UC Berkeley
UC Berkeley Electronic Theses and Dissertations

Title

Unsupervised Text Generation and its Application to News Interfaces

Permalink

https://escholarship.org/uc/item/308532p4

Author

Laban, Philippe

Publication Date

2021

Peer reviewed|Thesis/dissertation

Unsupervised Text Generation and its Application to News Interfaces

by

Philippe Laban

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Marti A. Hearst, Co-chair
Professor John Canny, Co-chair
Professor Laurent El Ghaoui
Associate Professor David Bamman

Fall 2021

Unsupervised Text Generation and its Application to News Interfaces

Abstract

Unsupervised Text Generation and its Application to News Interfaces

by

Philippe Laban

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Marti A. Hearst, Co-chair

Professor John Canny, Co-chair

Recent progress in automated text generation relies predominantly on the use of large datasets, sometimes requiring millions of examples for each application setting. In the first part of this thesis, we advance the field by developing novel text generation methods that balance the goals of fluency, consistency, and relevancy without requiring any training data. We achieve this objective on tasks such as text summarization and simplification by directly defining a multi-component reward, and training text generators to optimize this objective. The novel approaches that we introduce perform better than all existing unsupervised approaches and in many cases outperform those that rely on large datasets.

The second part of the thesis incorporates text generation into interfaces to help news readers navigate complex, unfolding news topics. We build a novel representation of news stories at scale and integrate new summarization, question generation and question answering modules into a chatbot and an automated interactive podcast. Human evaluations confirm that even though imperfect systems introduce friction for the user, they can serve as powerful tools to stimulate reader curiosity and help readers dive deeper into unfolding topics.

To my parents, my sisters, and Niyati.

Contents

v

List of Figures

List of Tables

Acknowledgments

In the last fives years, I have tremendously grown intellectually, thanks in large part to a community of advisors, co-workers, family and friends. I have learned, and become a better researcher and computer scientist thanks to the great examples I have had the opportunity to interact with.

First and foremost, thank you to my advisors during my time in Berkeley. Professor John Canny, thank you for your patience, and leading me to look at bigger, more challenging problems (Philippe, what would it look like to train a summarization model without any examples?). Professor Marti Hearst, thank you for shaping me into the researcher I am today, for believing and encouraging my vision of the world, disagreeing when needed and for the countless hours spent reviewing and improving our work. I can only hope to one day be half the mentor that you are.

To my committee members, I am grateful you took the time to be a part of this thesis' journey, and gave feedback along the way. Professor David Bamman, I learned a lot about NLP by interacting with you at the NLP Seminar, and always look forward to the astute discussions you lead on seemingly any topic. Professor Laurent El Ghaoui, your work on the Stat News project was inspiration for NewsLens in the early days of the Ph.D, thank you.

I want to thank my mentors at the Georgia Institute of Technology, who accompanied me in my first steps as a researcher. Professor Mary Ann Weitnauer and Van Nguyen, thank you for your encouragement and patience when teaching me the basic principles of research. Professor Evans Harrell, you taught me that research can be incredibly fun, thank you for the great conversations both in English and French!

I was lucky to interact with great researchers at Bloomberg, a veridic Disneyland for News+NLP. Thank you to David Rosenberg, Gary Kazantsev, and Konstantine Arkoudas, I learned a lot by working with you in Summer 2017, and hope the little piece of code we shipped to the Terminal remains useful. Thanks as well to Anju Kambadur, Ivo Dagan and Andrew Hsi, the work we started in Summer 2019 is some I am proudest of, and it is the second chapter of this thesis.

A sincere thank you to my mentors and friends at Microsoft Research. To Tobias Schnabel and Paul Bennett, thank you for initiating the contact of a truly great collaboration. From my presentation on a snowy day in Seattle, to the internship suddenly turned remote, the route was not easy, but your sharp opinion and ideas helped shape several chapters in this thesis. I also want to thank Dipendra Misra and Hal Daumé III for hosting me during Summer 2020, it was a pleasure working with you.

In Berkeley, I have had the chance to sit in a beautiful space, surrounded by brilliant people. As by osmosis, I learned to play with research ideas. During the five years, my desk barely moved in Bid, the Berkeley Institue of Design, a space with most grandiose windows. Dear Bidizens, together we perfected the art of procrastination, a necessary skill to the survival of the Ph.D. Thank you, Andrew Head for the e-bike road-trips and the many musical exchanges (and may the one-man-band prosper in Pittsburgh), Amy Pavel for the many lunches and inspiring work at the intersection of NLP and HCI, Eldon Schoop for the numerous baking adventures, including the Bid-hosted waffle challenge, Nate Weinman for the chats, Bobas and fun desk puzzles, Hezheng Yin for the swimming and skiing adventures (come back one day and finish that Ph.D. of yours!), to Elena Durán for the

Chapter 1

Introduction

Natural language processing (NLP), the sub-area in computer science tasked with the processing and generation of text, has been going through a period of rapid progress. This advance is fueled by the growing computational budgets to train large neural networks, as well as the curation of massive corpora of text, often containing billions of sentences.

Even though the abundance of data has led to improvements in the field across the board, there remain scenarios in which data is sparse, of low-quality, or costly to obtain. In this work, we make progress on creating *unsupervised* methods for several tasks in NLP, which do not rely on curated datasets and show that they can be competitive and sometimes outperform supervised methods.

In order to do more with less data, the methods we introduce rely on complex, hand-crafted rewards that encapsulate the essence of the task. We create rewards to automatically measure the quality of text summaries and simplifications, and then train modern text generators to optimize such rewards, through self-critical reinforcement learning, encouraging the writing of high-scoring texts.

As NLP technology matures, it is getting integrated into many user-facing products, most prominently through dialogue interfaces such as chatbots, auto-completion of emails, and the automatic captioning of images, and translation of texts. I focus my interest on the placement of NLP into interfaces for news reading.

Although today most news in consumed online through media devices [134], many popular news platforms are designed with the newspaper format in mind [58], and a focus on the latest breaking news. The atomic unit on a news website, a news article, is often organized into a headline, a lead image and static body of content. Although often rich in information, many news readers do not go past the information in the headline [134] which can be opinionated or exaggerated [38].

Furthermore, a person reading news should ideally take the time to get informed from several news sources for a given topic to gain an informed and nuanced opinion [60]. News aggregators, such as Yahoo and Google News, offer seamless access to lists of sources that cover a news articles, but leave the work of comparing and contrasting on the reader, requiring the reader's effort in selecting sources to read for each topic.

In this thesis, we present several prototype interfaces which were designed, built, and tested through human studies. We integrate state-of-the-art NLP technology into the interfaces, and
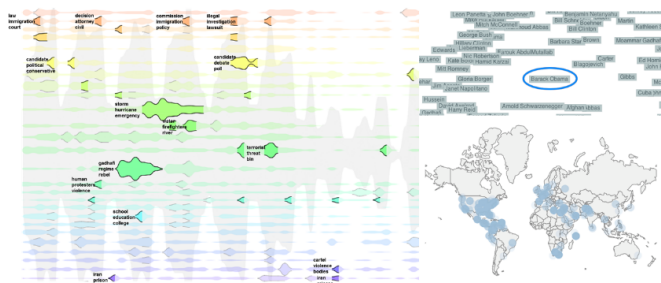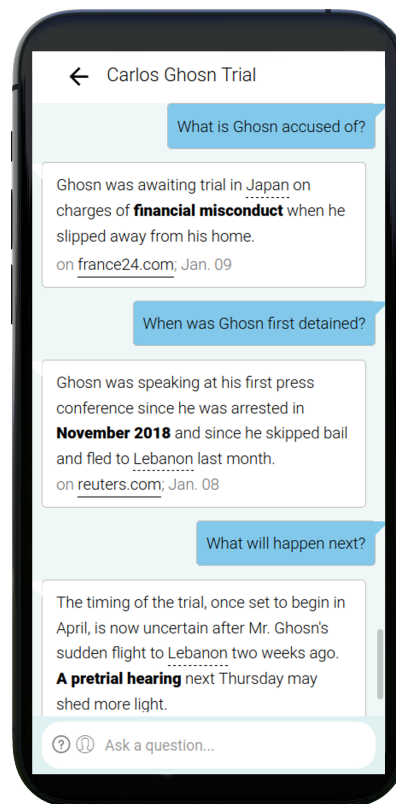
(a) Visualization-centric Interface   (b) Text-centric Interface

Figure 1.1: Prior work on news reading interfaces is predominantly visualization-centric [142, 33, 66]: users must interact with complex visualizations to access text. In this thesis, we build text-centric interfaces, using natural language for interaction.

compare with simpler baseline or human-generated content to assess the usefulness of using NLP in a news reading interface.

Some prior work has built advanced interfaces for the news domain, often involving panels of rich visualizations. Due to the abundance of text, the news articles are used to generate interactive visualizations which can be operated to access individual news articles. Interacting with such *visualization-centric* interfaces is non-trivial, and specialized training is often required for proper use.

In contrast with prior work, we build prototypes of *text-centric* interfaces for news reading. The interfaces are kept simple, and user-interaction occurs through a textual interface (spoken or written), facilitating the use for non-experts. We validate our designs through human evaluation, performed with crowd-workers without particular qualifications beyond fluency in English. In Figure 1.1, we show an example of the visualization-centric Leadline [33] interface, as well as screenshot of the text-centric NewsChat interface presented in this thesis.

Publication Venue	Publication Title	Author List
Workshop on Events and Stories in the News, ACL, 2017	newsLens: building and visualizing long-ranging news stories	Philippe Laban, Marti Hearst
System Demonstration at ACL, 2020	What's The Latest? A Question-driven News Chatbot	Philippe Laban, John Canny, Marti Hearst
ACL, 2020, Long Paper	The Summary Loop: Learning to Write Abstractive Summaries Without Examples	Philippe Laban, Andrew Hsi, John Canny, Marti Hearst
ACL, 2021, Long Paper	Keep It Simple: Unsupervised Simplification of Multi-Paragraph Text	Philippe Laban, Tobias Schnabel, Paul N. Bennett, Marti A. Hearst

Table 1.1: List of publications whose work overlaps with content on the thesis, each listed with a paper title and the list of co-authors.

