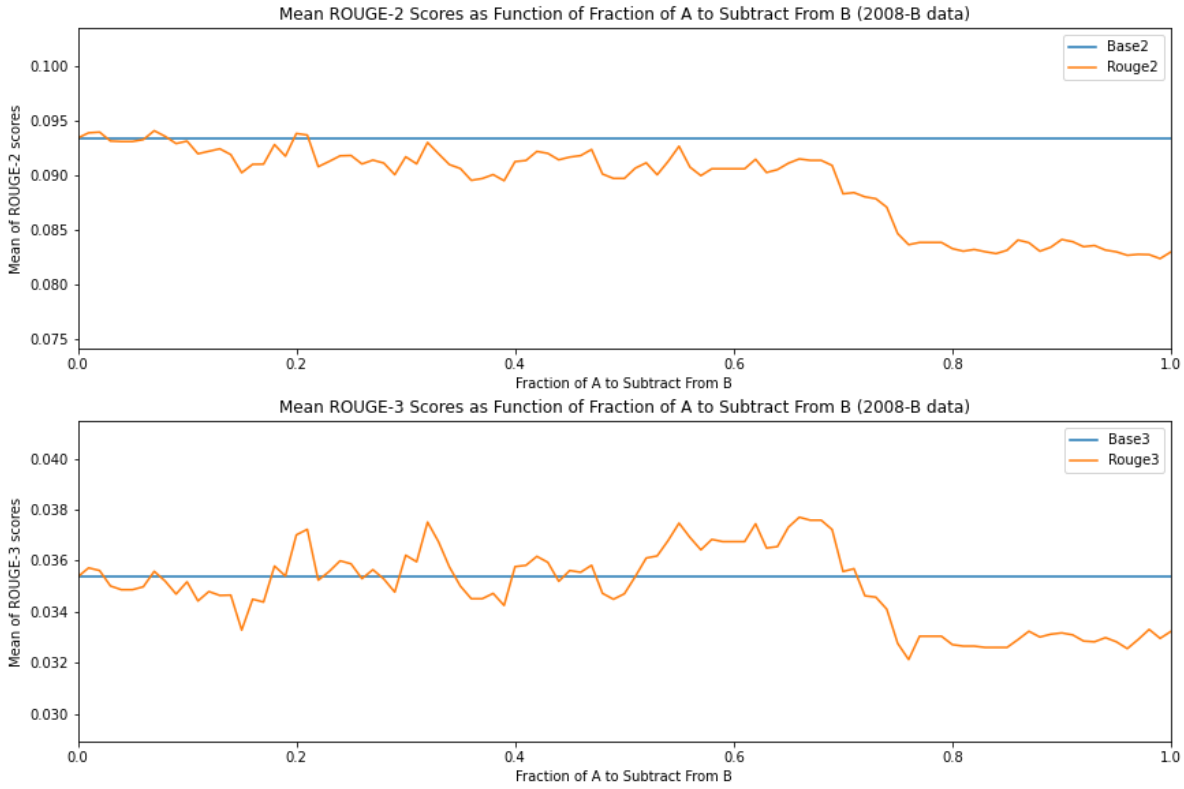


Figure 13: Mean Set B ROUGE-2 and -3 Score by Set A Term Weight Fraction on TAC 2008

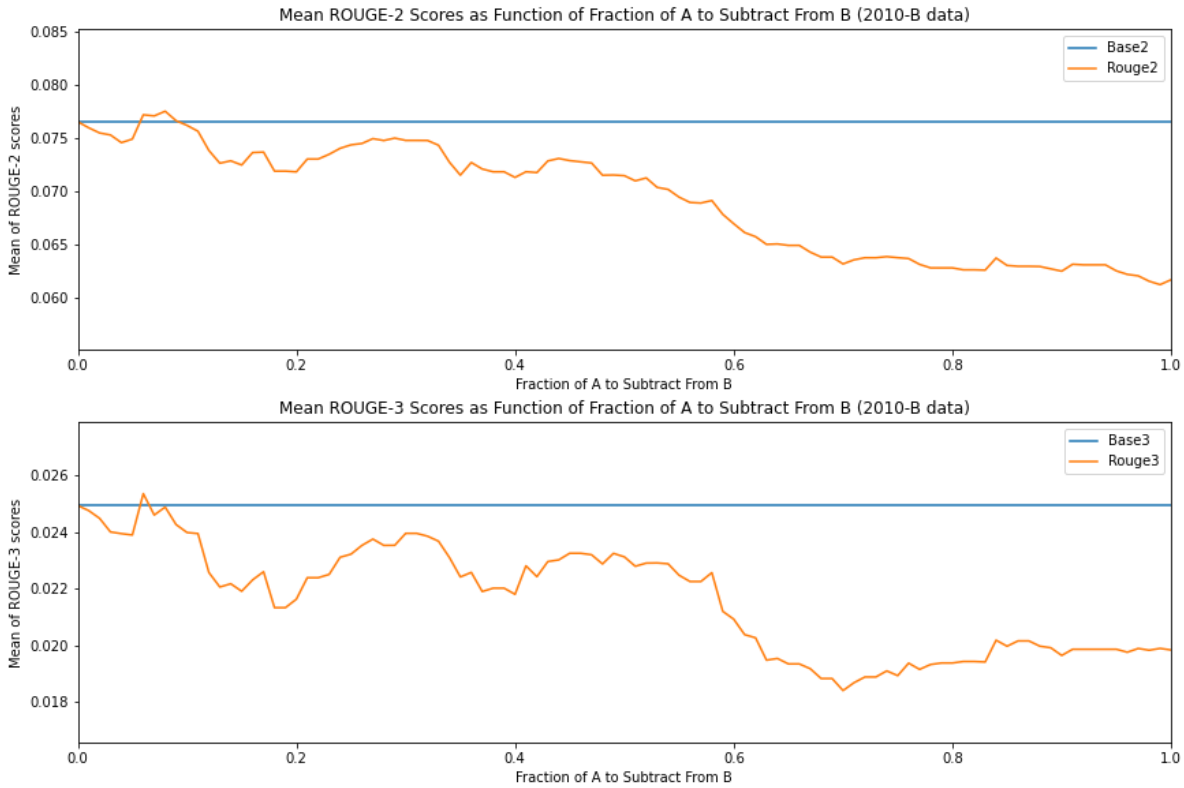


Figure 14: Mean Set B ROUGE-2 and -3 Score by Set A Term Weight Fraction on TAC 2010

We analyzed the data further and assessed that Set B sentences with bigrams that do not really reflect the content of the topics might overly influence the term weights. For example, in one topic, the sentence "That would have been good for the country" appeared in many extracted summaries over a range of values of *f*. Every bigram from this sentence appeared at least as often in the Set B articles compared to the Set 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 discussed.

To investigate this, we tested whether removing the influence of stop words and other high-frequency words would improve the `occams` summaries for the given topic. We used the bigram counts from the CNN/Daily Mail corpus as a background corpus to represent generic English language. For any terms with a frequency above a given threshold in the background corpus, we set the term weight to 0. We iterated over thresholds ranging from term frequencies of 1,000 to 100,000 and generated summaries with `occams` using the appropriate term weights for that threshold. We computed the mean ROUGE scores for those summaries, which are shown in Figures 15 and 16. The baselines correspond to summaries generated with original term weights on that data set.

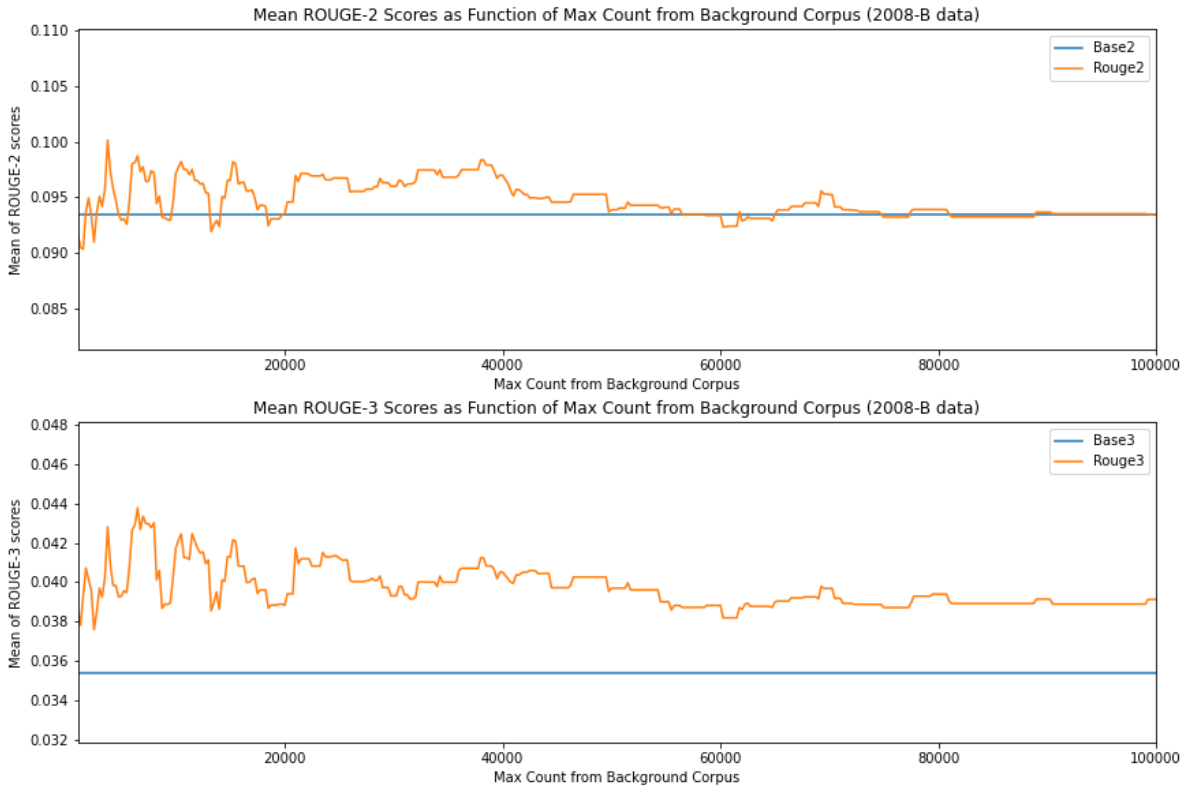


Figure 15: Mean ROUGE-2 and -3 Score by Term Frequency Exclusion Threshold on Set B TAC 2008

For the 2008 data, the ROUGE-2 scores were often above the baseline and the ROUGE-3 scores were almost uniformly above the baseline. The ROUGE-2 baseline for 2008 Set B data was 0.0934, while the maximum ROUGE-2 of 0.1001 occurred at a term frequency threshold of 3,750. We were not able to calculate statistical significance during the available time, but the peak values of ROUGE-2 are likely statistically significant deviations above the baseline. For the 2010 data, the ROUGE-2 scores only outperformed the baseline at a few term frequency thresholds, while the ROUGE-3 scores were often above the baseline. The 2010 Set B ROUGE-2 baseline was 0.0766 and the maximum ROUGE-2 of 0.0768 occurred at a term frequency threshold of 52,000.

This work reinforced the notion that text data varies from corpus to corpus, and often requires specific processing. For example, there were multiple dips in the ROUGE scores at different term frequency thresholds in 2008 and 2010, which suggests that those data sets differ in some way that needs to be considered during processing. In the future, we would like to investigate whether documents in a corpus could be grouped to optimize extractive text summarization approaches and what metrics could be used

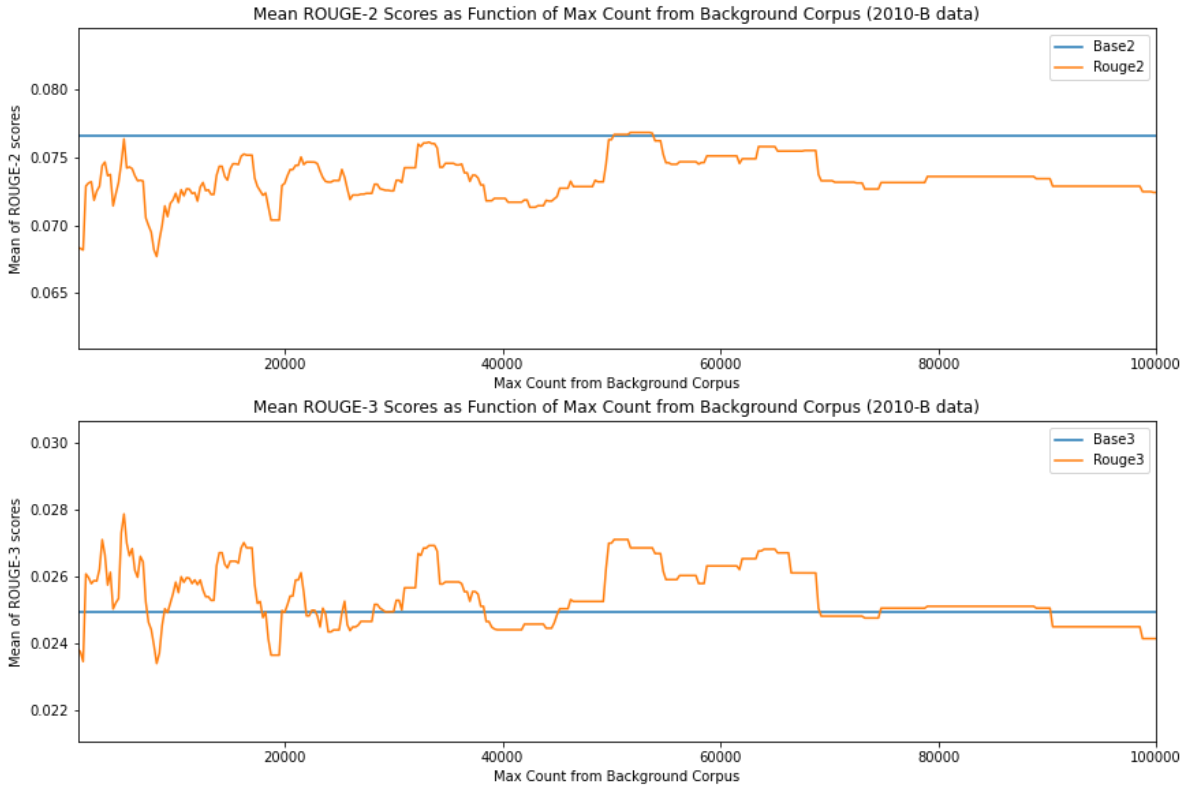


Figure 16: Mean ROUGE-2 and -3 Score by Term Frequency Exclusion Threshold on Set B TAC 2010

to capture this. We would also like to investigate using the `thresholdLogFrequencySummaryExtractor` class in `occams` to incorporate testing against a background corpus, normalizing all the terms so they can be used equivalently, as well as using different term frequency schemes and term weight types.

4.4.3 Excluding Previous Content

We used the NIST TAC data and "Two Days in the Life of an Analyst" scenario described in 4.4.2 as the context for additional text summarization efforts. In this case, we attempted to identify similar sentences across two sets of documents. We then generated a summary using the sentences that were not similar, working under the premise that non-similar sentences would contain information unique to that data set. We explored the use of sentence embeddings to identify similar sentences in both Sets A and B. Pretrained transformer neural language models provide a convenient representation of sentences, or "sentence embeddings," in which a string of input text is mapped 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 [RG19]. 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. We embedded sentences from Sets A and B of the TAC data into 384-dimensional space using three models: MiniLM-L6-v2 (MiniLM) [RG19], all-distilroberta-v1 (RoBERTa), and all-mpnet-base-v2 (MPNet).

We trained a Quadratic Discriminant Analysis (QDA) classifier using the sentence embeddings for Sets A and B to assign a probability that a sentence belonged to Set A or Set B. Using such a classifier, we could identify sentences that "look like day 1" sentences then add those to a list of sentences to be excluded from a summary of new information on day 2. We trained the classifier in such a way that it was possible to tune a parameter for how many sentences to exclude depending on the threshold value, which depends on several variables and is data set specific. Figure 17 shows the ROUGE-1 and ROUGE-2 scores for summaries generated using different embedding models and classifier threshold values as compared to the baseline ROUGE-1 and -2 scores, where the baseline was computed by running `occams` on Set B in isolation.

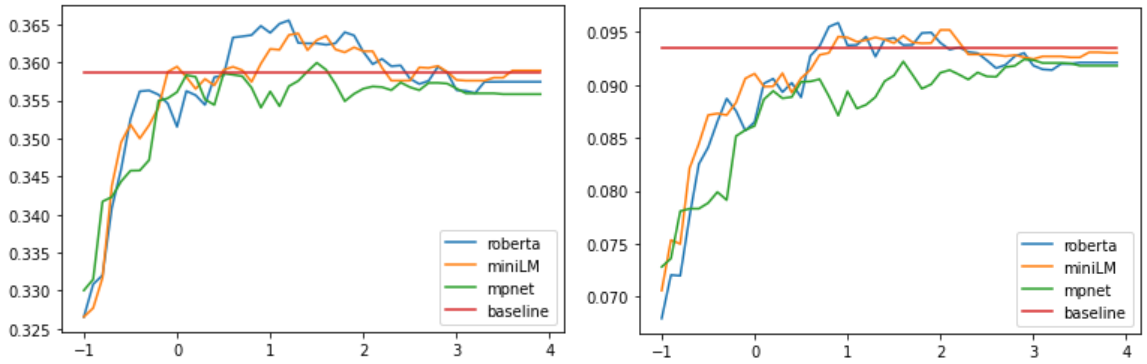


Figure 17: RoBERTa, MPNet, & MiniLM ROUGE-1 (left) and ROUGE-2 (right) vs Baseline by QDA Threshold

The RoBERTa embeddings show a modest improvement over the MiniLM embeddings, while the MPNet embeddings performed worse than the other two sentence encoders. There are ranges in which the RoBERTa and MiniLM methods outperform the baseline, and the improvements are more pronounced for ROUGE-1 scores than ROUGE-2. Our approach using sentence embeddings appears to be highly sensitive to which embeddings are used, and the results are not obviously statistically significant. This approach does not appear to be as productive as we had hoped in generating summaries that capture only new information in a data set.

4.4.4 Using Additional Information

`Occams` determines the most informative 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. We considered timestamps and sentiment predictions as additional information to incorporate into `occams` summary generation.

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, which consists of the first few sentences of the most recent document for a topic and does not exceed 100 words, and observations from manual inspection that more recent documents appear to have a greater influence on update summaries by humans. The temporal schemes did not outperform the default scheme, *POSITIONAL_DENSE*. Some variants were comparable.

For each topic, we grouped the 10 Set A documents and 10 Set B documents together and computed the term weights over the combined 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 five 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 1*D[19] + 2*D[20]
  - n=3: add additional 1*D[18] + 2*D[19] + 3*D[20]
  - n=4: add additional 1*D[17] + 2*D[18] + 3*D[19] + 4*D[20]
  - n=5: add additional 1*D[16] + 2*D[17] + 3*D[18] + 4*D[19] +5*D[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)

We also applied a few scalar multipliers to NEW_SCALED and 1_NEW_SCALED. Those results are omitted here, and only the best results are provided in Table 10.

Data Set	Weighting Scheme	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-4
2008 Set AB	POSITIONAL_DENSE	0.3466	0.0772	0.0250	0.0113
	POSITIONAL_NEW_SCALED_2	0.3451	0.0781	0.0262	0.0120
2010 Set AB	POSITIONAL_DENSE	0.3290	0.0672	0.0178	0.0066
	POSITIONAL_NEW_SCALED_2	0.3307	0.0666	0.0186	0.0067

Table 10: Set AB Temporal Metrics

The POSITIONAL_NEW_SCALED_2 ROUGE scores are higher for most versions of ROUGE, however 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 [DO08] shows the top performing group of systems as having ROUGE scores in the range of 0.094 to 0.130. This group included the eight human summarizers, with the lowest scoring human summarizer having a ROUGE-2 score of 0.108.

We hypothesized that the sentiment of a summary should reflect the sentiment of the source data. We assigned sentences in the data set a sentiment score using the TweetEval model [BCCAN20], which showed the TAC 2008 data set to consist primarily of neutral sentences with a small portion of negative sentences. We calculated the ratio of the negative score to positive score for each sentence, and used that ratio as a threshold metric for excluding sentences from summaries. 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. We calculated ROUGE scores by averaging over each topic, as shown in Figure 18, and compared to the baseline score of the document averages.

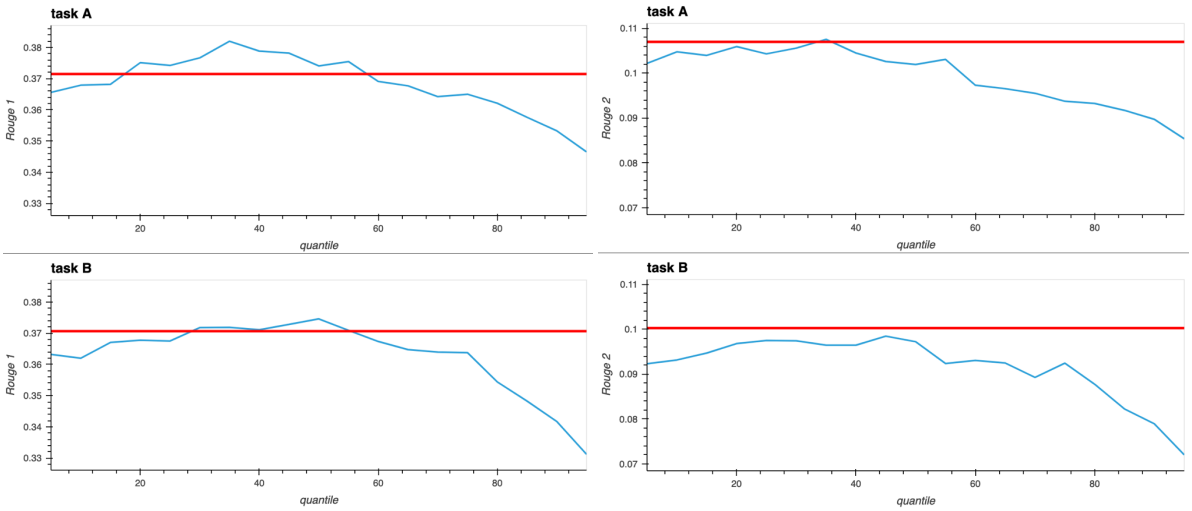


Figure 18: ROUGE-1 (left) and ROUGE-2 (right) vs Baseline by Negative-Positive Sentiment Ratio Threshold

We did not see improvement in ROUGE scores with this approach, and observed minimal variation in the generated summaries. Only 55 of the 96 topic summaries changed when using the best-performing quantile threshold, most of which resulted in a lower ROUGE-1 score. This is likely a result of filtering by negative-to-positive sentiment ratio threshold, which excludes more positive sentences that might be information-rich. Our hypothesis is that while there is information in timestamps and sentiment, it is insufficient to leverage using simple methods. The various methods we applied to tailor summaries overall did not result in improved ROUGE scores for the generated summaries.

4.5 Case Study: Summarizing Transcribed Audio

Audio recordings can be tedious to analyze, especially when they are 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. Based on conversations with knowledge workers, we inferred that a TLDR system that supports multimodal data should be able to convey to a user the different topics that are covered in audio. We explored whether text summarization methodologies might help facilitate how analysts navigate audio, and focused on how to segment audio transcripts to identify different portions to include in a summary.

For this work, we used transcripts of the Nixon-era White House recordings generated with Amazon Web Services (AWS) speech-to-text service as the source content to be summarized. To segment the conversations for summarization, we isolated portions of the transcripts according to transcript length, speaker turn, or semantic content, following the approaches listed below. We then passed the transcript segments as input to automatic summarization tools. For abstractive summaries we used the summarization pipeline from Huggingface. We also generated extractive summaries of the full transcripts with no segmentation for comparison using the command line version of `occams`.

Table 11 contains the speech-to-text transcript with no segmentation, an expert-generated gist for that transcript, and the extractive summary generated for that transcript. We noted that the extractive summaries we examined were all of limited value to a TLDR user, regardless of the method used to segment the transcript. In particular, the extractive summary missed a portion of the time noted in the human gist. The full time information was included in the automated transcript, but the information was transcribed in such a way that it did not appear to be a time ("11. 30.") and was tokenized into multiple tokens rather than one single token ("11." and "30.") Additionally, the human provided more context in the gist than the transcript or summary provide, such as who exactly the speakers were and the overall topic of conversation. These issues could potentially be addressed with additional preprocessing.

Automated Transcript	Human Gist	Extractive Summary
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	Nixon and Haldeman are discussing the President's schedule. Haldeman mentions catching the Apollo guys at **11:30**.	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?

Table 11: Extractive Audio Transcript Summary Example

The abstractive summaries we examined contained "hallucinations" (see Sec. 4.1.2) that would be detrimental in a TLDR scenario. Table 12 shows the abstractive summary generated for the same example transcript from Table 11. The abstractive summary captures the time information from the transcript but also generates additional numeric information that does not make sense to a reader, highlighted in bold, which does not appear in the transcript.
Other hallucinations in the abstractive summaries were obvious anachronisms:

> 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 first sentence of the abstractive summary is also confusing, as it would be unexpected for the president to call himself to thank a third party. While somewhat amusing given their obvious falsity, such hallucinations should not be included in a TLDR.