The interfaces we build, such as a chatbot and an automatic podcast, rely on imperfect NLP components that can make factual mistakes that would be unacceptable in a product deployed to a large audience. The interfaces are built and tested as advanced research prototypes, intended to question the place of advanced NLP machinery, such as automated question-generation, answering and speech synthesis, in the activity of news reading.

The human evaluation we perform reveal that even though deep-learning based models indeed introduce imperfections which create friction in the user interaction, the technology can still benefit many. For example, questions generated and used as prompts can help a reader deepen their knowledge on a given news topic, and a majority of news readers opt to interact with news content and ask their own questions when given the opportunity, even when the automated answer can be erroneous.

The work in this thesis aims to provide evidence that as automatic text generation becomes more fluent and consistent, it can serve as a useful tool in helping citizens of the world stay informed on complex, unfolding news events.

1.1 Prior Publications and Authorship

Part of the content in this dissertation was published prior to the writing of the thesis, and details of all published work is listed in Table 1.1. Although I was the first author on all published work, wrote the code-base used in all projects, and was in charge of running experiments and human evaluation, I list the contributions of my co-authors. John Canny and Marti Hearst, my advisors during the Ph.D. program, were close collaborators throughout the Ph.D., and were instrumental in helping with conceptualizing, framing, and describing the work clearly. Work described in Chapter 2 was partly completed during an internship at Bloomberg, in collaboration with my mentor Andrew Hsi.

Finally, Chapters 3 and 4 are the result of a year-long collaboration with Tobias Schnabel and Paul Bennett from Microsoft Research through the BAIR Commons project. All co-authors gave written approval for the work they participated in to be included in this thesis.

To reflect my collaborators' contributions, I use the pronoun 'we' in Chapters 2-7.

1.2 Structure of the Thesis

The thesis is divided into two parts, each consisting of three chapters.

Part I (Chapters 2-4) focuses on unsupervised approaches applied to text generation tasks. Although some attention is placed in Chapter 3 on human evaluation, the main contribution of these chapters is in NLP, more particularly in methods for text generation. The target audience for these chapters is NLP researchers, and the writing assumes some background knowledge (e.g., the inner works of the Transformer [159] architecture).

In Chapter 2, we introduce the Summary Loop. First, we define a summary as a brief, fluent text that covers the main points of an original document. We design reward components to measure coverage and fluency, which can be optimized using reinforcement learning to train a summarization model. An important contribution lies in the reward design not requiring reference summaries to be computed, rendering the Summary Loop the first abstractive and unsupervised approach to summarization. We show that Summary Loop trained models outperform previous unsupervised methods, and approach but do not exceed in performance state-of-the-art supervised methods. The summaries produced are more abstractive than ones generated by supervised methods, and can be controlled for attributes such as length.

In Chapter 3, we extend the Summary Loop and adapt it to the domain of text simplification in the Keep it Simple procedure. We introduce novel rewards to measure text simplicity and adapt others from the Summary Loop, which we optimize jointly. Unlike summarization, text simplification is a nascent domain with less readily available parallel data, and we are able to show that a Keep it Simple trained model outperforms both unsupervised and supervised prior work, setting a new state-of-the-art. We further iterate on the design of human evaluation for text simplification, and produce a evaluation protocol with which we demonstrate that a participant can complete a reading comprehension task faster with simplified texts, compared to the original text.

In Chapter 4, we address a major limitation of the current text generation models (including the Summary Loop and Keep it Simple models): their lack of factual consistency with respect to their original document. We show that models trained for the task of Natural Language Inference (NLI), can be adapted to the task of inconsistency detection, as long as they are used at the appropriate textual granularity. We build a benchmark dataset regrouping the six largest dataset for inconsistency detection, and find that our Zero-Shot NLI-adapted model outperforms all prior work. Adding a trained convolution layer finetuned on synthetic data further increases performance on the benchmark, demonstrating the usefulness of NLI in adjacent fields. We perform thorough experiments to validate the approach and find the most compatible adaptation of NLI models to the inconsistency detection task.

In Part II (Chapter 5-7), we change gears to focus on applying modern NLP pipelines to the news domain, incorporating question generation/answering, summarization, simplification, and speech recognition/synthesis into news reading interfaces. The target audience for Part II are individuals interested in news consumption, or applications of NLP, and does not require advanced knowledge of NLP.

Chapter 5 sets the foundation for the interfaces by describing the NewsLens system. NewsLens consists of a database of around seven million English news articles that were collected over a period of 13 years (2009-2021), as well as a pipeline of NLP that processes the individual news articles. Individual news articles is first merged into an *event*: which we define as a group of news articles from different sources that describe the same physical event in space and time. Events are further assigned to a *story*, a timeline of related events over time. This organization is intended to enable *story-centered* interfaces, that can surface past related events to the reader, and combine content from diverse news sources.

In Chapter 6 we introduce NewsChat. Each story in NewsLens with a recent event is instantiated as a chatroom, which news readers can enter to read and converse about the news story. News readers can ask open-ended questions, which the system attempts to answer with an automatic extractive question-answering system. The system also recommends automatically generated questions, and a human evaluation study determines that recommending relevant questions can lead participants to read more deeply into the news story, when compared to not recommending questions, or recommending questions without selecting them by relevance.

In Chapter 7, we expand the chatbot into an automated interactive podcast we name NewsPod. Through automatic speech synthesis, we generate news podcasts by concatenating podcast segments each focused on a single news story from NewsLens. The podcast is composed by the listener, who stories to include in the podcast, and a target podcast duration in minutes. The listener can interrupt the podcasts during any segment to ask open-ended questions. Unlike the chatbot in which the content is built through interaction with the user, content in the podcast must be outlined ahead of time by the system with the desired duration in mind. We compose the podcast with three distinct synthesized voices, each taking on a specific role (e.g, questioner vs. responder), to simulate a conversation and reduce monotony. A large-scale human study finds that using several voices and setting the podcast in a conversational setting is favored by listeners, and that a majority of participants intervene to ask their own questions, when the podcast includes sufficient inviting pauses.

Finally, Chapter 8 summarizes the contribution of the thesis, and presents next directions for future work that can bring the presented work further and address current limitations.

1.3 Research Contributions

The key contributions of this thesis include:

- **The design and implementation of unsupervised rewards to measure coverage, fluency and simplicity for text generation tasks.** In Chapters 2 and 3, we design conceptually simple

rewards intended to measure a specific quality in generated text. The reward component designed follow several uniforming rules: (1) they can be computed without the need of a reference text and (2) computationally inexpensive, such that several hundred scores can be computed on a single machine per minute, permitting the use of the scores in RL-based unsupervised learning.

- **k-SCST, an improved RL-optimization method to train text generators to optimize multi-component rewards.** The optimization algorithm introduced in Chapter 3 is a simple improvement of the standard Self-Critical Sequence Training (SCST). In k-SCST, a larger number of text candidates are generated and scored avoiding a common issue of null-gradients which can occur and slow down training in SCST. Experiments validate that k-SCST stabilizes and speeds up training as the number of candidates is increased.

- **State-of-the-art text generation model for the news domain.** The Summary Loop model introduced in Chapter 2 outperforms all previous unsupervised summarization models on a common news summarization dataset, the Keep it Simple model in Chapter 3 outperforms all supervised and unsupervised prior work for text simplification, and the SUMMAC model of Chapter 4 sets a new state-of-the-art on the eponymous benchmark for inconsistency detection in summarization. These models are released publicly, with links available in the respective chapters.

- **A task-based human evaluation protocol for text simplification.** The protocol is based on the assumption that when completing a comprehension questionnaire about a text, reading a simplified version of the text leads to a speed-up in the completion time of the questionnaire, at equal accuracy. An experiment with the protocol confirms this assumption on several documents and simplification algorithm. The protocol can be used to evaluate and compare simplification models, and get a direct estimate of the usefulness of generated simplifications.

- **A benchmark for inconsistency detection in summarization.** The SUMMAC benchmark introduced in Chapter 4 puts together the six largest datasets of inconsistency detection for summarization, and standardizes them into a binary classification task. Because each of the datasets in the benchmark was produced with summaries of different models, on differing articles, and created under distinct procedures, the benchmark is more general than individual datasets.

- **The design and implementation of two text-centric, NLP-powered news interfaces.** Chapters 6-7 introduce the NewsChat and NewsPod systems, both interactive interfaces for news reading. Emphasis is put on scalability, as both interfaces function source the information in a large, evolving dataset containing more than 7 million news articles, and function in real-time with minimal delay. Details on the back-end system and processing steps are described in Chapter 5.

- **The evaluation of varying news interfaces, giving insights on which design elements add value to a user's news-reading experience.** The human evaluations performed are

conducted by including a system with all features, as well as several baselines with a specific feature removed. Participants are observed while freely using the interface in one setting, and usage signals are recorded. By measuring statistical difference in usage signals between variants of the interface, the effect of a feature on usage can be established. For example, recommending automatically generated questions in NewsChat leads to longer and more in-depth conversations for participants on average, and in NewsPod, generating podcast sections in conversational format with multiple voices leads to an increase in engagement and satisfaction.

Part I

Unsupervised Text Generation

Chapter 2

The Summary Loop: Unsupervised Summarization

2.1 Introduction

Summarization, or the task of condensing a document's main points into a shorter document, is important for many text domains, such as headlines for news and abstracts for research papers.

This chapter presents a novel unsupervised abstractive summarization method that generates summaries directly from source documents, without the aid of example summaries. This approach simultaneously optimizes for the following important properties of a good summary:

- **coverage** of the keywords of the document,

- **fluency** of generated language, and

- **brevity** of generated summaries.

One of the main contributions of this work is a novel method of inducing good coverage of important concepts from the original article. The coverage model we propose takes as input the original document with keywords masked out (see Figure 2.1). It uses the current best automatically generated summary to try to uncover the missing keywords. The more informative the current summary is, the more successful the coverage model is at guessing the blanked out keywords from the original document. A resulting coverage score is fed back into the training process of the summarization model with the objective of producing summaries with high coverage.