Automated Transcript	Abstractive Summary
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	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.

Table 12: Abstractive Audio Transcript Summary Example

When we reviewed extractive summaries generated from segments based on clustering, we noted that the sentence clusters from within-transcript and cross-transcript clustering were often too short to allow for meaningful summarization. For example, the within-transcript sentence cluster shown in Table 13 repeatedly mentioned profits and specific types of profits. The extractive summarizer, however, excluded the mention of the specific type of profits. So, while a reader looking at the cluster might be able to discern that the conversation was likely about windfall profits and controlling those profits, the extractive summary did not include that information.

Sentence Cluster	Extractive Summary
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 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.	The point you raised about the profits. I think this decision was to And if we can tow work profits as a kind of. 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.

Table 13: Sentence Cluster and Summary Example

We noted that some of the sentence clusters from cross-transcript clustering consisted of phrases and errors that could be cleaned from the data set. Some of the errors were artifacts of the speech-to-text transcription, such as the repeated references to "toe" in the following cluster:

> 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.

This information could be useful to analysts when searching speech transcripts by identifying alternate search terms that might be confused with content they are looking for. Other cross-transcript clusters appeared to contain related sentences that would be relevant to a TLDR user but might not appear in conventional search results. Table 14 contains a portion of a cluster that included sentences containing "Laos," "Cambodia," "Saigon," and "Vietnam," along with the extractive summary. In this case, the clustering process grouped sentences referring to the same geographical region and apparent references to the Vietnam War. While the extractive summary generated for this cluster was of limited utility, the cluster itself could be useful for highlighting relationships in the source data.

Sentence Cluster	Extractive Summary
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.	Hurt the North Vietnamese for us. North Vietnam, this is your credibility that we wanna be. That's a Vietnam thing. 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.

Table 14: Large Sentence Cluster and Summary Example

4.6 Case Study: Summarizing Cyber Knowledge Graphs

When working with a knowledge graph as an information source one approach is to take a large, potentially dynamic knowledge graph that contains important contextual information (along with a significant volume of information that is not relevant to the current analytical context), identify a relevant subgraph, and present a human-readable text summary of the information it contains to the analyst [FGI+20]. To explore this in an operationally-relevant context, we implemented Structured Threat Information Expression (STIX™) 2.0/2.1, "a language and serialization format used to exchange cyber threat intelligence (CTI) [20221]"in the Neo4j graph data platform. We imported example STIX Advanced Persistent Threat (APT) reports to demonstrate the viability of using STIX to pivot information into KG format.

The STIX knowledge graph has a clearly-defined ontology with a limited number of node and edge types, making it a good use case for our proof-of-concept. One disadvantage of the STIX knowledge graph was that there is no provenance, so we were unable to trace back the source of the information contained within the graph. We also note that data in a STIX KG format would still need to be enriched to provide an easy automated way to traverse from low-level network traffic-type data to the higher-level concepts in STIX format, which provide context for the behaviors observed in more granular data. We implemented a proof-of-concept in Python and used the NetworkX package to work with the knowledge graph. We presumed that a node or nodes of interest had already been identified. We then took those nodes of interest and built a subgraph defined by the neighborhoods of those nodes, as shown in Figure 19.

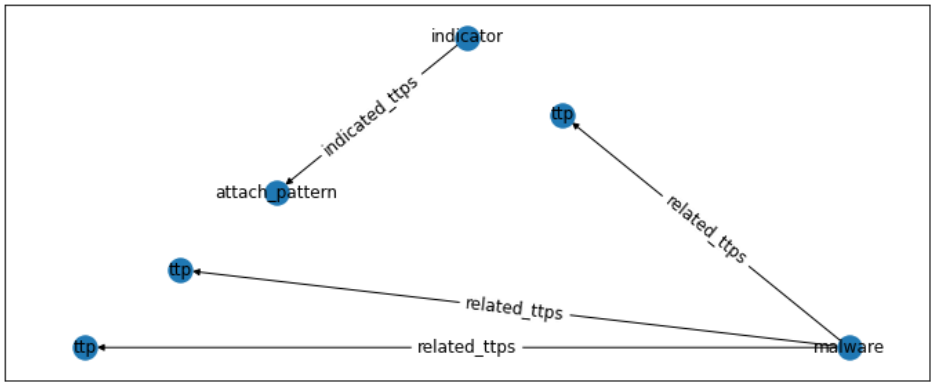


Figure 19: Example subgraph based on neighborhoods of nodes of interest

The knowledge graph ontology defined a finite set of entity and relationship types to explain how data in the KG is linked, which provided a handy grammar for converting graph substructures into text descriptions. We used a template-based approach, in which edges in the graph were mapped to particular sentence templates as shown by the examples in Table 15. The templates were filled in using the attributes of the nodes that form a given edge.

To simplify the subgraph and limit the number of output sentences, we bundled nodes of the same type that were adjacent to a common node, reducing the number of edges and output sentences to only one. Figure 20 shows an example of this bundling. In some cases there were multiple possibilities for bundling graph components. We leave the task of determining the best approach to bundling in these situations to future work.

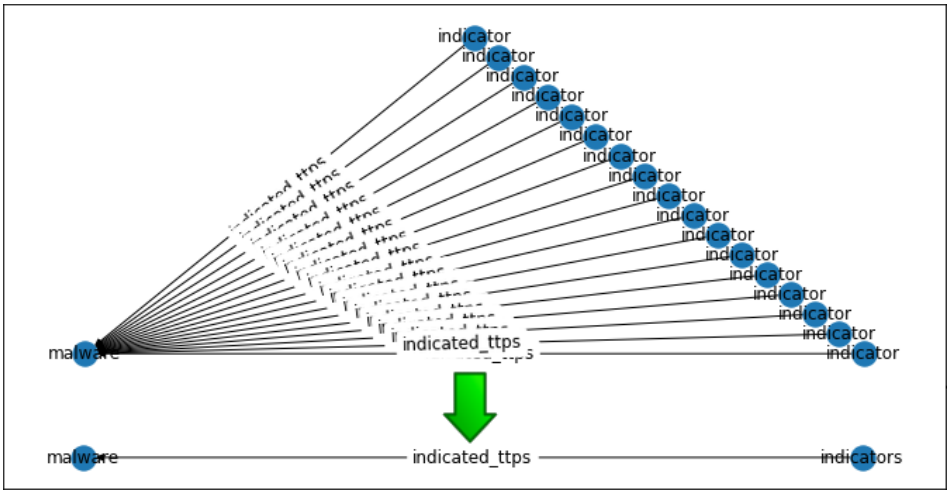


Figure 20: Multiple nodes of the same type adjacent to a common node (top) and the resulting bundled subgraph (bottom).

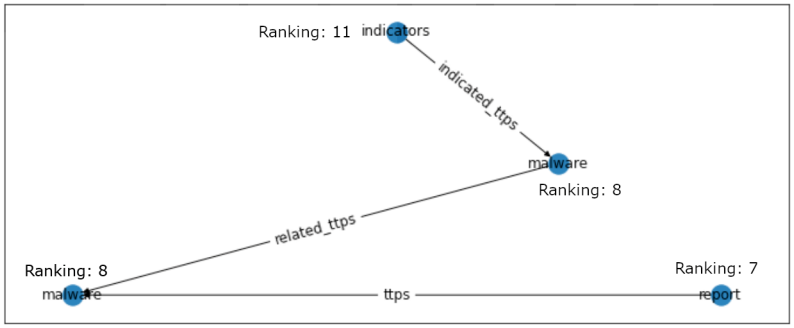
Once a set of sentences was produced based on the edges in the subgraph, we then had to determine how to order those sentences in the output. We assumed that users provided a ranked list of which node types interested them, and ordered sentences according to that ranking to tailor the generated reports to user interests. As shown in the example in Figure 21, the sentences generated using the template are technically grammatical but the information provided directly from the knowledge graph, such as malware names and indicator labels, is not very readable. A reader must also be familiar with detailed information in the KG, like the significance of different types of malware, to be able to interpret and analyze the generated sentences.

4.7 Discussion

Automatic summarization techniques are incorporated into tools we use in our daily lives, such as summarizing the results of an internet search. Summarization tools such as `occams` have demonstrated viable results in scenarios designed to simulate knowledge worker tasks, such as the "Two Days in the Life of an Analyst" use case and summarizing scientific documents [CD18]. Incorporating a tool like `occams` into a TLDR system would require additional consideration on what the appropriate use case is and what the user experience working with automated summaries should be. For example, we

Relationship Type	Sentence Template
associated_actors	<node1_value> is associated with <node2_value>
attributed_threat_actors	<node1_value> is attributed to <node2_value>
related_campaigns	<node1_value> is related to <node2_value>
indicated_ttps	<node1_value> is indicative of <node2_value>
mitigates	<node1_value> mitigates <node2_value>

Table 15: Sample of Edge-Sentence Mapping



Report: XPCTRA steals credentials for cryptocurrency wallets mentions
Malware: XPCTRA of type(s) **Bot, Bot - Credential Theft**

Malware Variant: XPCTRA 737491 of type(s) **Bot - Loader** has related
Malware: XPCTRA of type(s) **Bot, Bot - Credential Theft**

Indicator: COCA.CHEDDARMCMELT.TOP of type(s) **Domain Watchlist**,
Indicator: 4fec5a95ba822297b80c0fc83f81edd of type(s) **File Hash Watchlist** and
Indicator: 339c48b0ac25a9b187b8e76582580570 of type(s) **File Hash Watchlist**
are indicative of **Malware Variant: XPCTRA 737491** of type(s) **Bot - Loader**

Figure 21: Example Subgraph after bundling with Node Type ranking (top) and output after sentence ordering based on subgraph (bottom)

would need to determine if there is an optimal summary length and what factors go into determining summary length. Other outstanding questions include how a summary might vary based on source content, how much preprocessing to apply to source documents, or whether to reflect source document sentiment. As we observed when summarizing noisy audio transcripts and cyber knowldge graphs, the resulting summaries can be esoteric implying that a TLDR system might need to provide some other metadata in addition to the summary to aid in comprehension.

We explored some methods for tailoring a summary to a specific topic or toward new information in a corpus with varying degrees of success, however additional work would be needed to get user feedback on the utility of those summaries and increase understanding on what degree of personalization is needed. Some of the available data sets included human-written summaries that we could use as reference summaries when computing ROUGE scores, however those summaries were written for a general audience rather than tailored to a specific user or user group. Question-based evaluation could be useful to address this issue. Potential TLDR users could write a set of questions for a document or set of documents for which the answers should be provided in the summary. This would incorporate the user input without taking the amount of time needed to have the potential users write full summaries. Those questions could then also be used to evaluate the automatically-generated summaries and flag potential hallucinations in summaries created using abstractive summarization methods.

While working on approaches to personalize automatically generated summaries, we extended the functionality of the `occams` package to leverage different types of term weighting methods. These updates are available in the most recent version of `occams` and were used in our experiments on using term weights to influence generated summaries. Some of those experiments used a background corpus developed for use during SCADS based on the CNN/Daily Mail data set, which enabled researchers to calculate various term frequencies and influence the content of extractive summaries using those frequencies. This background corpus can be updated as needed for additional experiments to reflect the context of the use case on which summarization is being used.

To tell the story of the information contained in a knowledge graph, we developed a proof-of concept that takes a subgraph of a knowledge graph created using the STIX ontology and returns a text summary that describes network behavior represented in the graph. We started with nodes of interest to the user, built a subgraph based on those nodes and the connections to those nodes, and then produced a narrative that summarizes the relationships in the graph. We focused on generating a single text summary from a knowledge graph and presenting that in an automatically generated report. This demonstrated the ability to use a knowledge graph ontology to form a template for text-based reports on information contained within the knowledge graph, and highlighted points in the report-generation workflow that are challenging to automate.

5 Findings

In this section we revisit the research questions that motivated the work described in this report and present the main findings derived from the work related to that question. We share additional observations in Appendix H.

5.1 Understanding User Needs

We raised the following questions and identified the associated findings in an attempt to understand user needs for a TLDR.

What kinds of information do intelligence analysts engage with in their analysis work?

There is a gap in existing research around information needs in analyst workflows. The literature review in Section 2.1 described the large body of research that exists on analyst workflows, little of which focused on analysts' information needs. Additionally, much of the existing work on the topic of information was conducted with proxy subjects rather than subjects with experience as an analyst. Preliminary analysis of interview data showed that analysts work with formal and informal types of information throughout their workflow. Further investigation into an analyst's information needs is important in the creation of a TLDR to provide insight into where and how to incorporate personalization within the system.

How does information flow in the analysis work of intelligence analysts?