A second contribution is our unsupervised training procedure for summarization, the *Summary Loop*, which leverages the coverage model as well as a simple fluency model to generate and score summaries. During training, the procedure is conditioned on a desired summary length, forcing the Summarizer model to adapt to a length budget. Figure 2.1 shows Summary Loop summaries obtained for the same document under three different length budgets.

A third contribution is a set of specialized techniques employed during training to guide the model away from pathological behavior. These *guard rails* include a method for reducing repetition, for encouraging the model to complete sentences, and to avoid frame filling patterns.

The models trained through the Summary Loop outperform all prior unsupervised summarization methods by at least 2 ROUGE-1 points on common news summarization datasets (CNN/DM and Newsroom), and achieve within a few points of state-of-the-art supervised algorithms, without ever being exposed to any summaries. In addition, summaries generated by our method use 50% more summarization techniques (compression, merging, etc.) than prior automatic work and achieve higher levels of abstraction, reducing by almost half the gap between human-generated summaries and automatic summaries in terms of length of copied spans.

2.2 Related Work

Supervised Abstractive Summarization. Sequence-to-sequence (seq2seq) [152] models trained using teacher-forcing are the most common approach to abstractive summarization [104]. A common architecture is the Pointer-Generator [140]. Performance can further be improved by constraining the attention [47, 54, 164] and using pretrained Transformer-based language models

---

Original Document: **Chilean President** announced Wednesday that his country, which has been **paralyzed** by **protests** over the last two weeks, will no longer **host** two major international **summits**. [...] The President has now canceled the **hosting** of the **economic APEC forum** and **COP25 environmental summit**, which were both due to take place later this **year**. [...]

Masked Document: ____ ____ announced Wednesday that his country, which has been ____ by ____ over the last two weeks, will no longer ____ two major international ____. [...] The ____ has now ____ the ____ of the ____ ____ ____ and ____ ____ ____, which were both due to take place later this ____. [...]

Summary Loop [10 word constraint]: Pinera cancelled the APEC summit at Santiago.
Coverage Score: 0.22

Summary Loop [24 word constraint]: Pinera said Chileans have been canceled the hosting of the APEC summit, which was scheduled to take place in November.
Coverage score: 0.33

Summary Loop [45 word constraint]: Sebastian Pinera announced Wednesday that his country will not hold the APEC summit, which was scheduled to take place in Santiago. Pinera said that Chileans had been paralyzed by protests over the last two weeks.
Coverage score: 0.39

Figure 2.1: Motivating example. A document from CNN.com (keywords generated by masking procedure are bolded), the masked version of the article, and generated summaries by three Summary Loop models under different length constraints.

[88, 23, 39]. Through architectural changes, the training procedure remains constant: using a large corpus of document-summary pairs, the model is trained to reproduce target summaries.

Unsupervised Summarization. Most unsupervised summarization work is extractive: sentences deemed relevant are pulled out of the original document and stitched into a summary, based on a heuristic for a sentence's relevance [101, 6, 166]. Nikolov and Hahnloser [109]'s abstractive approach is partially unsupervised, not requiring parallel data, but only a group of documents and a group of summaries. In contrast, our work does not require any summaries, and is trained using only documents.

Radford et al. [128] summarize documents using a language model (GPT2) in a Zero-shot learning setting. The model reads the document followed by a special token "TL/DR", and is tasked with continuing the document with a summary. Our work is an extension of this work: we initialize our Summarizer model with a GPT2 and specialize it with a second unsupervised method.

Summarization and Q&A. Eyal, Baumel, and Elhadad [41] and Arumae and Liu [5] turn reference summaries into fill-in-the-blank (FIB) questions, either as an evaluation metric or to train an extractive summarization model. In this work, we directly generate FIB questions on the document being summarized, bypassing the need for a reference summary.

Scialom et al. [138]'s work stays closer to a Q&A scenario, and uses a Question Generation

module to generate actual questions about the document, answered by a Squad-based [129] model using the generated summary. We refrain from using actual questions because question generation remains a challenge, and it is unclear how many questions should be generated to assess the quality of a summary.

RL in Summarization. Paulus, Xiong, and Socher [119] introduced Reinforcement Learning (RL) to neural summarization methods by optimizing for ROUGE scores, leading to unreadable summaries. Since then, Reinforcement Learning has been used to select sentences with high ROUGE potential [22], or optimize modified versions of ROUGE that account for readability [118]. In all cases, the reward being computed relies on a reference summary, making the methods supervised. We craft a reward that does not require a target summary allowing our training process to remain unsupervised.

2.3   The Summary Loop

For this work, the definition of a summary is:

> "A summary is a <u>brief</u>, <u>fluent</u> text that <u>covers</u> the main points of an original document."

Brevity, fluency and coverage are the three pillars of a good summary. Under a length constraint, a good quality summary should contain as much information about the original document as possible while retaining fluent and coherent English.

Subsection 2.3 lays out the steps in the Summary Loop. Subsections 2.3–2.3 specify how each component is represented by a neural network. Section 2.4 shows how to train a summarizer model using this architecture in an unsupervised manner.[1]

Summary Loop Steps

Numbers in Figure 2.2 correspond to the following steps:

1. Summarizer receives a document D and length-constraint L, and produces a summary S fulfilling the length constraint.

2. Using a Masking Procedure, D is modified into a masked document M, where important words have been replaced with blanks.

3. Coverage receives S and M, and uses them to fill in each blank in M with a word, producing F. F and D are compared, and the resulting fill-in accuracy is called the Coverage Score.

4. Fluency receives S, and gives a Fluency Score based on its assessment of the quality of the Summary's writing.

---

[1]The code, model checkpoints and other resources are available at `https://github.com/CannyLab/summary_loop`.
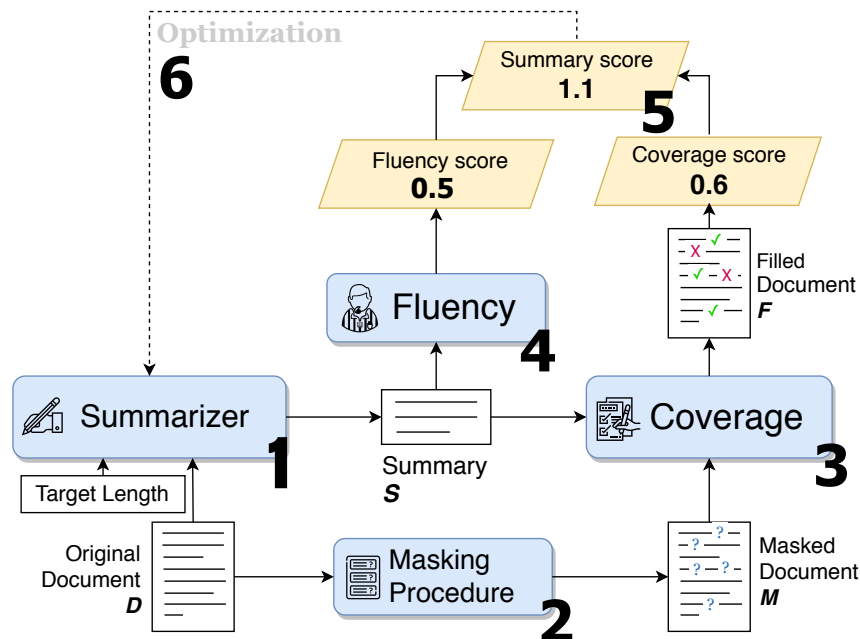
Figure 2.2: The Summary Loop involves three neural models: Summarizer, Coverage and Fluency. Given a document and a length constraint, the Summarizer writes a summary. Coverage receives the summary and a masked version of the document, and fills in each of the masks. Fluency assigns a writing quality score to the summary. The Summarizer model is trained, other models are pretrained and frozen.

5. The Fluency Score is added to the Coverage Score (as a weighed sum) into a Summary Score for the (D, S) pair.

6. Reinforcement Learning is used to train the Summarizer to produce summaries with high Summary Score.

The Summary Loop does not rely on the use of a target/reference/human-written summary, but only the summaries produced by the Summarizer model. The process can therefore be iterated upon without supervision from Summarization datasets.

Summarization Model

We use a Generative Transformer [128] as the model architecture of the summarizer. We make this choice for two reasons. First, Generative Transformers can produce text one word at a time, allowing the system to produce abstractive summaries. Second, we use the pretrained Generative Transformer to initialize the Summarizer.

Practically, the Summarizer first reads through the entire document, followed by a special *START* token, signaling summarization. The Summarizer produces a probability distribution over words in

---

Summarizer Architecture

GPT2-base: 12-layer, 768-hidden, 12-heads

Summarizer Initialization

GPT2 base model from Radford et al. [128]

Coverage Architecture

BERT-base: 12-layer, 768-hidden, 12-heads

Coverage Initialization

Pretrained model obtained in Section 2.3

Fluency Architecture

GPT2-base: 12-layer, 768-hidden, 12-heads

Fluency Initialization

GPT2 base model from [128], finetuned with Language modeling on news text.

Figure 2.3: The model size choice as well as initialization method for the Summarizer, Coverage and Fluency models in the Summary Loop. Each model leverages a pretrained Transformer.

its vocabulary, and a word is picked from the distribution and fed back as an input into the model. This procedure is repeated and halts either when the summary reaches a length constraint, or when the Summarizer produces a special *END* token. See

Figure 2.3 shows the model size and initialization model used for each of the Summarizer, Coverage and Fluency models.

Masking Procedure

The Masking Procedure decides on a set of keywords that are important elements in the document that should be recoverable using a summary. The keywords are replaced with blanks, indirectly indicating which information should be present in the summary. We use a tf-idf-based approach to decide on the set of masked keywords, as it is both simple and has been shown to represent word relevance to a document [130].

The masking procedure follows these steps:

1. We randomly sample 5,000 documents in the domain being summarized (e.g. News) as a training corpus,

2. The training corpus is tokenized using the tokenizer of the Coverage model. In our case, we tokenize with the Word Piece model of the BERT Base model [30],

3. We train a tf-idf transformation model using the tokenized training corpus using default parameters of scikit-learn's tf-idf implementation [120],

4. Given a document to be masked, we use the trained tf-idf model to produce a tf-idf for the document,

5. The words present in the document are ranked in decreasing order of tf-idf score, and the k words with highest tf-idf form the masking set,

6. All occurrences of the words in the masking set are replaced by a mask in the document, creating the masked document.

We select the k words with highest tf-idf score for the document to serve as the masked words. The k parameter represents a balance: if too many words are masked, the filling-in becomes impossible, but if too few are masked, the Summarizer model will not be encouraged to include sufficient content in its summary. Varying the value of k (10,12,15,20) yielded only small discernible difference in the Summarizers produced, and we use $k = 15$ in all our final experiments.

The masking procedure can be adapted to a specific domain. For instance, if summarizing financial documents, the masking procedure could systematically mask all numbers, encouraging the Summarizer model to add numbers to its summary.

Coverage Model

The Coverage Model receives a computationally generated summary and the masked document and attempts to fill in each blank word. The task of filling in blanks is similar to masked language modeling (MLM), used to pretrain BERT-like [30] models. In MLM, some of the words are replaced with a special $MASK$ token, and the model must use other information (unmasked words) to fill in the masked words. Because of the similarity to our task, we use a BERT-based neural network as the architecture for the coverage model. However, the coverage task differs from MLM in two ways. First, we modify the masking procedure: instead of masking a random percentage of the words (often 15% for BERT), we mask all appearances of the keywords selected by the masking procedure described in Section 2.3. Second, the input to the coverage model is a concatenation of the unmasked summary, a separator token and the masked document. The model can leverage unmasked information available in the summary to fill in the masked document. The Coverage Model is illustrated in Figure 2.4.
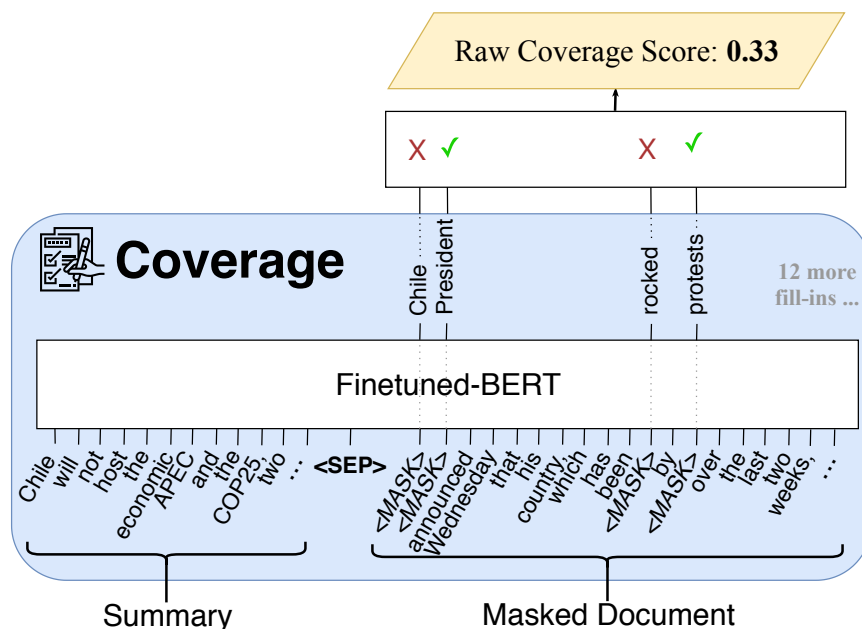
Figure 2.4: The Coverage model uses a finetuned BERT model. The summary is concatenated to the masked document as the input, and the model predicts the identity of each blank from the original document. The accuracy obtained is the raw coverage score.

Computing a Coverage Score

Using the masking procedure, we obtain $M = f(D)$, the masked document. The coverage model produces the filled document $F = g(M, S)$. *Raw coverage score* is the fraction of correctly filled in words in F. Let D_i, F_i and M_i correspond to the ith word in their respective document, I_M the set indices of words that have been masked. Then:

$$\text{RawCov}(D, S) = \frac{\|i \in I_M \text{ if } D_i = F_i\|}{\|I_M\|} \tag{2.1}$$

The model can use information in the unmasked (visible) words of M to predict the masked words. For instance, if the word "Chile" is visible, then "Santiago" would be a well-informed guess near the word "capital", which might not be masked out. This is undesirable, because coverage should account for what information the model can learn from the summary S, not what it can guess from the unmasked portion of D. To counteract this problem, we modify the raw coverage score by computing how much information the model can guess without the summary present, using an empty string summary: $F_\emptyset = g(M, \text{" "})$. We then normalize a summary's coverage by subtracting the empty string coverage from the raw coverage, leaving only filled-in words answerable using S, as shown in Equation 2.2.

$$\text{NormCov}(D, S) = \text{RawCov}(D, S) - \text{RawCov}(D, \text{" "}) \tag{2.2}$$

Summary Dataset	Summary Length	Raw Coverage	Norm. Coverage
Empty String	0	0.334	0
Headline	9.59	0.478	0.144
First 10 words	10.0	0.428	0.094
Newsroom	23.41	0.525	0.191
First 24 words	24.0	0.537	0.203
CNN/DM	45.75	0.726	0.392
First 46 words	46.0	0.649	0.315

Table 2.1: Analysis of the raw and normalized coverage of three existing human-written summary datasets, as well as first-k word baselines.

In a nutshell, raw coverage score answers the question: "What fraction of blanked words can be correctly filled in with this summary?" and normalized coverage score answers: "What is the increase in the fraction of blanks that can be correctly filled in with this summary, compared to having no summary?" In the rest of this thesis, Coverage Score refers to Normalized Coverage Score.

Training the Coverage Model

We train the Coverage Model once, and its weights are then fixed during the training of the Summarizer. In order to train the Coverage Model, we need pairs of documents (D) and summaries (S). However, we operate under the assumption that we do not have access to summaries (to keep the procedure unsupervised). In order to remove this dependency, we use the first 50 words of the unmasked document ($D[:50]$) as a proxy for document summaries. The Coverage Model is initialized with a trained BERT model [30], and trained using $(D, D[:50])$ pairs on the coverage task. Because BERT is already trained on the similar MLM task, the Coverage model is able to leverage knowledge accrued by BERT. The Coverage Model converges after roughly 5 hours of training on a Titan X GPU.

Analysis of Coverage

We present properties of the raw and normalized coverage through the analysis of existing human-written summary datasets. We focus our analysis on three datasets in the news domain: (1) a headline dataset obtained from common US news websites [83], (2) the Newsroom dataset [53], and (3) the CNN/DM dataset [104].

For each dataset, we take document/summary pairs and obtain raw and normalized coverage score through our Coverage model, reported in Table 2.1.

First, longer summaries obtain higher coverage scores: a CNN/DM summary with an average of 45 words can be used to fill in 73% of the blanks correctly, compared to 48% for a 9 word headline.

Across datasets, the correlation between summary length and raw coverage score is 0.56, confirming that longer summaries contain more information, according to coverage.

Second, we simulate the first k words[2] of the document as a summary. We use $k = 10, 24, 46$ to match average word length in the three datasets. For two of the three values (10 and 46), the coverage of human-written summaries is higher than the first-k word counterpart. This is remarkable: even though the summary is farther away lexically (i.e., is not a subset of the original words), it obtains higher coverage, demonstrating that the coverage model can account for reworded information.

Fluency Model

A model solely trained to optimize coverage has no incentive to write in good English, use punctuation, determinants or pronouns, as these are not words removed by the masking procedure. The objective of a Fluency Model is to judge the writing quality of the summary, independent of its coverage.

Given the right corpus, we argue that a language model's probability can be modified into a Fluency Score. Therefore, we adapt a language model into the Fluency Model.

We choose the generative Transformer [128] architecture for our Fluency model, as it can be trained into a powerful language model. Just as with the Summarizer, by using a standardized architecture and model size, we can make use of pretrained models. However, it is important for Fluency to fine tune the language model on the target domain, so that the Summarizer is rewarded for generating text similar to target content.

To produce a uniform Fluency Score, we linearly scale the language model's log-probability of a given summary ($LM(S)$) between an ideal value LP_{low} and a maximum value LP_{high}:

$$\text{Fluency}(S) = 1 - \frac{LM(S) - LP_{low}}{LP_{high} - LP_{low}} \tag{2.3}$$

This ensures that the $\text{Fluency}(S)$ is usually in the range $[0, 1]$. LP_{low} and LP_{high} are picked specifically for a particular language model, and ensure that the log-probability magnitudes of a specific language model do not affect the overall scores.

Table 2.2 provides examples from the Headline dataset of sampled headlines and their corresponding Fluency Score. The Fluency Score, a normalized language model log-perplexity, ranges from 0 to 1. Even though all these headlines are written by a human, the Fluency scores vary, with the higher-scoring headlines using more standard grammatical constructs. Note that the use of complex entity names does not prevent the model from obtaining a high Fluency score.

Summary Score

The final Summary Score is a weighed sum of the Coverage and Fluency Scores:

$$\text{SummaryScore}(D, S) = \alpha \cdot \text{NormCov}(D, S) + \beta \cdot \text{Fluency}(S) \tag{2.4}$$

---

[2]We choose the first k words due to the similarity to Lede 3 (first 3 sentences), a common baseline in news.

Example Headline	Fluency Score
Henry's Monaco recruit giant Brazilian Naldo for relegation scrap	0.16
Tesla shares dive after price cut, production numbers	0.41
French police arrest gilets jaunes protests leader Eric Drouet	0.59
Carlos Ghosn will appear in public for the first time since his arrest	0.75

Table 2.2: Example selected headlines and their Fluency score. The headlines were picked from a corpus of human-written news headlines. The average Fluency in the corpus is 0.479.

α, β are hyperparameters giving relative importance to Coverage and Fluency. We set $\alpha = 5, \beta = 1$ in all our experiments. Model choice, size, and initialization are summarized in Figure 2.3.

2.4 Training Procedure

We first outline the training procedure and then detail several guard-rail mechanisms used during training to prevent the Summarizer from learning pathological writing strategies.