Non-technical factors influence the understanding and acceptance of knowledge worker output. Multiple interviewees in the study described in Section 2.2 noted that the nature of the small community within which they work helps to distribute information. Individuals tend to leverage their networks and contact authors of various reports to better understand the information they contain. It will be necessary to consider these types of interactions when creating TLDRs. For example, providing attribution for information included in a TLDR might support acceptance by users because they are familiar with the individual who created the source data.

What factors influence how intelligence analysts engage with information in their analysis work?

Finished reporting does not always reach users to whom the information is relevant. Multiple interviewees in the study described in Section 2.2 noted that when a report has reached the dissemination stage, it is not guaranteed to reach relevant readers in a timely manner or at all. TLDRs could be useful tools in addressing challenges in the information dissemination stage by connecting users to relevant information in an automated manner.

5.2 Identifying Relevant Content

While exploring methods of identifying content for a TLDR, the questions asked and related findings were as follows:

What is the impact to recommender model performance when training models with less training data than in top-performing scenarios?

Training a neural recommender model with less user history data than top-performing versions of that model only slightly reduced model accuracy. The NRMS neural recommender model described in Section 3.1.2 was trained on interaction history of many fewer users than the NRMS model version that topped the MIND recommendation competition, yet the effectiveness along all metrics was only marginally different. Additionally, the model hyperparameters were not tuned for the smaller training data set size, offering a clear avenue for model improvement. This suggests that state-of-the-art neural recommender models could be feasible for a TLDR system that has a smaller user base (and therefore a smaller amount of data from which to learn) than the large-scale systems for which these models demonstrated state-of-the-art performance.

Is natural language inference an effective mechanism for filtering redundant information from recommender model output?

The natural language inference model tested here falsely predicted contradictions between recommended article titles. In the context of a TLDR, Natural Language Inference could provide readers with insight as to whether recommended content supports or contradicts previously seen content. The NLI model used for the work described in Section 3.1.4 tended to predict that one sentence contradicted another when no contradiction was present. Falsely predicting that recommended content contradicts previously-seen content would be unacceptable for TLDR users. Additional refinement of NLI models to reduce false positives would be necessary prior to incorporating them into a TLDR system.

What are effective methods for explaining predictions generated by neural recommender models?

LIME can provide human-understandable, intuitive explanations for recommendations generated by a neural recommender model. We implemented the Local Interpretable Model-agnostic Explanations (LIME) Python module to generate explanations of the NRMS model behavior, as described in Section 3.1.5. LIME provided human-readable explanations of the features that factored into the model's recommendations and demonstrated a possible approach to explainable neural recommenders. Such explanations could be incorporated into a TLDR interface along with recommended content to provide users with additional context for why that content might be relevant.

What are effective unsupervised methods for detecting anomalies in tabular data and knowledge graphs?

Anomaly detection methods tested here generated too many false predictions to be a reliable method of identifying content to include in a TLDR based on changes in a data source. The Isolation Forest and DeepSphere models described in Section 3.2 generated high levels of false predictions in the experiments using the OpTC and PAYSIM data sets. In the context of a TLDR system that leverages anomaly detection to identify content to include in a TLDR, false positive predictions could lead to a user being overwhelmed with irrelevant information, while false negative predictions could result in a user never seeing relevant information. Additional effort would be needed to improve model performance or determine other mechanisms for identifying changes in a data source that warrant mention in a TLDR.

5.3 Content Summarization

In the area of summarizing content to include in a TLDR, we explored the following questions and reported the corresponding findings.

How can pre-proccesing improve summarization accuracy or informational value?

Applying coreference resolution as a preprocessing step in a text summarization pipeline could enhance the information value of extractive summaries. The updated coreference resolution method described in Section 4.3 modified sentences in a source text to use specific entity names throughout the text rather than references to that entity. Because extractive summarizers tend to consider sentences containing entity mentions to be more informative than sentences without entity mentions, the information value of the modified sentences increases. The resulting extractive summaries might then include different, more informative, content that is relevant to a reader than the summaries generated from the original source.

Does providing a domain-specific corpus improve summarization accuracy or informational value?

Favoring terms from topic-specific background corpus when choosing content for an extractive summary helps tailor the summary toward that topic. As noted in Section 4.4.1 extractive summaries generated using topic-applicable term weights to prioritize sentences to include in the summary had higher ROUGE scores than summaries generated using generic term weights. This suggests that tailoring

summaries by prioritizing sentences with terms that are prevalent in a user's area of interest could be an effective approach for generating tailored summaries to provide in a TLDR. Additionally, we evaluated several mechanisms for generating extractive summaries that favor new information in the second set of documents when given two sets of documents in sequence, and having reference summaries with which to calculate ROUGE scores that were written to favor new information in the second set of documents:

- *De-emphasizing previously-seen terms when choosing content for an extractive summary does not improve summary accuracy or information value.* Section 4.4.2 described efforts to prioritize content to be included in the summary by reducing the weight of terms that have already appeared in the first document set. ROUGE scores did not not improve for summaries generated with this approach. This suggests that tailoring summaries to prioritize new information over previously-seen information will require the development of alternative approaches.
- *De-emphasizing sentences in new documents that are semantically similar to previously-seen sentences when choosing content for an extractive summary does not improve summary accuracy or information value.* Section 4.4.3 described efforts to exclude previously-seen content from extractive summaries that highlight new information by comparing the sentence embeddings in both sets of documents. Sentences from the second document set with embeddings similar to from the first document set were excluded from the extractive summary. This approach was sensitive to the method used to generate sentence embeddings, though ROUGE scores did not improve for summaries generated with this approach regardless of embedding method. This suggests that tailoring summaries to prioritize new information as compared to a set of previously-seen information will require an alternate approach for leveraging sentence embeddings to prioritze new content.
- *Emphasizing new documents when generating an extractive summary does not improve summary accuracy or information value.* Section 4.4.4 described weighting terms in documents with more recent timestamps more heavily than terms in older documents as a method to influence extractive summaries to favor new information. ROUGE scores did not improve for summaries generated with this approach. This suggests that tailoring summaries to prioritize new information will require an alternate approach to incorporating timestamp information into the process.
- *Filtering sentences to include in an extractive summary based on sentence sentiment does not improve summary accuracy or information value.* Section 4.4.4 described efforts to generate extractive summaries that reflected the sentiment of the source documents. Sentences with sentiment values lower than the overall positive-to-negative sentiment ratio of the source documents were excluded as candidates for the extractive summary. ROUGE scores did not improve for summaries generated with this approach. This suggests that sentiment might not be a useful measure by which summaries are tailored for a TLDR, or that an alternate approach is required to incorporate sentiment information in the summarization process.

Does applying text summarization techniques to transcriptions of noisy audio generate summaries with any level of accuracy or information value?

Extractive summaries of noisy audio transcripts provide limited information value. As shown in Section 4.5, the summaries of automatically transcribed Nixon White House recordings generated using the extractive summarization tool `occams` contained little informative content and were often incomprehensible. This was the case regardless of the segmentation method used to divide transcripts into smaller, potentially more meaningful, portions. Such summaries would not be useful to include in a TLDR.

Semantic clusters of noisy audio transcript segments can provide insight into common topics within those transcripts. While segmenting noisy audio transcripts into smaller portions was not useful for generating informative summaries, clustering the segments did provide some information value. Converting the segments into sentence embeddings and clustering those embeddings created semantically similar groups ("clusters") of transcript portions. By reviewing those clusters, it was possible to identify some topics that occurred throughout the transcripts. This type of semantic information could be useful metadata to include in a TLDR to help the user assess the potential value of noisy audio data.

What are effective methods for generating personalized text summaries of data contained in knowledge graphs?

Template-based summarization of cyber knowledge graphs can be meaningfully personalized and provide a text-based format for consuming information contained in knowledge graphs. As demonstrated in Section 4.6, users can define the manner in which nodes and edges in a knowledge graph should be organized for summarization, along with the order in which information from the knowledge graph should be presented. Summary readers would still require familiarity to interpret details provided in the summary, such as specific entity names and their significance, but would not need the ability to directly query a knowledge graph to identify that information.

Automatically contextualizing detailed information requires mapping between low- and high-granularity data. For example, as noted in Section 4.6, the STIX knowledge graph format needs further enrichment to enable automated traversal between low-level data and higher-level concepts. Using the current STIX format as a knowledge graph requires manual effort to connect granular network traffic-type data to higher level conceptual data that provides context for behavior within a network. Extending the STIX knowledge graph to enable automated traversal could allow for quicker identification of larger scale trends across different networks. Similar challenges in automatically contextualizing detailed information might occur in other domains where knowledge is represented at varying levels of granularity.

6 Concluding Remarks and Future Work

In this report, we presented selected results compiled from the 2022 Summer Conference on Applied Data Science (SCADS). We made foundational progress on problems associated with the grand challenge and creating tailored daily reports (TLDRs) for individual knowledge workers. The work corresponded to research questions aimed at understanding TLDR user needs, identifying information to report in a TLDR, and summarizing that information for inclusion in the TLDR. While investigating user needs, we focused on the types of information knowledge workers interact with and how that information flows through their respective workflows. This understanding will help to inform TLDR design to optimize the TLDR user experience.

While investigating ways to identify and summarize information for a TLDR, we identified existing technologies that might be incorporated into a TLDR system, including neural recommendation models and extractive summarization tools. In some cases we tested those technologies to determine their potential utility or effectiveness in various TLDR use cases, while in other cases we sought to expand the current capabilities of those technologies to make them more suitable for a TLDR system. This included creating proof-of-concept systems that mimicked components of a future TLDR system and improving upon existing resources that could be used in generating a TLDR. We also determined some techniques which were *not* useful for a TLDR use case, which is valuable knowledge as we transition into eventual TLDR development.

We identified various areas for future work, ranging from short-term activities that extend the work described in this report to longer-term research related to the grand challenge. Short-term efforts on **understanding TLDR user needs** should focus on creating a common coding scheme as we complete another iteration of coding on interview notes, described in Section 2.2. This would enable more technical rigor and theory development, per the grounded theory approach. At that point we could conduct a systematic analysis of the interview data. By focusing the interviews on information passing within and outside of an intelligence organization, we hope to be able to identify similarities and differences in various information flows that will help inform the content decisions the "tailored" part of the TLDR would entail. For instance, mapping *types of input information* to *types of output* can help uncover what content an analyst needs that could be provided in a TLDR. We also recommend exploring other research questions that we might be able to answer by using the data developed during this study. Additionally, triangulating findings across the interview data and other related data sets, such as the VAST and University of Kentucky data sets may help bridge the gap between how analysts behave in a controlled environment versus in their regular workflow.

Immediate next steps in **identifying content** to include in a TLDR should focus on running models, retraining models, and improving model efficiency. This would include scoring candidate articles using the DKN models mentioned in Section 3.1.2 to fully compare performance across the DKN and NRMS models. Furthermore, we recommend implementing the Enhanced NRMS model, which is the updated version of the NRMS model we tested during SCADS, and tuning hyperparameters for the NRMS and DKN models. To investigate whether efficiency gains are possible, we recommend updating the code from this work to use TensorFlow's `model(input)` rather than `model.predict(input)` to run inference to see if that results in any improvements. Other shorter-term future work could include training the NRMS model on the VAST/University of Kentucky data sets, which provide user interaction data from a simulated knowledge worker scenario. This would help us understand how the NRMS model translates to data from other use cases and domains, which could inform eventual TLDR design and architecture decisions. We do not recommend further pursuing the unsupervised anomaly detection algorithms explored here as a method for identifying content to report in a TLDR. Instead, we recommend exploring other methods for identifying changes in data to report in a TLDR, particularly any approaches that incorporate user-provided input on what type or magnitude of changes to flag.

In the area of **summarizing content** to include in a TLDR short term efforts should include measuring different aspects of the summarization task and incorporating contextual information. For example, we recommend measuring the performance of the updated coreference resolution process described in Section 4.3 to compare to the original and other existing coreference resolution solutions. We also recommend incorporating the updated coreference resolution code as a preprocessing step in the various experiments to quantify its impact on summary quality and informativeness. One experiment could be to compare summaries generated by pipelines that do and do not apply coreference resolution prior to summarization leveraging topic-applicable term weights to see if the resulting summaries are

higher quality or more informative summaries for a TLDR user.

To incorporate contextual information into summaries, we recommend extending the graph-to-text proof of concept to summarize a knowledge graph that contains provenance information for the information contained in the graph. The provenance could be used to provide context for information in the graph and generated summary. We also recommend creating more sophisticated and/or tailored sentence-generation templates for use in a pilot system, such as in the work by [MdSM20], or incorporating advanced language modeling techniques for language generation, such as in the work described in [FGI+20]. This might include further investigating abstractive summarization, which would require employing methods to identify or prevent "hallucinations" to ensure that the abstractive summaries only include information that is contained in the knowledge graph. Another effort we recommend related to improving summaries of knowledge graphs is tuning existing named entity recognition (NER) models for different relevant domains such as cyber. Improved domain-specific NER models available would enhance the accuracy of knowledge graph creation for a specified domain of interest, enable us to generate knowledge graphs based on additional text data sources, and reduce reliance on existing data sets for TLDR research.