Training with Reinforcement Learning

We use Reinforcement Learning to train the Summarizer component (agent), such that it achieves high summary score (reward). Note that the Coverage and Fluency models are frozen, and their weights are not trained. We make this choice as allowing Fluency and Coverage models to evolve could enable the models to coordinate and cheat.

We use the Self-critical sequence training (SCST) method [133], as it has been shown to perform well on similar text generation tasks optimizing BLEU for image captioning or ROUGE scores in summarization.

In SCST, the Summarizer is used to produce two summaries of document D: a greedy summary $\hat{S}$, using a decoding strategy that always picks the most likely next word, and a sampled summary S^s, picking the next word in the summary by sampling from the word distribution.

Summaries are scored using the Summary Loop:

$$\hat{R} = SummaryScore(D, \hat{S})$$
$$R^s = SummaryScore(D, S^s)$$

Then we minimize the following loss:

$$L = (\hat{R} - R^s) \sum_{i=0}^{N} \log p(w_i^s | w_1^s, ..., w_{i-1}^s, D)$$

Method	R-1	R-2	R-L	Coverage Score	Fluency Score	Brevity (avg words)
Baselines						
Human-written Summaries	100	100	100	0.392	0.612	58.5
$\mathbb{X}$ Lead-3 baseline	40.3	17.7	36.6	**0.421**	**0.656**	**84.0**
Supervised Methods						
Pointer Generator [140]	36.4	15.7	33.4	0.342	0.547	55.6
PG + Coverage [140]	39.5	17.3	36.4	0.377	0.508	61.7
Bottom-Up [47]	41.2	18.7	38.3	0.378	0.538	73.9
$PEGASUS_{BASE}$ [173]	41.8	18.8	38.9	-	-	-
$PEGASUS_{LARGE}$ [173]	**44.1**	**21.3**	**40.9**	-	-	-
Unsupervised Methods						
$\mathbb{X}$ TextRank [101]	35.2	12.9	28.7	0.370	0.612	49.62
GPT2 Zero-Shot [128]	29.3	8.3	26.6	-	-	-
Summary Loop 45	**37.7**	**14.8**	**34.7**	0.404	0.627	47.0

Table 2.3: ROUGE Results (F-1) on the non-anonymized CNN/DM test-set for supervised and unsupervised methods. Extractive methods indicated with $\mathbb{X}$. Our ROUGE scores have a 95% confidence interval of at most ± 0.30. Coverage, Fluency and Brevity (average number of words) included for systems where summaries are available, using Coverage and Fluency models from our work.

Where $p(w_i^s|...)$ represent the probability of the ith word conditioned on previously generated word, according to the model.

Intuitively, if $R^s > \hat{R}$, minimizing L maximizes the likelihood of the sampled sequence — which is desired because it outperformed the greedy summary — and increases expected reward of the model.

Training guard rails

During training, the Summarizer model learns pathological summarization strategies. We build training guard rails to detect the pathological behavior and penalize the model during training.

A guard rail has a binary effect: if a pathology is detected in a summary, its Summary Score is reduced by a penalty amount δ. We use $\delta = 2$ for all experiments. We found three training guard rails to be useful: No-repetition, Finish-your-sentence, and No-frame-filling.

No-repetition

A common problem in neural text generation is repetition of text. Based on the observation that 3-grams seldom repeat in common summarization datasets, the "No-repetition" training guard rail raises a penalty on a summary when it contains any repeated 3-gram.

Finish-your-sentence

When generating a summary, the model can either produce the END token, or generate a number of words up to the length constraint. We observe that if the model does not produce the END token, it often generates partial sentences, which is undesirable. Because we want to encourage the model to generate an END token, the "Finish-your-sentence" raises a penalty if a summary has no END token.

No-frame-filling

During training, the model sometimes learns to overly rely on sentence patterns that achieves high reward as a one size fits all summary. In one example the model learns to produce summaries solely of the form: "X talks with Y about the Z". The model uses this frame, filling in the X, Y and Z slots with relevant keywords and entities to achieve a small but positive coverage. This form of frame-filling is undesirable, as the model often produces inaccurate information to fit the entities to the pattern.

We implement a guard rail to penalize the model when frame-filling patterns are observed. During training, we keep track of the last 100 summaries produced by the model. We then aggregate the frequency of words for each word position in the 100 summaries. If any word appears more than 50% of the time at a specific word position, we raise the "No-frame-filling" penalty. In the example given above, the word "talks" appeared in the second word position in more than 50% of the summaries, as well as the word "about" in the fifth position.

These rule-based training guard rails are simple and effective. In our finalized trained models, very few summaries exhibit penalized behavior: 2% for no-repetition, 5% for finish-your-sentence, and 2.5% for no-frame-filling.

Figure 2.5 presents the plots of key variables we obtain during the training of the length 10 Summary Loop model. The training occurred over 10 days using a single Titan X GPU. During a first phase which occurs in the first 2 days of training, the model learns to copy content from the news article, which helps it achieve high Fluency and Coverage. In a second phase starting around the second day, the Summarizer learns to gain Coverage which maintaining Fluency mostly constant, which makes the overall Summary Score rise. The Summarizer model quickly learns to use its word budget, and after 10 days of training, the model uses an average of 9.7 words in its summaries.

(a) Fluency score

(b) Coverage score

(c) Summary Score

(d) #Words in summary
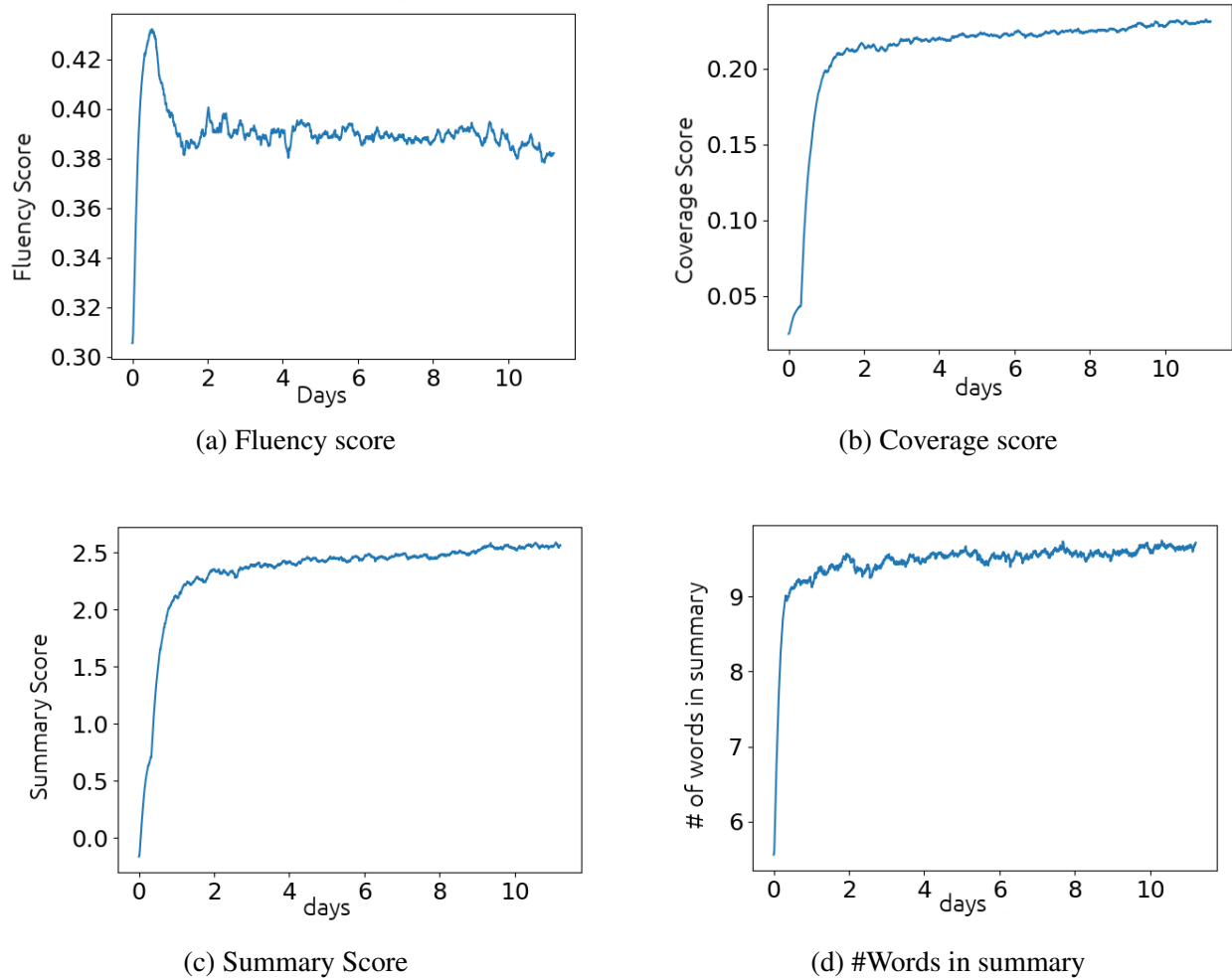
Figure 2.5: Plots of key variables during the training of the length 10 Summary Loop: (a) is a plot of the average Fluency Score, (b) is a plot of the average normalized Coverage Score, (c) is a plot of the average Summary Score (taking guard-rails into account), and (d) is a plot of the average number of words in summaries produced.

Supervised Methods	R-1	R-2	R-L
𝕏 Lead-3 baseline	**32.0**	**21.1**	**29.6**
PG + Coverage	27.5	13.3	23.5
Unsupervised Methods	**R-1**	**R-2**	**R-L**
𝕏 TextRank	24.5	**10.1**	20.1
Summary Loop 24	**27.0**	9.6	**26.4**

Table 2.4: ROUGE Results on the released test set of Newsroom. 𝕏 indicate extractive methods. Summary Loop outperforms other unsupervised method, is competitive with supervised Pointer-Generator.

2.5 Results

We present results for Summary Loop models trained in the news domain under three different length constraints: 10, 24, and 46 words, matching the distributions of the Headline, Newsroom [53] and CNN/DM [104] datasets. We compare our summaries using the standard ROUGE metric, and by analyzing summaries for the errors made, the technique used and the level of abstraction. Finally, we show the Summary Loop can be complemented with supervision, reducing the amount of data needed to achieve comparable ROUGE results.

News ROUGE Scores

Table 2.3 and Table 2.4 present ROUGE results on the CNN/DM and Newsroom datasets respectively. In both cases, Summary Loop outperforms other unsupervised methods, and is competitive with supervised methods despite not being exposed to any example summaries. On CNN/DM, Summary Loop performs in between the Pointer Generator and Bottom Up architecture in terms of ROUGE-1. On the Newsroom, Summary Loop is within 0.6 ROUGE-1 points of the Pointer-Generator with Coverage and surpasses it by 2 ROUGE-L points.

Recent breakthroughs in pretrained Transformer models have shown that using larger models in Summarization can lead to large improvements. For instance, a "large" version of the PEGASUS model [173] outperforms the "base" version by 2.3 ROUGE-1 points. Because Summary Loop experiments were performed using "base" models, we expect that using larger Transformer models could lead to similar gains.

Table 2.3 confirms that human-written summaries obtain amongst the highest Fluency and Coverage scores. Human-written summaries are only outperformed by Summary Loop summaries, and the Lede-3 baseline. However, the Summary Loop summaries are obtained by directly optimizing for Fluency and Coverage, and Lede-3 baseline summaries achieve their higher Coverage at the expense of being much longer (i.e. 84 words on average compared to 58 in human-written summaries).

Error Made	PGC	BU	SL
Inaccurate (%)	**11**	31	24
Ungrammatical (%)	**7**	15	18
Technique Used **(Success/Total)**	**PGC** **(S/T)**	**BU** **(S/T)**	**SL** **(S/T)**
Sent. Compression	86 / 110	96 / 177	118 / 194
Sent. Merging	13 / 27	29 / 65	71 / 121
Novel Sentence	0 / 1	4 / 18	33 / 70
Entity Manipulation	7 / 10	15 / 27	27 / 40
Total Technique	106 / 148	144 / 287	249 / 425

Table 2.5: Error and Technique analysis on 200 randomly selected summaries on the CNN/DM test-set for the Point-Gen with Cov. (PGC), Bottom-Up (BU) and unsupervised Summary Loop (SL). For each summarization technique, we report two numbers: the number of successful occurrences in summaries with no error, and the total number of occurrences in the 200 summaries.

Technique and Error Analysis

We perform a manual analysis of 200 randomly-selected summaries on the test set of CNN/DM from the Pointer-Generator with Coverage (PGC), Bottom-Up (BU) and the unsupervised Summary Loop (SL). We annotated each summary with two types of errors: Inaccurate (information in summary contradicts document), Ungrammatical (one sentence or more is not properly constructed), and four summarization techniques: Sentence Compression (summary sentence is a document sentence with words removed), Sentence Merging (2 or more document sentences are merged into a summary sentence), Novel Sentence (original sentence in the summary), and Entity Manipulation (a named entity is modified or simplified, e.g. changing a full name to a last name). We present Summary Loop examples illustrating each error and technique in Figures 2.7 – 2.12.

The analysis was performed by the first author of the paper, labeling article/summary pairs without knowledge of model origin. A summary can manifest any number of summarization Techniques, or none. Labeling is binary: if a summary exhibits more than one or instances of a Technique, it receives a 1, otherwise it receives a 0. Results of the analysis are summarized in Table 2.5.

SL uses significantly more summarization techniques (425) than PGC (148) and BU (287) summaries. Beyond raw counts, SL is more successful at applying summarization techniques (59% success) than BU (50% success), but less successful than PGC (72%). Note however that PGC takes little risk: 19% of the summaries go beyond sentence compression, and 39% are extractive, using none of the summarization techniques.

Span	Gold	Summary Loop	Bottom Up	PG + Cov
Novel	9.8%	0.6%	1.3%	1.0%
Length 1	23.6%	6.0%	2.9%	1.4%
Length 2	20.8%	11.4%	4.5%	1.1%
Length 3-5	24.7%	26.4%	13.0%	3.6%
Length 6-10	12.0%	29.7%	21.1%	9.3%
Length 11+	9.1%	25.9%	57.2%	83.6%
Avg. Length	4.2	7.8	14.8	25.2

Figure 2.6: Histogram and average copied span lengths for abstractive summaries. A summary is composed of novel words and word spans of various lengths copied from the document. Summary Loop summaries copy shorter spans than prior automatic systems, but do not reach abstraction levels of human-written summaries.

Level of Abstraction

All methods generating summaries one word at a time have potential for abstraction. In Figure 2.6 we analyze human and system written summaries for abstraction level. We measure a summary's level of abstraction by looking at the length of spans copied from the document. Summary Loop is the most abstractive automated method, although less so than human written summaries. SL cuts nearly in half the length of copied spans compared to other automated methods.

Supervision is not the enemy

If summaries are available, we show that they can complement the unsupervised Summary Loop. We run supervised experiments on CNN/DM using a generative Transformer architecture and varying the initialization. We compare initializing with (1) random weights, (2) the original GPT2 weights, and (3) the Summary Loop weights of target length 45. We train each model with teacher forcing, comparing using the entire CNN/DM training set to just 10% of it. The results are summarized in Table 2.6.

First, initializing with the Summary Loop leads to higher ROUGE score both in the 10% and full dataset setting. As expected, results improve when using the entirety of the data, and the Summary Loop initialized model trained with the entirety of CNN/DM obtains a ROUGE-1 F1-score of 41.0, within the confidence interval of the supervised Bottom Up [47] architecture. This is a strong result as the Transformer we use is a generic language model, and is not specialized for summarization.

Second, initializing with Summary Loop and training with 10% of CNN/DM yields comparable ROUGE scores to initializing with GPT2 and using the entire CNN/DM, showing that Summary Loop can be useful when fewer summaries are available.

Initialization Method	R-1	R-2	R-L	Test Loss
28k samples from CNN/DM (10%)				
Random Initialization	7.0	0.9	8.8	6.05
GPT2	37.1	15.9	31.9	2.21
Summary Loop S10	**38.7**	**16.2**	**35.1**	2.07
All of CNN/DN (100%)				
Random Weights	20.4	4.1	19.1	4.22
GPT2	38.4	17.2	35.0	2.02
Summary Loop S100	**41.0**	**18.1**	**37.3**	1.89

Table 2.6: ROUGE Results on the CNN/DM test-set for supervised generative Transformers. Initializing with the unsupervised Summary Loop outperforms random and GPT2 initializations.

2.6   Example Annotated Summaries

Figures 2.7, 2.8, 2.9, 2.10, 2.11, and 2.12 show example documents and the generated Summary Loop summary from the error and technique analysis of Section 2.5. Each summary manifests a summarization technique or error observed.

2.7   Discussion

Customizing summaries. In Figure 2.1, we illustrate the effect of the length constraint by summarizing the same document under three different length constraints. Each model adapts to its word budget. However, length is only one way to customize summaries. One might want to summarize based on point of view, chronology, theme, etc.

   Fluency vs. Grammaticality. By choosing to represent the validity of summaries with a Language model, we encourage fluent summaries (i.e., with likely sequences of words) but not necessarily grammatical ones. Extending the scoring to include grammaticality, either by using a parsing model, or leveraging the Corpus of Linguistic Acceptability [165] could prove useful.

   Summarization in the wild. Because our method is unsupervised, it can be applied to new domains and languages. In this work, we benefited from pretrained BERT and GPT2 models in English, which do not yet exist publicly for other languages. Once they become available in other languages, the Summary Loop can be ported over.

   Abstraction dangers. Recent work around measuring factuality in generated text, using Natural Language Inference [57] or rule-based fact extraction [178] becomes increasingly important with summaries that are more abstractive. This work can be naturally included into the Summary Loop, with a fact-checker model generating an accuracy score.

Sentence Compression Example

Document: He has long struggled to convince voters that he is a suitable choice for prime minister. Now Ed Miliband has hired a leadership coaching firm that helps people overcome anxiety and find their "inner voice". **The consultants drafted in by the Labour leader claim to work with politicians** to build "leadership skills" using "neuroscience" and "business psychology". Ed Miliband, pictured, has hired a US guru who can help him convince himself that he can be Prime Minister. [...]

Summary: Ed Miliband has hired a US guru who can help politicians on their leadership skills using neuroscience. Mr Miliband has hired the firm that can help politicians to build their leadership skills. **The consultants drafted in by the Labour leader claim to work with politicians.**

Figure 2.7: Summary Loop summary from the Error and Technique analysis (Section 2.5) illustrating the **Sentence Compression** technique. The blue boldface highlight is an example of sentence compression.

2.8   Conclusion

In this work we present a new approach to unsupervised abstractive summarization based on maximizing a combination of coverage and fluency for a given length constraint. When tested on common news summarization datasets, our method significantly outperforms previous unsupervised methods, and gets within the range of competitive supervised methods. Our models attain levels of abstraction closer to human-written summaries, although with more abstraction, more potential for factual inaccuracies arise.

Sentence Merging Example

Document: A single mom and her three kids who **"lost everything but their lives" in the East Village apartment explosion last week** are getting an incredible outpouring of support from their fellow New Yorkers. [...] Dr **McLean**, a 58-year-old child psychiatrist in the South Bronx, **says she and daughter Rose, 8, and twins James and Annabelle**, 5, had nothing more than the clothes on their backs after the disaster. *Diane McLean*, 58, *and her three children* lost "everything but their lives" when fire destroyed their apartment last week. Rose, 8, (left) and twins James and Annabelle, 5, lost everything except *the clothes on their backs* in the fire that destroyed their apartment building. [..] A GoFundMe campaign has raised nearly $ 90,000. [...]

Summary: **Diane McLean says she and daughter Rose, 8, and twins James and Annabelle, lost everything but their lives at East Village apartment explosion last week.** *Diane McLean and her three kids had the clothes on their backs.* A GoFundMe campaign has raised nearly $ 90,000.

Figure 2.8: Summary Loop summary from the Error and Technique analysis (Section 2.5) illustrating the **Sentence Merging** technique. The bold blue and italicized red selections are two examples of sentence merging. In the blue example "Dr McLean" is replaced by "Diane McLean" in the summary, an example of entity manipulation.