We do not recommend continuing the approach described in 4.5 to apply text summarization methods to transcripts of noisy audio, as this approach did not reliably produce summaries of any notable information value. Instead we recommend exploring the value of incorporating audio and/or transcript metadata into a TLDR to provide context for the audio. For example, the Nixon tapes provide information about the location where the conversation occurred, and often include information about the day and time the conversation happened as well as who participated in the conversation. It could also be possible to generate other metadata, such as descriptions of non-speech noises on the tape and speaker sentiment or tone, using other audio processing techniques. We also recommend further exploring clustering sentence embeddings to identify recurring topics throughout the data set that might not be obvious, which could also serve as metadata about audio data. All of this metadata, if deemed to be useful and reliable, could be incorporated into a template that describes the contents of an audio clip more effectively than an automatically-generated summary. 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 approaches might also utilize tools that were not available at the time this work was conducted.

As longer term efforts to extend this work, we recommend measuring the quality of the text summaries of knowledge graphs. This presents some challenges as there are no existing summaries of the data with which to calculate something like ROUGE scores, nor is there a source document that could be leveraged in an existing question-answering evaluation pipeline. Some approaches to measuring graph-to-text summary quality might include having humans generate reference summaries to enable ROUGE score calculation or adapting question-answering approaches to work with knowledge graphs as information sources. This would allow us to more easily quantify the impact of improvements to the knowledge graph generation on graph-to-text pipelines, e.g. tuning domain-specific NER models on graph-to-text summaries.

Another promising avenue for enabling quality and effectiveness measurements within a TLDR system could be to create a data set that is appropriate for research spanning multiple TLDR components. The data sets used throughout this work were mostly compatible with experiments for one or two TLDR components rather than the complete TLDR workflow. The MIND data set was useful for recommender system research, but was limited in the way it could be used for automatic summarization work. We propose a data set constructed from academic papers from multiple topics and domains, such as from *arXiv* or a similar repository. The data set would contain the full content of a paper, including the abstract, along with the complete set of prior work referenced by that paper. We envision that such a data set could be useful for summarization work, and propose a corresponding experiment in which we try to automatically generate the abstract or executive summary of a paper by summarizing the novel content in the paper as compared to its references. Additionally, we could run a study in which a number of analysts and researchers answer questions using the data set and summarize their results. This user interaction data could then be used to enable quality evaluations of the generated summaries, as well as generate knowledge graphs for analysis scenarios.

To achieve the goal of automatically creating a TLDR, we recommend building a comprehensive prototype system. The pilot TLDR could incorporate a proof-of-concept recommender system to present article recommendations to users along with contextual information, such as article metadata

or information about other users' interactions. Assuming there is some kind of graphical user interface (GUI) for the TLDR system, it could include LIME-style explanation visualizations to provide users with insight into why certain articles were or were not recommended. Other supplemental information to incorporate into the protoype interface might include NLI predictions as to whether information supports or contradicts previously-seen recommendations.

A prototype TLDR system would offer a mechanism for users to interact with different versions of the underlying recommender and summarization models, and gather information about their utility. For example, the recommender system model could be retrained using the article abstracts or full MIND articles, then incorporated into the prototype system and compared to the performance and efficiency of the recommender models described in Section 3.1. Similarly, the NLI model from Section 3.1.4 could be retrained using additional data and used to generate inference information for full articles, which could also be incorporated into a pilot TLDR system and compared with previous versions. Other available NLI data sets include Multi-Genre Natural Language Inference (MNLI), Question Natural Language Inference (QNLI), and Recognizing Textual Entailment [ZWZ+19]. We also recommend that the prototype incorporates summaries of the recommended articles, such as those generated by `occams` or a graph-to-text tool as described in Section 4.6. For example, prototype users could select specific portions of a knowledge graph to summarize or update the template to prioritize certain types of information in the generated summary.

A pilot TLDR system would also enable user testing on different aspects of the TLDR, which would be particularly valuable in a classified environment on operational data. By observing user interaction with a TLDR, we could better understand the implications of model behavior metrics on TLDR utility. In this work, we computed accuracy metrics for recommender models, but we do not know whether a TLDR user's experience would be drastically different if the TLDR system incorporated a faster but slightly lower-accuracy recommender model over a higher-accuracy but less efficient model. User testing could help to determine the significance of the novelty and diversity scores we examined for recommender systems, as well as provide insight on whether including things like explanations from LIME or another mechanism affects users' perceptions of system trustworthiness. A pilot TLDR system would also be helpful in running experiments to address concerns about the potential for creating feedback loops between a recommender model, user behavior, and other TLDR components. While such experiments and associated interface details would need to be carefully designed, there are many opportunities to incorporate user controls, evaluative surveys, direct preference elicitation, and conversational interfaces that can help to mitigate feedback loops. We could leverage the results from the HCI study to inform some of these design decisions, with particular emphasis on findings that provide insight on how best to incorporate personalization into the system.

Acknowledgements

We, Sean L. and Liz R., as the SCADS Leads, would like to thank the many individuals and organizations whose enormous contributions of time, effort, funding, and thoughtful consultations were key to bringing SCADS to life. Primary amongst these is former Chief Data Scientist at the National Security Agency, Dr. J David Harris. Dr. Harris's long-term vision and compelling advocacy were the critical driving forces behind the creation of SCADS, and continue to serve as inspiration for everyone involved. The event was made possible by funding which was provided by Operations, Cyber Security, Data Science, and Research organizations in the Intelligence Community to support staffing, travel, venues, hardware, software licenses, and many other miscellaneous expenses. Many thanks to these organizations for believing in this mission. Also many thanks to the leadership team of the Laboratory for Analytic Sciences at North Carolina State University for their continuous guidance, assistance, and patience as we navigated the many hurdles of organizing an inaugural event of this nature together.

To Ken T., thank you for your immense contributions throughout both the planning and execution phases of this conference. Your assistance in refining the grand challenge and problem book was absolutely essential, and you acted as a leader in representing user/mission interests. Even in light of your leading role, your lesser-known/behind-the-scenes contributions probably outweighed these. Thank you for a job well done.

To Stephen S., your various experiences with events of this nature was invaluable, as was your leadership assistance. SCADS 2023 is in good hands.

To Heather B., SCADS would simply not have been possible without your efforts and contract expertise.

To everyone who shared what a "day in the life" in their role is like with the participants, formally and informally, and how the grand challenge applies to their work, Ken T., Patti K., Michelle W., Susie B., Al J., Sue K., Tina K., Pauline M., and Jacque J., thank you for the insightful presentations and the ongoing interaction throughout the conference.

To those who made sure we had a place to run code and helped troubleshoot so it actually ran, John S., Brent Y., and Troy W., thank you for being so responsive and patient.

Thank you to Christine B. and Jascha S. for helping us obtain IRB approval for the interview study.

To those who helped organize and host the pre-event workshop, Stephen S., John S., Maria G., Robin C., Addie K., Liz C., Mark G., Cliff J., George C., Brenda P., Dustin A., Amanda P., and John C., thank you for sharing your time and incredible expertise.

Many thanks to all of our speakers who shared experience, perspective, and in many cases wisdom, regarding operational challenges, grand challenge motivations, the current and future technological landscape, and the necessity of collaboration. We are extremely grateful that you were able to make time for SCADS in your busy schedules, and count ourselves fortunate. Please keep SCADS in mind in future years.

To everyone who helped organize some unjustly less heralded aspect of SCADS, thank you: Christine B., Kelli C., Jacque J., Paul N., Matt S., John S., Stephen S., Katherine K., Brent Y., and Maggie E., you administrated organizational and staffing activities, set up infrastructure to use, planned activities to attend, navigated HR and security policies, developed event communications, facilitated feedback, and recruited participants.

Thank you to Alyson W., Amy G., Stephen S., Matt S., Jordan C., and Katherine K. for editing and proofreading various iterations of this report.

To Emily, you were the cornerstone holding everything up. You entered the planning process a short time before the conference kicked off and brought your organization, attention to detail, and boundless energy to the process. You made sure we had conference halls and accommodations, fielded countless questions, kept us on schedule, and did more than can be mentioned here. Thank you for everything you did before, during, and after SCADS - with a smile on your face and a cup of coffee in hand.

Finally, we could not have pulled this off without the enthusiasm and energy of the first SCADS cohort. You chose to spend eight weeks of your summer engaged with fellow researchers and all arrived ready to dive into the grand challenge. You shared ideas and expertise, encouraged each other, and demonstrated endless curiosity. You were flexible as we navigated almost everyone's first in-person conference in years. The camaraderie in this group was beyond anything we ever imagined and it truly helped bring this conference to life. We extend a huge thank you to each and every one of you:

Aaron A., Abhishek Kulkarni, Alex G., Amanda P., Andrew E., Ashley W., Ben Strickson, Blake C., Brandi H., Cas Laskowski, Cody Taylor, Cory S., Deborah Littlejohn, Donald Honeycutt, Eric Ragan, Grant Forbes, Ilana B., James Hardaway, Jane A., Jeffrey D., Jeremy Block, John M. Conroy, Jonathan Stray, Joseph Kepler, Josh Sheinberg, Josh T., Joshua L., Julia Y., Matthew K., Natalie Kraft, Neil Molino, Nick Gawron, Priscilla C., R. Jordan Crouser, Rebecca Jonas, Sharon Chu, Sophie T., Suzanne K., and Yancy Vance!

References

[20221] STIX Version 2.1, 10 June 2021.

[AG08] Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Comput. Surv.*, 40(1), feb 2008.

[AIH21] Md. Monowar Anjum, Shahrear Iqbal, and Benoit Hamelin. Analyzing the usefulness of the DARPA OpTC dataset in cyber threat detection research. In *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*. ACM, jun 2021.

[AWHK21] Rody Arantes, Carl Weir, Henry Hannon, and Marisha Kulseng. Operationally transparent cyber (OpTC), 2021.

[BAPM15] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[BCCAN20] Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. TweetEval: Unified benchmark and comparative evaluation for tweet classification, 2020.

[CAR18] Sumit Chopra, Michael Auli, and Alexander M. Rush. Abstractive sentence summarization with attentive recurrent neural networks, 2018.

[CCL+20] Lei Cai, Zhengzhang Chen, Chen Luo, Jiaping Gui, Jingchao Ni, Ding Li, and Haifeng Chen. Structural temporal graph neural networks for anomaly detection in dynamic graphs. *arXiv preprint arXiv:2005.07427*, 2020.

[CD18] John M. Conroy and Sashka T. Davis. Section mixture models for scientific document summarization. *International journal on digital libraries*, 19(2-3):305–322, 2018.

[CLVF19] M. Ariel Cascio, Eunlye Lee, Nicole Vaudrin, and Darcy A. Freedman. A team-based approach to open coding: Considerations for creating intercoder consensus. *Field Methods*, 31(2):116–130, 2019.

[Com21] National Security Commission. National security commission on artificial intelligence 2021 final report. Technical report, 2021.

[CWLL18] Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. Faithful to the original: Fact aware neural abstractive summarization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr 26, 2018.

[DCS12] S. Davis, J. Conroy, and J. Schlesinger. OCCAMS – an optimal combinatorial covering algorithm for multi-document summarization. In *12th International Conference on Data Mining Workshops*, pages 454–463. IEEE Computer Society, 2012.

[DMGP20] Michaël Defferrard, Martino Milani, Frédérick Gusset, and Nathanaël Perraudin. DeepSphere: a graph-based spherical CNN. *arXiv preprint arXiv:2012.15000*, 2020.

[DO08] Hoa Trang Dang and Karolina Owczarzak. Overview of the TAC 2008 update summarization task, 2008.

[DS18] G. Dineshnath and S. Saraswathi. Comprehensive survey on abstractive text summarization. *International Journal of Innovations Advancement in Computer Science*, 7(1), 2018.

[Dun93] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.

[EBE19] Matan Eyal, Tal Baumel, and Michael Elhadad. Question answering as an automatic evaluation metric for news article summarization. *Proceedings of the 2019 Conference of the North*, 2019.

[FGI+20] Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina, editors. *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*. Association for Computational Linguistics, Dublin, Ireland (Virtual), 2020.

[FWLX22] Alexander R. Fabbri, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. QAFactEval: Improved QA-based factual consistency evaluation for summarization, 2022.

[GF09] Dan Gillick and Benoit Favre. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 10–18, Boulder, Colorado, 2009. Association for Computational Linguistics.

[GG18] Som Gupta and S. K. Gupta. Abstractive summarization: An overview of the state of the art. *Expert systems with applications*, 121:49–65, 2018.