Novel Sentence Example

Document: For most of us, the dream of a holiday **home** is one that will probably never be realised. But for the lucky minority with a few extra million in the bank, its seems the world is quite literally your oyster when looking for property around the world. From a Lake Garda mansion with a pool overlooking the water to an Italian villa that looks like a castle and an Antigua retreat with Giorgio Armani as a neighbour, these are some of the most spectacular holiday homes on the market at the moment. **On the Lombardy side of Lake Garda**, this Lionard property is a luxurious villa with one serious waterfront view. Lake Garda. On the Lombardy side of Lake Garda, in northern Italy, lies a **luxury villa with a view** - just several miles north of Brescia. And for € 18 million (about £13 million or $20 million) it can all be yours. Not only is there a large swimming pool looking out on the water, but also a large deck with plenty of space for sun beds, gazebos and al fresco dining spots, overlooking a 4000 square metre garden. Inside, the house is just as breathtaking. For about 18 million Euros (or $ 13 million), the modern home, complete with pool, gazebo, and al fresco dining options, can be yours. [...]

Summary: **The Lake Garda home is a luxury villa with a view on the Lombardy side of Lake Garda.** This villa with gazebo and al fresco dining options. Inside, the house is just as breathtaking. For about 18 million Euros.

Figure 2.9: Summary Loop summary from the Error and Technique analysis (Section 2.5) illustrating the **Novel Sentence** technique. The first sentence of the summary uses pieces from the original document (in boldface blue) to form a sentence with an alternative but correct meaning.

Entity Manipulation Example

Document: Sipping a glass of glorious red wine which has been carefully aged in a hand-crafted oak barrel is my idea of heaven. [...] A $ 5 bottle has suddenly become $ 12 because the wine has lingered in an oak barrel before bottling. So when I read this week about a new gadget that **claims to be able to "oak age" wine in hours rather than years**, my curiosity was seriously roused. The Oak Bottle promises to impart an authentic aged flavour – a process that can take up to two years – in just a day or two. Who wouldn't drink to that ? Scroll down for video. TV wine expert Oz Clarke puts to the test this oak bottle that claims to "oak age" wine in hours rather than years. The product, which retails at $ 50, is the brainchild of 30-year-old entrepreneur **Joel Paglione**. [...]

Summary: Joel Paglione said the Oak Bottle promises to be able to oak age wine in hours rather than years. The Oak Bottle promises an authentic aged flavour that can take up to two years. A bottle has been made in an oak barrel.

Figure 2.10: Summary Loop summary from the Error and Technique analysis (Section 2.5) illustrating the **Entity Manipulation** technique. The entity Joel Paglione (in boldface blue) is correctly inserted to represent the company.

Inaccurate Example

Document: The traditional cookie cutter wedding no longer exists - new reports suggest Brits are ditching tradition in favour of alternative practices when it comes to getting hitched. Two of the biggest changes are the fact **that religious services have fallen out of favour** and that brides are opting for bold colour schemes for their big day. A new study, which has tracked the decisions of brides and grooms over the past five years interviewed 1,893 newlyweds and compared them to answers they have collated since 2010. Scroll down for video. [...]

Summary: **The new study showed that British couples are opting for religious ceremonies** when it comes to their big day with services falling from 40 per cent of the past five years. The study showed that couples are opting to holiday in the UK.

Figure 2.11: Summary Loop summary from the Error and Technique analysis (Section 2.5) illustrating the **Inaccurate** error. The summary inaccurately claims religious ceremonies are increasing, when the document says they are in decline. Key phrases are highlighted in boldface blue.

Ungrammatical Example

Document: Despite his daughter remaining in a medically induced coma since she was found unresponsive in a bathtub at her Atlanta home in January, singer Bobby Brown told an audience on Saturday night that she is "awake.". Bobby was performing at the Verizon Theatre in Dallas when he told the stunned audience that "Bobbi is awake. She's watching me." The singer didn't elaborate on if his daughter had regained consciousness or if he was talking instead about her spirit. After the 46-year-old's comment, his sister Tina posted on Facebook," [...] Whitney Houston's family insists the 22-year-old is not awake and is the same condition she was when she entered the facility. "**She's in the exact same condition she was in when she went into the facility.**" a source told the site [...]

Summary: Bobby Brown was performing at the Verizon Theatre in Dallas when Bobbi was awake. He said that Tina posted on Facebook that her daughter was awake. She was the singer. She was going to be awake. **She is the same condition.**

Figure 2.12: Summary Loop summary from the Error and Technique analysis (Section 2.5) illustrating the **Ungrammatical** error. The last short summary sentence (in boldface blue) is not properly constructed, based on an unsuccessful attempt to compress a sentence in the document (also in boldface blue).

Chapter 3

Keep it Simple: Unsupervised Simplification

3.1 Introduction

The main objective of text simplification is to make a complex text accessible to a wide audience by increasing its readability. In contrast with text summarization – in which key content is selected to remain in the summary and other content is elided – in text simplification, ideally all relevant content is preserved.

> **Original:** NASA's Curiosity rover just celebrated a major milestone — 3,000 days on the surface of Mars. To mark the occasion, the space agency has released a stunning new panorama of the red planet, captured by the rover.
>
> **Model Full:** NASA's Curiosity rover has now passed 3,000 days of travel on the surface of Mars. To mark the milestone, the space agency released a huge panorama of Mars, as seen by the rover.
>
> **Model No Fluency:** NASA's Curiosity rover. celebrated. A major milestone — 3,000 days on. The of.. To mark. The space agency has. a stunning new panorama.. red planet. captured by. The rover. However
>
> **Model No Salience:** NASA's Curiosity rover just celebrated a major milestone. The space agency has released a stunning new panoramic of the red planet, captured by the team. It was by the rover's panoramic camera.
>
> **Model No Simplicity:** NASA's Curiosity rover has celebrated a major milestone, 3,000 days on the ground of Mars. To mark the occasion, the space agency has unveiled a stunning new panoramic view of the red planet, captured by the rover.

Figure 3.1: Motivating example for the KiS method, based on a CBS article [89]. We optimize a three-component reward: fluency, salience and simplicity. We show model outputs when trained with all three components, and with a missing component.

We propose that text simplification algorithms need to balance three properties: (1) **fluency**: the simplified text should use well-formed English sentences, (2) **salience**: the simplified text should relay the same information as the original, and (3) **simplicity**: the simplified text should be syntactically and lexically simpler than the original.

Figure 3.1 provides intuition for the necessity of each of the three properties. It shows the original text and the output of the full proposed model compared to three reduced versions:

Without Fluency, the generator has no incentive to generate full sentences, and learns it can boost the simplicity score by generating short phrases with excessive punctuation.

Without Salience, the generator does not gain by covering facts in the original text, and can improve the simplicity score by learning to remove facts (e.g., not mentioning planet Mars by name).

Without Simplicity, the generator is not guided to favor syntactically and lexically simpler re-writes. In Figure 3.1, *Model No Simplicity* is in fact more complex than the original according to readability measures.

As we show in the related work section (Section 3.2), there are no high-quality, large datasets publicly released for text simplification. In this work, we build on recent progress of reinforcement learning (RL)-based training of text generators: we formulate a reference-free reward for text simplification and directly optimize it, circumventing the need for aligned data.

Our main contribution is the Keep it Simple (KiS) procedure, a novel unsupervised method for text simplification. Applied to the English news domain, KiS outperforms several supervised models on common simplification metrics such as SARI [171] and the Flesch-Kincaid Grade Level [74].

A second contribution is a new algorithm for RL-based training of text generators, k-SCST, which is an extension of Self-Critical Sequence Training [133]. For each input, we generate k sampled outputs (vs. 2 in SCST), and use the mean population reward as a baseline. We show in Section 3.4 that in our domain, k-SCST outperforms models trained with SCST.

A third contribution is a novel evaluation method for text simplification. Based on the assumption that simplified text should enable faster reading with better understanding, we propose a realistic Text Comprehension task. We show that people reading texts simplified by KiS are able to complete comprehension tasks faster than comparison texts.

Another departure from previous work is that we work with *paragraphs* as units of text. Most work in text simplification is done at the sentence level, despite work such as Zhong et al. [183] showing that common simplification phenomena occur at the level of the paragraph, (e.g., the deletion, insertion or re-ordering of full sentences). Specifically, we train our models to simplify full paragraphs, and evaluate our models in a human evaluation on short *documents* (i.e., 3-4 paragraphs).

Through rigorous empirical evaluation, we demonstrate the strong performance of our approach; automated results show that this unsupervised approach is able to outperform strong supervised models by 4 SARI points or more. We publicly released the code and model checkpoints[1].

3.2   Related Work

Simplification Datasets. Early datasets were first based on Simple Wikipedia[2]: WikiSmall [184], later expanded into WikiLarge [177]. Xu, Callison-Burch, and Napoles [170] show there are quality concerns with Simple Wikipedia datasets, and propose Newsela[3] as a replacement. Newsela is a project led by educators re-writing news articles targeting different school grade levels. We view

---

[1]`https://github.com/tingofurro/keep_it_simple`
[2]`https://simple.wikipedia.org/`
[3]`https://newsela.com/`

Newsela as the gold-standard for our work, and use the public Newsela release of 1,911 groups of articles to design and evaluate our work. Using a coarse paragraph alignment algorithm, we extract 40,000 paired simple/complex paragraphs targeting a separation of 4 grade levels. We call this dataset the *paired Newsela dataset*, which we use for analysis and baseline training.

Seq2Seq for Simplification. Text simplification is most commonly framed as a sequence-to-sequence (seq2seq) task, leveraging model architectures of other seq2seq tasks, such as natural machine translation [184, 169]. Martin et al. [98] introduce ACCESS, a finetuned Transformer model that achieves state-of-the-art performance on WikiLarge. ACCESS can customize simplifications on parameters such as compression rate and paraphrase amount. We directly compare our approach to ACCESS.

Data availability remains one of the main limitations to seq2seq-based text simplification. We side-step this issue entirely by working with unsupervised data, only requiring a small dataset with coarse-level alignments for calibration.