[GGM+19] Zheng Gao, Lin Guo, Chi Ma, Xiao Ma, Kai Sun, Hang Xiang, Xiaoqiang Zhu, Hongsong Li, and Xiaozhong Liu. Amad: Adversarial multiscale anomaly detection on high-dimensional and time-evolving categorical data. *arXiv preprint arXiv:1907.06582*, 2019.

[GHG+16] Vijay N. Gadepally, Braden J. Hancock, Kara B. Greenfield, Joseph P. Campbell, William M. Campbell, and Albert I. Reuther. Recommender systems for the department of defense and intelligence community. *Lincoln Laboratory Journal*, 22(1):74–89, 2016.

[HBCB21] Pranabjyoti Haloi, M. K. Bhuyan, Dibyajyoti Chatterjee, and Pooja Rani Borah. Unsupervised story segmentation and indexing of broadcast news video. *Multimedia tools and applications*, 2021.

[Hea21] Aidan Hogan and et al. *Knowledge Graphs*. Springer CHAM, 2021.

[HKG+15] Karl M. Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *Advances in Neural Information Processing Systems (NIPS)*, (28):1684–1692, Nov 19, 2015.

[Joh05] Rob Johnston. *Analytic culture in the US Intelligence Community: An ethnographic study*. Central Intelligence Agency, 14 edition, 2005.

[K.20] Dhiraj K. Anomaly detection using isolation forest in python, March 2020.

[KP17] Matevž Kunaver and Tomaž Požrl. Diversity in recommender systems–a survey. *Knowledge-based systems*, 123:154–162, 2017.

[KSV+16] Danai Koutra, Neil Shah, Joshua T. Vogelstein, Brian Gallagher, and Christos Faloutsos. DeltaCon: Principled massive-graph similarity function with attribution. *ACM Trans. Knowl. Discov. Data*, 10(3), feb 2016.

[KVF13] Danai Koutra, Joshua T Vogelstein, and Christos Faloutsos. DeltaCon: A principled massive-graph similarity function. In *Proceedings of the 2013 SIAM international conference on data mining*, pages 162–170. SIAM, 2013.

[Lin04] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. page 74–81. Association for Computational Linguistics, July 2004.

[LREA16] Edgar Alonso Lopez-Rojas, Ahmad Elmir, and Stefan Axelsson. PAYSIM: A financial mobile money simulator for fraud detection. In *The 28th European Modeling and Simulation Symposium, EMSS 2016, Larnaca, Cyprus, 2016, Proceedings*, 09 2016.

[LSC21] Zhengyuan Liu, Ke Shi, and Nancy F. Chen. Coreference-aware dialogue summarization, 2021.

[Luh58] H. P. Luhn. The automatic creation of literature abstracts. *IBM journal of research and development*, 2(2):159–165, Apr 1958.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[MdSM20] Abelardo Vieira Mota, Ticiana Linhares Coelho da Silva, and Jose Antonio Fernandes De Macedo. Template-based multi-solution approach for data-to-text generation. In *Advances in Databases and Information Systems: 24th European Conference, ADBIS 2020, Lyon, France, August 25–27, 2020, Proceedings*, page 157–170, Berlin, Heidelberg, 2020. Springer-Verlag.

[Mer20] Mohamad Merchant. Semantic similarity with BERT, 2020. Accessed: 2022-08-01.

[MG14] Nikita Munot and Sharvari S. Govilkar. Comparative study of text summarization methods. *International Journal of Computer Applications*, 102(12):33–37, 2014.

[MM21] Marta Maślankowska and Paweł Mielniczuk. How to make an effective coreference resolution, 2021.

[MPMS19] Abhishek Mahajani, Vinay Pandya, Isaac Maria, and Deepak Sharma. *A Comprehensive Survey on Extractive and Abstractive Techniques for Text Summarization*, pages 339–351. Ambient Communications and Computer Systems. Springer Nature Singapore, Singapore, 2019.

[Mul22] Britney Muller. BERT 101 state of the art NLP model explained, 2022. Accessed: 2022-08-01.

[MWX+21] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Quan Z. Sheng, and Hui Xiong. A comprehensive survey on graph anomaly detection with deep learning. *CoRR*, abs/2106.07178, 2021.

[NGJ+19] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: Lessons and challenges. *Commun. ACM*, 62(8):36–43, jul 2019.

[NKI20] Rungsiman Nararatwong, Natthawut Kertkeidkachorn, and Ryutaro Ichise. Knowledge graph visualization: Challenges, framework, and implementation. In *2020 IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 174–178, 2020.

[Nol13] Bridget Nolan. *Information sharing and collaboration in the United States Intelligence Community: an ethnographic study of the National Counterterrorism Center*. PhD thesis, University of Pennsylvania, 2013.

[OP09] Tore Opsahl and Pietro Panzarasa. Clustering in weighted networks. *Social Networks*, 31(2):155–163, 2009.

[otDoNIa] Office of the Director of National Intelligence. U.S. Intelligence Community Budget.

[otDoNIb] Office of the Director of National Intelligence. What is the PDB?

[PC20] Theodoros Papadopoulos and Yannis Charalabidis. What do governments plan in the field of artificial intelligence? analysing national AI strategies using NLP. In *Proceedings of the 13th International Conference on Theory and Practice of Electronic Governance*, pages 100–111, New York, NY, USA, 2020. Association for Computing Machinery.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543,, Doha, Qatar, oct 2014. Association for Computational Linguistics.

[RD22] Shaina Raza and Chen Ding. News recommender system: a review of recent progress, challenges, and opportunities. 55:749–800, Jan 2022.

[RG19] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[RHB] Jennifer Reif, Michael Hunger, and Florent Biville. Fraud detection with the PAYSIM financial dataset, Neo4j graph data science, and Neo4j Bloom.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[Rud22] Sebastian Ruder. NLP-progress: Natural Language Inference, 2 2022.

[RVG+18] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Muller, and Marius Kloft. Deep one-class classification. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4393–4402. PMLR, 10–15 Jul 2018.

[Shn96] Ben Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE Symposium on Visual Languages*, pages 336–343, 1996.

[SK16] Sheetal Sonawane and Parag Kulkarni. The role of coreference resolution in extractive summarization. In *2016 International Conference on Computing, Analytics and Security Trends (CAST)*, pages 351–356, 2016.

[SOWC21] Asara Senaratne, Pouya Ghiasnezhad Omran, Graham Williams, and Peter Christen. Unsupervised anomaly detection in knowledge graphs. In *The 10th International Joint Conference on Knowledge Graphs*, IJCKG'21, page 161–165, New York, NY, USA, 2021. Association for Computing Machinery.

[TDM11] Gokhan Tur and Renato De Mori. *Spoken language understanding : systems for extracting semantic information from speech*. Hoboken, NJ : Wiley, 2011, Hoboken, NJ, 2011.

[TF21] CSIS Technology and Intelligence Task Force. Maintaining the intelligence edge: Reimagining and reinventing intelligence through innovation. Technical report, January 2021.

[TYEL18] Xian Teng, Muheng Yan, Ali Mert Ertugrul, and Yu-Ru Lin. Deep into hypersphere: Robust and unsupervised anomaly discovery in dynamic networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2724–2730. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[Vea21] Andrew Veal. Classifying Computer Processes in the DARPA OpTC dataset, 2021.

[WFMD21] John Wenskovitch, Corey Fallon, Kate Miller, and Aritra Dasgupta. Beyond visual analytics: Human-machine teaming for AI-driven data sensemaking. In *2021 IEEE Workshop on TRust and EXpertise in Visual Analytics (TREX)*, pages 40–44, 2021 2021.

[WQC+20] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, and Ming Zhou. MIND: A large-scale dataset for news recommendation, 2020.

[WWG+19] Chuhan Wu, Fangzhao Wu, Suyu Ge, Tao Qi, Yongfeng Huang, and Xing Xie. Neural news recommendation with multi-head self-attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6389–6394, Hong Kong, China, Nov 2019. Association for Computational Linguistics.

[WZXG18] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. DKN: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 1835–1844. International World Wide Web Conferences Steering Committee, 2018.

[XCL+20] Luguo Xue, Yan Chen, Minnan Luo, Zhen Peng, and Jun Liu. An anomaly detection framework for time-evolving attributed networks. *Neurocomputing*, 407:39–49, 2020.

[YS22] Jia Yu and Hongxiang Shao. Broadcast news story segmentation using sticky hierarchical dirichlet process. *Applied intelligence (Dordrecht, Netherlands)*, 1, 2022.

[ZKW+19] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTscore: Evaluating text generation with BERT. *CoRR*, abs/1904.09675, 2019.

[ZLL+19] Li Zheng, Zhenpeng Li, Jian Li, Zhao Li, and Jun Gao. AddGraph: Anomaly detection in dynamic graph using attention-based temporal GCN. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4419–4425. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[ZWZ+19] Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. Semantics-aware BERT for language understanding. *arXiv preprint arXiv:1909.02209*, 2019.

A SCADS Grand Challenge

Grand Challenge
Summer Conference on Applied Data Science
Laboratory for Analytic Sciences @ North Carolina State University

The Summer Conference on Applied Data Science (SCADS) is an annual 8-week workshop hosted by the Laboratory for Analytic Sciences. The overarching goal is to bring together industry, academic, and government professionals to collaboratively attack a grand challenge in the space of machine learning and artificial intelligence. The grand challenge (described below) is a somewhat lofty, 5-10 year R&D goal having the potential to greatly influence analysis practices of knowledge workers in all sectors. Stepping-stone problems along the way toward achieving the grand challenge offer near-term value potential.

A.1 Grand Challenge

Generate tailored daily reports for knowledge workers that capture information relevant to their individual objectives and interests.

A.2 Grand Challenge Overview

The essential goal of the challenge is to develop modern AI/ML capabilities to proactively provide individuals (or organizations) with information relevant to their particular needs. The vision of a "tailored daily report" is a fairly 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 Intelligence Community manually produces a report of this nature for the President every morning, summarizing new updates of intelligence information that the President needs to know. Conceptually, AI/ML technology could produce something similar for all of us.

A.3 Grand Challenge Description

A broad range of objectives and interests exist among knowledge workers, and these vary in time, sometimes rapidly. The first key element of this challenge is to model these objectives and interests in real-time. Many data types could potentially feed such a model, including those obtained through passive, active, or even interactive methods. Passive examples could include an individual's organization, their position/title, OS and application instrumentation data (documents accessed, documents produced, applications used, application-specific usage data, etc). Active examples could include custom-designed surveys to elicit responses to direct questions about the general goals and interests of individuals/organizations, or prompts to classify and log the purpose of their various activities. Conversational agents could even be leveraged to interact with users and obtain information regarding their current task. In the end, a model must be created which can be used to evaluate the relevance and value of information against the user's objectives and interests.

An enormous quantity of information is published daily, far too much for a knowledge worker to review. The second key element of this challenge is to utilize the objectives/interests models discussed above to discover and prioritize relevant information for the user across multiple types of data. Existing "off-the-shelf" recommender algorithms are commonly used in unimodal applications, and typically recommend entire objects. They are also typically limited to using in-domain data to model a user's interests. For example, a Netflix recommendation engine may recommend entire films, and only films, based only on what films the user has previously watched. Of course similar recommendation engines are in use for news articles, songs, books, and numerous other products. An extremely difficult challenge is to break multimodal information sources into constituent elements of knowledge, organize these elements, identify those of value to the user, prioritize and "recommend" them accordingly. Instead of recommending 3 news articles to a reader amongst the hundreds of potential options, perhaps an algorithm could identify and recommend pieces of information from across the hundreds of articles that the user needs to know. Perhaps it could even synthesize those pieces of information into a naturally understandable form. One potential concept is to employ a knowledge graph. It is reasonable

to imagine that a personalized recommendation algorithm could be designed over a knowledge graph to identify elements of interest. With the benefit of a richly informed user interest model, and the structure of a knowledge graph, it may be possible that a recommendation engine could present the most exciting scenes from a set of movies, the most intriguing sections from a set of books, or the most beautiful (in the user's eyes) passages from a set of songs. While not particularly exciting in regards to music and movies, in the knowledge worker's domain such a capability would be immensely useful.

The third key element of this challenge is to present the relevant information to the user in an effective and consumable manner. Innumerable options exist regarding the structure/format of information presentation, and in some cases the format is trivial (such as presenting a pre-existing document to a user). As alluded to in the introduction above, the vision of a "tailored daily report" is a clear and concise report presenting, perhaps summarizing, the most useful and interesting information available. The report could include source links, verbatim extractions of text/audio/visual components from source materials, or potentially text/audio/visual components reconstituted out of elements extracted across a variety of source materials.

B Problem Book

Achievement of the above challenge will require advances and/or developments in several areas of research (user modeling, personalized information retrieval, data summarization and synthesis), and will benefit immeasurably from dozens more. SCADS participants will be free to pursue any line of research reasonably believed beneficial to achieving the stated challenge. However, in the interests of fostering collaboration, adding some clarity to the types of granular and tangible research goals participants may choose to pursue, and simply getting the ball rolling, this document will later describe a set of "optional" research topics/objectives that have been vetted as being of particular interest to SCADS' stakeholders and supporters. These are each of great value in their own right as in most cases successful advances would offer short-term practical utility to knowledge workers, but they also offer great value as stepping-stone solutions toward achievement of the above challenge. We anticipate that many SCADS participants will choose to engage with the optional research topics/objectives described below, which will offer great opportunities for collaboration. However, we reiterate that participants will be free to chart their own research path to help achieve the challenge.

B.1 Suggested Research Topics and Objectives

The research topics/objectives described below were selected through an iterative process of brainstorming and conceptual development with numerous members of the intelligence analysis and research communities. Again, the below is not at all meant to be an exhaustive list of research goals that are of interest to SCADS, rather it is a small sample. Conceptually, the objectives fall into five classes (Data-focused, Human-focused, Presentation-focused, Engineering-focused, and Meta Analysis) which will likely remain true even for larger sets of research objectives in this challenge:

1. HUMAN-focused
    (a) Explainable recommender algorithms (particularly neural)
        i. Example: Develop and implement neural recommenders that are explainable so that user interests can be better understood and used to inform the creation of new content.
        ii. Example: Create a method of determining what an individual user may deem a "significant" change to a knowledge graph (or social network graph) for purposes of subsequently detecting such changes. For instance, design a feasible annotation experiment that enables users to label changes of interest in a knowledge graph (or social network graph). Examples of "changes" may be statistically relevant changes over time in such characteristics as a node group's size or activity, a modification in a group's aggregate node attributes, or anomalous edge presence/non-presence.
    (b) Personalized extractive text summarization and reader interaction measurements, evaluate readers' interaction patterns
        i. Example: Given a text corpus along with reader characteristics to determine text features of interest to the readers, broken down by individual and group preferences and trends. Features of potential interest include subjects in the text (who, what, when, where) or more general topics.
        ii. Example: Adapt or develop extractive text summarization methods which are personalized in the sense that a model of an individual user's objectives and interests is taken into account, creating a capability to extractively summarize a document for an individual reader's interest. For example, given a document corpus, a target document within the corpus, and the set of documents from the corpus that the user previously chose to read, estimate the level of interest the user will have in each portion of the target document.
    (c) Continual learning / AutoML on information retrieval tasks
        i. Example: Develop a continual learning method / autoML for the space of personalized, standing, open-source queries. Assume the user is willing to annotate their personal interest level in significant numbers of query results daily. Utilize the individual user's ongoing feedback to customize standing open-source queries in a way that optimizes query results to their individual interests.

(d) Analysis of user activity for the purpose of gaining insight into, but not limited to, behavior, pain-points, preferences, goals, and success.

i. Example: Develop a capability that can analyze analyst event logs, and provide a narrative/visualization/etc that identifies and describes common behaviors/workflows, pain-points, successful behaviors/workflows, and trends in behaviors/workflows. A simple use-case of such a capability is one that identifies that many analysts have started using a particular tool, and the tool is contributing to impactful analysis outcomes, however, it also observes that analysts are spending 25% of their time preparing data to input into the tool.

2. DATA-focused

(a) Knowledge graph research (temporal knowledge graphs in particular)

i. Example: Given a set of items (nodes, connections, anomalies, and cause and effect/-correlations amongst topics/objects, etc) that are known to be of interest to the user, develop methods to efficiently detect those types of items in a knowledge graph. Note: See HUMAN-focused-1-b

ii. Example: Given a text corpus with temporal references (ie daily news reporting, microblogs, etc), identify user-relevant topics, determine trends, and provide alerts to significant changes in those trends. Apply dynamic topic modeling (see BERTopic for example) and present results to a user in a way that is useful. Explore the use of temporal knowledge graphs with topic modeling.

iii. Example: Using available resources, create a recommendation engine that recommends subgraphs of a knowledge graph to users.

iv. Example: Given a knowledge graph, or a subgraph thereof, that was generated from a text document/corpus, develop a method of generating a narrative representation of the graph/subgraph. Conceptually, this would be similar to inverting the process via which the graph was generated.

v. Example: Adapt/develop/implement a clustering algorithm for a single type of node within a multipartite multidimensional graph.

vi. Example: Create a knowledge graph representing a technical environment (perhaps through a data set such as CRAWDAD), and changes thereof.

(b) Personalized abstract text summarization

i. Example: Given annotations noting a user's interest in various portions of documents in a text corpus (e.g. all articles within the last month's worth of a daily newspaper), generate a personalized summary of a target document (e.g. today's newspaper).

ii. Example: Create a personalized summary of a given knowledge graph. Note: A personalized version of DATA-focused-1-c above.

iii. Example: Develop state-of-the-art (non-personalized) abstract text summarization capabilities in the data domain of short, text-based evaluations. Given a set of short text-based evaluations (for example evaluations of a particular article, recipe, essay, or commercial product), create a capability to effectively summarize those evaluations into a single evaluation which fairly represents the entire set.

(c) Contextualize information of interest (for example the information already selected for inclusion in a TLDR)

i. Example: Given some information of interest (which could be a place, a human subject, a topic, a document, a sentence, or some else), and a text corpus with temporal references such as daily news reporting (AP, GDELT, etc), develop an information retrieval capability over the text corpus that provides context to the given information of interest.

ii. Example: Given a set of national/world news articles tagged as of interest to the user, identify a set of local/regional news sources and articles which have greater detail.

3. PRESENTATION-focused

(a) Visualize knowledge graphs, and significant changes

i. Example: Create a method which can effectively communicate a significant change in a knowledge graph (or a social network graph) to a user. This could be communicated visually, narratively, or otherwise.

(b) Develop more effective manners of presenting documents to individual users

i. Example: Given a quantified estimate of a user's interest level in each word/sentence/-topic in a document, explore possibilities of modifying the presentation of the document to benefit the reader. For example, highlighting, rearranging, summarizing portions, or any other alterations could be considered and experimented with.

(c) Summarize multimodal information items into a single, short, easy to consume report

4. ENGINEERING-focused

(a) Design a TLDR system having an extensible, modular structure

5. META ANALYSIS

(a) Establish new connections among knowledge workers with similar interests/tradecraft

i. Example: Create an automated capability to suggest that a knowledge worker converse with a colleague on a particular topic due to similar interests, or similar work roles/-tradecraft used. One option: Given a set of experimentally collected instrumentation and journaling data from a previously conducted HSR study, develop an algorithm capable of measuring the similarity between all study subjects' current interests and tradecraft methodologies.

(b) Study cross-affiliation collaboration effectiveness/ineffectiveness and develop recommendations to improve

C Interview Guide

Semi-Structured Interview Guide

This document lists topics for questioning in the semi-structured interview. The listed questions are examples and may be tailored to better match each participant and request elaboration on mentioned information. The order of question topics may be adapted for conversational flow of the interview.

Questions will not ask for personally identifying information, and we will not ask about details of subject matter data. We are not asking participants to provide names of datasets, specific entities, or names of specific groups of people they have worked with. Questions and data collection are designed to be limited to high-level information about how analysts understand their analysis process.

Materials Needed

- ☐ Printed Informed Consent Form (Gov and Non-Gov Versions)
- ☐ Opening and Closing Scripts
- ☐ Interview Questions (Printed or Electronic)
- ☐ Note Taking Template (Printed or Electronic)
- ☐ Papers and Pen (for the participant, interviewer, and note taker)
- ☐ Visible Clock (Current Time and Time Remaining) for the Interviewer and the Note Taker

Table of Contents

Overview of Time Allocation

Minutes	Topic
10	Opening Script
45	Interview Proper
5	Closing Script
60	**Total**

Opening Script (10 minutes)

Hi *interviewee*, thank you for taking the time to be a part of this interview. I am *interviewer's name,* and this is *note taker's name*.

I would like to begin by giving you an overview of what we'd like to achieve from our conversation. Our primary goal is to understand your experience as an analyst in the context of dealing with information and/or data.

Our conversation will not be highly structured. Please feel free to explore any and all thoughts that come to mind. I may interrupt you and try to get us back on track given we only have an hour together. We are providing some papers and markers you can use if you think something is easier to answer by drawing it out. You do not need to use those.

I would also like to preface our conversation by saying that I'm still learning the jargon of the intelligence community. Most words that I will use are meant to be taken in the broadest sense possible. If I use a word that means something very specific to you, please try to bring it up and clarify before we move forward in our conversation and I will adapt my language as I learn from you over the course of the next hour.

During the interview, *note taker's name* will be taking notes. We will not associate any personal identifiable information on our notes. However, it's possible that someone who knows you might still recognize you from our notes based on your answers.

These notes are only accessible to the research team from the HCI group here at SCADS. There is a possibility that in the future, other researchers will be able to access these notes. They can only do so after having complied with the imposed procedures of NC State, DoD, and any other institutions involved. We won't share these notes with the general public.

We might use isolated quotes based on your responses in future publications or reports. Also, if you make any sketches or drawings during the interview, we may take pictures of them and could potentially use them in publications. Again, we will not associate this to any of your personal identifiable information. We will do our best to avoid including anything that would let someone recognize you, but we can't guarantee this.

Finally, as a reminder, we are in an unclassified environment. I encourage you to consider potential metaphors to describe your work if need be, but you do not need to answer all of my questions. Just let me know if you are uncomfortable with a question and we will move on.

You can stop the interview at any point for any reason without consequence to you.

Please review the informed consent form in front of you (gov) (non-gov). This is the same document we shared with you during our initial contact. Do you have any questions before we proceed? Do you agree to voluntarily participate in this research study?

Interview Questions (45 minutes)

- Formulate follow-up questions as necessary to satisfy each goal as well as possible. Use your individual judgment to determine whether the goal has been satisfied.
- Begin with general questions then proceed with more specific questions to minimize the risk of biasing the respondents' answers.

Themes	Goals	Suggested Questions
Background :03	Understand where the analyst is coming from, and how his/her perspectives may have been formed	1. How long have you been with LAS? Is SCADS your first experience working with LAS? How long have you been an analyst? When did you last work as an analyst?
Topic Area / Experience :09	Understand what the analyst does/did at a general level, as well as on a day-to-day basis	What types of analysis work have you done/do you currently do? What were your role(s)? Did you manage other analysts?
Visualizing Information Flow :12	Model the flow of analyst information inputs/outputs	Drawing yourself in the center, we're going to create a diagram as a discussion aid.

Themes	Goals	Suggested Questions
Inputs :24	Understand the sources of triggers that lead to an analysis task and the interactions with those sources	Please draw and describe the things that may cue you to begin an analysis task? Where or Who do these triggers typically come from? <If customer> At a high-level, what types of customers are you serving? Are they typically external individuals, entire institutions, other analysts? What's the ratio? In what form do these triggers usually come in? Imagine that you received a *[mentioned trigger]*. How do you go about understanding what someone may want from that [*trigger*]? How much reframing is involved in the process to understand what someone really wants to know based on their request? How much interaction takes place between [*requester*] and the analyst in the process? How much redirection and discussion is there, if any? (i.e., are any changes made to the request, do you respond to [the requester] to get clarification, do they provide additional context?)
Outputs :36	Understand the outputs and how they're delivered from an analysis task	Describe the form(s) your output(s) (finished intelligence) take? Small pieces or all at once? How is it communicated? Is it structured, unstructured, textual, conversational? Imagine you are providing the same information to three different recipients - how do you change information output for each recipient? How do you adapt your output based on the specific needs or intentions of the customer?

Themes	Goals	Suggested Questions
Cooperation :45	Understand the kinds of interactions that the analyst has/had with others. Identify types, frequency, amount and targets of interactions (and any other features of the interactions that may seem relevant)	Explain how your position involves coordinating with others (e.g., analysts, teams or customers) For example, did analysts work in teams? If not, why did you work independently? What are the challenges you faced in coordination with others? Are these collaborations self-initiated or directed? Is there overlap with similar requests with other analysts? What methods do you currently use to understand the work being done by other analysts? Do you ever check the work of other collaborators/teammates?
Information Types [**process**] :51	Understand the data used in the processes used when doing an analysis task.	In a general sense, what forms of data do you work with to **answer** a [request/customer]? [**process**] Why do you work with these particular forms of data? What are the types of information that you deal with during your work? Textual? Physical and virtual conversations? Signals? HUMINT?
Process Variability :57	Understand the dynamic information flow during the sensemaking process.	Tell us about your workflow - when information comes in, is it one static piece or a continuous stream? How do you triage to decide what's important? How time sensitive is your work, and when is it due? What kinds of factors impact your process? In what ways? Is there a typical sequence for the process from a [trigger] to output? If not, how does it vary?
Feedback with Requester :60	Understand the interactions with the recipients of analysis outputs.	What form does feedback take? How does it impact your outputs? Do you ever send feedback to your [requester]? - To improve the efficiency of future analysis.

Closing Script (5 minutes)

All right, those were all the questions I had! Do you have any final thoughts?

As we wrap up, I'd like to request you not to disclose the contents of this interview with anyone outside of this room throughout the remainder of SCADS as this is an ongoing study.

Again, thank you for your time and valuable insight! I hope you have a great day!

Concept and Definitions

Feedback - This refers to the feedback you receive as an analyst doing your job but also the feedback you may provide to other people (e.g., customers, teammates, bosses, or others)

Input - the things you reference when doing an analysis task. (e.g., do you query systems, Do you speak to other people, are there other things you reference when working on a problem?)

Output - anything, both formal and informal, that you make when completing an analysis task. Preferably these are things that you make at the end of your analysis, but could also include items you make along the way.

Process - The steps you take and the pieces of Information used in the analysis (think of things like databases, stream data, network data, etc.)

Redirection - we're referring to the back and forth involved with a trigger? (e.g., Are any changes made to the request, do you respond to [the requester] to get clarification, do they provide additional context or details?)

Trigger - Something, in a broad sense, that may cue you to begin an analysis task. (e.g., a verbal request, a formal email, something else?)

D Guidelines for Initial Open Coding

This guideline is specifically designed for the first pass of the coding process. The following guide is a result of multiple iterations of a team-based approach for inductive analysis of qualitative data as inspired by [CLVF19]. It follows an open coding approach of semi-structured interview notes. The same pair has performed the open coding on five interview notes to improve the coding process and make it more systematic, thus leading to a consistent process.

D.1 Coding Assignments and Logistics

Ideally, each interview note should be assigned to at least two independent coders (one who was in the interview and another who was not in the interview). The rationale is to limit disagreements during the initial phase. After each person individually has done open coding, the pair performs open coding simultaneously and produces the coded note where consensus has been reached. Any disagreements should be resolved through discussion between the pair.

D.2 Coding Process

The objective is to do open coding in the by reading through each line of the notes. Think of the initial pass as flattening the data set and making sense of what transpired in the interview. These codes will serve as the data points for succeeding coding iterations. Also, do the coding in the context of not being able to access the raw notes (i.e., if you read your codes without looking at the raw notes, will they still make sense?). As much as possible, use the actual words from the notes. There are times coding in-vivo is the way to go. Add context or rephrase as necessary. Rephrasing is often employed if interviewees employed analogies in their answers or context needs to be addeded. The goal is to extract the essence of their answer. Use your best judgment in the process. This is the reason why one of the pair needs to be present during the interview as they can provide the context to some of the answers to ensure that the code is accurate. You want to capture multiple information as much as possible. This means it is possible to have multiple codes in a single line. Lastly, demographics are also part of the coding process, "interviewee is..."

While reading through the notes, determine whether the answer is somehow interview-specific or not or if it is something that can be generalized. When in doubt, err on the side of safety by assuming that it is interviewee-specific by including phrases that reflect it. For example, "interviewee perceives work should be easy." In a way, assume that it is subjective and is only something that is perceived by the interviewee. Always confirm if the code you are writing is something that you think is generalizable or not. There can be various levels in which a statement is true: interviewee, role, agency, etc. In the next iteration of the coding process (i.e., axial coding), where you now begin to look at multiple coded notes, it is where such generalizations can be made if the same themes are seen from multiple coded notes. For example, if we keep seeing multiple statements, it might be generalizable for the role. This is why it is critical to assume that any uncertainty should be treated as subjective.

Finally, this final guideline is still being validated, but ideally, you do not want to have identical codes for multiple instances. The idea here is that once a code has been identified, it does not make sense to have a duplicate. It does not have any impact on the analysis. If you are concerned on identifying emerging themes, it would take place in the next iteration when multiple interview notes are reviewed.

E The `recommenders` Python module

For building the recommendation engine, we utilized the recommenders python module available at https://github.com/microsoft/recommenders. It may also be installed via PyPi. We used version 1.1.1 of the module with Python version 3.8.10 and tensorflow version 2.8.0.

We found that there are pros and cons to using this module. Pros: It is cleanly written python code and contains Keras deep learning implementations that may not be worth re-coding. Additionally, it is open source, so the community may make improvements over time. Cons: The code is very specific to the MIND data set and may not generalize well to other data sets. The API is not well documented; understanding its use requires delving into complex python object oriented programming. And, we found a handful of bugs in the the module. Future SCADS teams may want to weigh the benefits of using this module vs copying the Keras portion of the module and writing their own data manipulation code.

Bug reports submitted by our team:

- https://github.com/microsoft/recommenders/issues/1800
- https://github.com/microsoft/recommenders/issues/1801
- https://github.com/microsoft/recommenders/issues/1803

Outstanding questions submitted by our team:

- https://github.com/microsoft/recommenders/issues/1807

F Model Serving Details

The NRMS model was saved in TensorFlow SavedModel format so it could be loaded, shared or deployed without the need to rebuild again from the original code.

Initially, we loaded the saved NRMS model in a Jupyter notebook through the TensorFlow keras API and tried to understand the predict method which was not well-documented.

Since we had expertise with TensorFlow Serving, we deployed and gained insight into the model using that capability. TensorFlow Serving is an easy to deploy serving system for machine learning models.

TensorFlow Serving was useful for viewing the actual input and output tensor shapes/types for the NRMS predict serving method and gaining a better understanding of the model. For example, the `saved_model_cli show` command displayed the following:

```
signature_def['serving_default']:
  The given SavedModel SignatureDef contains the following input(s):
    inputs['input_1'] tensor_info:
        dtype: DT_INT32
        shape: (-1, 50, 30)
        name: serving_default_input_1:0
    inputs['input_3'] tensor_info:
        dtype: DT_INT32
        shape: (-1, 1, 30)
        name: serving_default_input_3:0
  The given SavedModel SignatureDef contains the following output(s):
    outputs['activation_1'] tensor_info:
        dtype: DT_FLOAT
        shape: (-1, 1)
        name: StatefulPartitionedCall:0
  Method name is: tensorflow/serving/predict
```

We then crafted the following json file from test data in our NRMS predict notebook as input to the TensorFlow Serving predict method (Note: "input_1" contains an array of 50 arrays of 30 ints each):

```
{"instances": [{"input_1": [[5436, 19, 5228, 1070, 5437, 1133, 5, 836, 1, 12260, 5685, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [1443, 2312, 920, 224, 4308, 4307, 1360, 2483,
7777, 3830, 24, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [5166, 5167, 7509,
2823, 6106, 15, 522, 2022, 4258, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
..., [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]],
"input_3": [1325, 18368, 41, 3990, 3991, 5, 4155, 5185, 4761, 21, 10361, 18768, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]}]}
```

We ran the predict method as follows:

```
curl -d @nrms_input.json -X POST http://localhost:8501/v1/models/nrms:predict
{
    "predictions": [[0.442318797]]
}
```

Once we understood the model, we developed code within a Jupyter notebook enable model inference through the TensorFlow keras API. We ran predictions and saved off top-k predictions for users.

Being confident that our model was working correctly and with an understanding of the predict method's inputs and outputs, we moved on to Bailo.

To build, test and store a model in Bailo, we followed detailed directions at https://github.ncsu.edu/SCADS/BailoTemplate. Here is a summary of the basic steps we followed:

1. Run setup script to set up a project directory and Bailo virtual environment which includes required dependencies
2. Create the following files in the project directory:
   (a) requirements.txt - list of project dependencies (ex: tensorflow==2.8.0)
   (b) Model.py - model class wrapper that includes init, load, and predict functions
   (c) Optional: Python files required by Model.py (ex: preprocessing.py)
   (d) Optional: Model artifacts (ex: SavedModel dir for TensorFlow SavedModel)
   (e) TestModel.ipynb - notebook containing test code for model. This includes code to test the following:
      i. Model.py functions directly
      ii. endpoints for model API built locally through Seldon
      iii. endpoints for model API associated with Bailo docker container
   (f) Optional: Data files for testing (ex: input_data.json)
3. Test Model.py functions directly through TestModel.ipynb until working as expected
4. Build Seldon model API locally and test endpoints through TestModel.ipynb until working as expected
5. Create zip files for upload to Bailo:
   (a) code.zip contains 2a + 2b + 2c
   (b) model.zip contains 2d
6. Upload model details and zip files to Bailo
7. Deploy docker container in Bailo
8. Log into docker and pull docker container via instructions in its deployment screen
9. Run docker container via instructions in its deployment screen
10. Test Bailo docker container endpoints through TestModel.ipynb

```
Note: The Bailo input json file had to be in the following format:

{"data": {"ndarray": [
[ [[5436, 19, 5228, 1070, 5437, 1133, 5, 836, 1, 12260, 5685, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [1443, 2312, 920, 224, 4308, 4307,
1360, 2483, 7777, 3830, 24, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], ...,
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]],
[1325, 18368, 41, 3990, 3991, 5, 4155, 5185, 4761, 21, 10361, 18768, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
],...,
[ [[543, 190, 228, 170, 537, 113, 50, 836, 1, 12260, 5685, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [1443, 212, 920, 224, 4308, 4307,
1360, 2483, 7777, 30, 24, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],... ,
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]],
[1325, 18368, 41, 3990, 3991, 5, 4155, 5185, 4761, 21, 10361, 18768, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]] ] }}

The deployed model was tested as follows:

docker_predict_url = 'http://localhost:9999/api/v1.0/predictions'
```

```
# Test predict from json file through curl for single and multiple input data files
!curl $docker_predict_url -d @predict_data_bailo.json -H 'Content-Type: application/json'
!curl $docker_predict_url -d @predict_data_bailo_multi.json -H 'Content-Type: application/json'

{"data":{"names":["t:0"],"ndarray":[[[0.44231879711151123]]]},"meta":{}}
{"data":{"names":["t:0"],"ndarray":[[[0.44231879711151123]],[[0.5117172598838806]],
[[0.339644193649292]],[[0.5147402286529541]]]},"meta":{}}
```

G Beyond Accuracy Metric Definitions

Catalog coverage is the number of unique items in all top-k recommendation lists divided by the number of unique items in the full catalog. It can be useful for a system owner to know the percentage of their catalog that is being recommended to at least one user.

We define item **novelty** as the proportion of all users to whom the item is *not* recommended (considering all users' top-k recommendation lists). Novelty at k is the mean item novelty over all users' top-k items.

Recommendation **diversity** can be defined in many different ways (see [KP17] for an overview). In Table 3 we show the aggregate (mean) dissimilarity between pairs of top-k recommendation lists (comparing the items on each list). We used Jaccard similarity to determine dissimilarity ($1 -$ similarity). Because this is an n^2 computation, where n is the number of users, we sampled 100 users and ran 500 replications, averaging the diversity over all replications.

H Notes and Observations

Here we list observations and other notes generated throughout the course of this work. These notes and observations are not necessarily tied to specific research questions, but might provide further insight into this work.

- The worked described in Section 3.1.2 marked the first use of Bailo at LAS, aside from basic testing. We worked closely with an LAS software engineer familiar with the capability.
- When training DeepSphere we noted that the penalty terms seem to be skewed to numbers close to zero. Increasing *gamma* helped to more evenly distribute penalty terms, meaning more data points were predicted to be normal rather than anomalous.
- To enable the ability to designate a list of sentences to be excluded from the summary optimization process used in Section 4.4.3, we wrote the `occams` *SummaryExtractor* class.
- To enable running multiple text embedding and classifier experiments using different thresholds as described in Section 4.4.3, we refactored code to group sentences in a topic together and avoid embedding the same sentence multiple times. The `occams` *Document* class keeps track of document boundaries and maintains a flat list of sentences, which allowed us to group sentences. With this update, we were able to reduce the necessary computation by a factor of ten and make hyperparameter searches more viable for the sentence embedding methods.
- It would be helpful to have a data set that is applicable to mutiple TLDR components. For example, the MIND data set was useful for recommender system research but was limited in the way it could be used for automatic summarization work. It could be possible to construct such a data set from academic papers from multiple topics and domains, such as from *arXiv* or a similar repository. The data set would contain the full content of a paper, including the abstract, along with the full content of the papers referenced by that paper. We envision that such a data set could be useful for summarization work, and proposed a corresponding experiment in which we try to automatically generate the abstract or executive summary of a paper by summarizing the novel content in the paper as compared its references. Additionally, we could run a study in which a number of analysts and researchers are tasks to answer questions using the data set and summarize their results. This user interaction data could then be used to study questions on recommenders, enable quality evaluations of the generated summaries, and generate knowledge graphs for analysis scenarios.