Lexical Simplification focuses on the substitution of single words or phrases with simpler equivalents, with diverse approaches using lexical databases such as WordNet [156], to using contextualized word vectors [127]. These methods tend to be limited, as they do not consider syntactic complexity, and have no direct way of modeling deletions and insertions. We incorporate a lexical score (L_{Score}) as one of the rewards in our simplicity component.

Text-edit for Simplification. Recent work [31, 148] has modeled text simplification as a *text-edit* task, learning sequences of word-edits that transform the input into the output. Text editing offers explainability, at the cost of added model complexity. We find that without explicitly representing edits, the KiS model easily learns to copy (using attention heads) and deviate from the original text. Outputs can be post-processed into edits, if desired.

Unsupervised Simplification has mostly been limited to lexical simplification. Recently Surya et al. [151] (Unsup NTS) proposed a system that can perform both lexical and syntactic simplification, with a joint encoder, and two decoders (simple and complex). We directly compare our unsupervised approach to Unsup NTS.

RL for Simplification. Prior work [177, 56] used Reinforcement Learning (RL)-based simplification. However, in both cases, components of the reward or training procedure involved reference simplifications, requiring an aligned dataset. By designing a reference-free reward, we are able to train our model with RL without supervision.

Evaluation of Simplification. This usually falls into two categories: automatic offline evaluation, and human evaluation. Automatic evaluations usually involve using n-gram overlap calculations such as BLEU [116] and SARI [171]). SARI was shown to correlate better with human judgements of simplicity than BLEU, and it has since become a standard [177, 151, 98]. In our experiments, we report both SARI and BLEU.

Human evaluation is typically done in an *intrinsic* way – e.g., by directly rating factors like fluency, simplicity and relevance of model outputs [151, 169]. In this work, we propose an extrinsic, task-based protocol. In our comprehension study, we directly measure how much simplified texts can help a human reader answer questions more efficiently. The closest to our evaluation design is that of Angrosh, Nomoto, and Siddharthan [3] with the important difference that we require
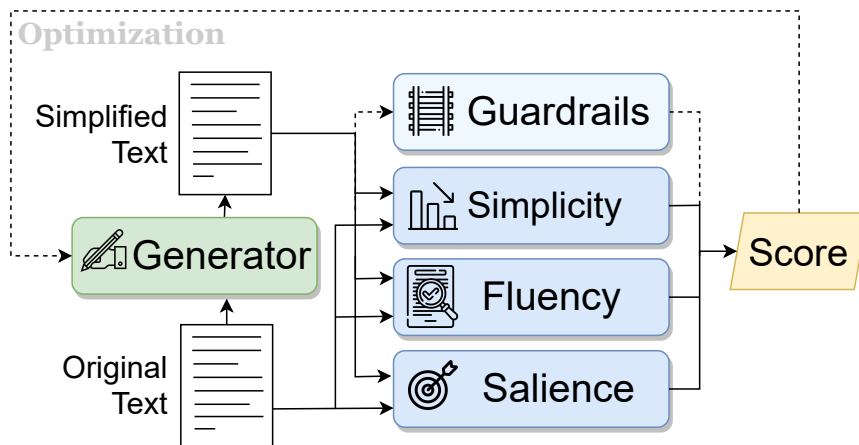
Figure 3.2: Keep it Simple is an unsupervised training procedure for text simplification. The text generator (GPT-2) produces candidate simplifications, scored according to *fluency*, *simplicity*, *salience*. *Guardrails* enforce the model does not learn high-scoring shortcuts.

participants to resubmit after erroneous answers. In pilot studies, we found this step to be crucial for high-quality responses.

3.3 KiS Components

In KiS, we approach unsupervised simplification as a (non-differentiable) reward maximization problem. As shown in Figure 3.2, there are four components to the reward: simplicity, fluency, salience and guardrails which are jointly optimized. This is essential to avoid trivial solutions that only consider subsets. We therefore use the product of all components as the total reward, because the product is sensitive to the sharp decrease of a single component. For example, the triggering of a single guardrail leads to the zeroing of the total reward. Each component is normalized to the $[0, 1]$ range.

Simplicity

The simplicity score should establish whether the generator's output uses simpler language than the original text. We follow prior work [45] and organize our score into a syntactic score S_{Score}, and a lexical score L_{Score}. Syntactic simplification focuses on reducing the complexity of a sentence, for example by reducing the number of words in a clause, or reducing distant dependencies. In lexical simplification, the objective is to replace complex phrases with simpler synonyms. To produce a single simplicity score, we take the product of S_{Score} and L_{Score} (both in $[0, 1]$).

```python
def S_Score(original,simple):
    Fstart = fkgl(original)
    tgt = target_delta(Fstart)
    Fend   = fkgl(simple)
    D = Fend-Fstart
    return clip(1-((D-tgt)/tgt),0,1)
def target_delta(Fstart):
    # Line-fitted from analysis
    if Fstart < 4.0:
        return 0.1
    if Fstart < 12:
        return 0.5*Fstart-1.9
    return 0.8*Fstart-5.6
```

Figure 3.3: S_{Score} algorithm. `fkgl` computes the Flesch-Kincaid grade level.
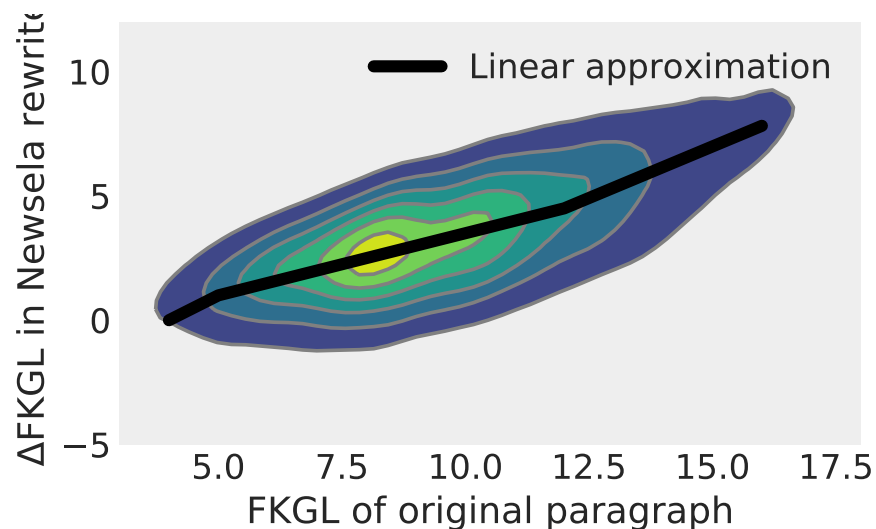
Figure 3.4: Analysis (Kernel Density Estimate plot) of change in Flesch-Kincaid Grade Level in the paired Newsela dataset. Most simple paragraphs have lower FKGL than the original paragraphs (positive $\Delta FKGL$). When the original paragraph's FKGL is higher (x-axis), the change in FKGL tends to be larger (y-axis). We fit a linear approximation, which we use to compute the $Sscore$.

Syntactic Simplicity: S_{Score}

We measure syntactic complexity via the Flesch-Kincaid grade level (FKGL) as it is easy to compute and maps to a grade-level which also corresponds to the scale used by Newsela. Other readability metrics such as Dale-Chall formula [28], or the Gunning-Fog index [55] could be used, and future work could examine the effect of choosing one readability metric over the other. Another viable

option is the Lexile score [146], however, because its implementation is not publicly released, we cannot use it during training and we report it only for evaluation (done manually on the Lexile Hub[4]).

Figure 3.3 shows the S_{Score} algorithm. We compute the original paragraph's FKGL (`FStart`), used to compute a target FKGL (`tgt`). The score is a linear ramp measuring how close the achieved FKGL (`Fend`) is to the target, clipped to $[0, 1]$.

In the initial design, the target drop was a constant: 4 grade levels, independent of `FStart`. However, analysis on the paired Newsela corpus revealed that the target FKGL should depend on the initial FKGL. This makes sense intuitively: an already syntactically simple paragraph should not require further simplification, while more complex paragraphs require more simplification. Figure 3.4 shows the positive correlation between the original paragraph's FKGL and the drop of FKGL in the simplified text. We fit a piece-wise linear function to calculate the target FKGL drop from the initial paragraph.

Lexical Simplicity: L_{Score}

Lexical simplicity focuses on whether words in the input paragraph (W_1) are more complex than ones in the output paragraph (W_2). We rely on the observation that word frequency and difficulty are correlated [13], and use word frequency in a large corpus of text [16] to determine simplicity.

Because word frequency follows a Zipf power law, we use Speer et al. [147]'s log normalization, adjusting the frequency on a $[0, 8]$ range, with words at 0 being non-existent in the corpus, and 8 for most common words. As an example, the word *vigorous* has a frequency of 3.54, while its more common synonym *strong* obtains 5.23.

We compute the average Zipf frequency of the set of inserted words ($Z(W_2 - W_1)$), and the set of deleted words ($Z(W_1 - W_2)$). The difference

$$\Delta Z(W_1, W_2) = Z(W_2 - W_1) - Z(W_1 - W_2) \tag{3.1}$$

should be positive. Analysis of the *paired Newsela corpus* reveals that 91% of pairs have a positive $\Delta Z(W_1, W_2)$, with a median value of 0.4. We use this median as the target Zipf shift in the L_{Score}, and use a ramp shape similar to the S_{Score}, clipped between 0 and 1 (denoted as $[\cdot]^+$):

$$L_{Score}(W_1, W_2) = \left[1 - \frac{|\Delta Z(W_1, W_2) - 0.4|}{0.4} \right]^+ \tag{3.2}$$

Fluency

We use two sub-components for the fluency component: a pre-trained language-model, and a discriminator trained dynamically with the generator.

---

[4]https://hub.lexile.com

Language-Model Fluency

Language models assign a probability to a sequence of words. This probability is often used to measure fluency of generated text [68, 136]. The KiS fluency score is based on a language model in a way similar to Chapter 2. The language model is used to obtain a likelihood of the original paragraph ($LM(p)$) and of the generated output $LM(q)$. We use average log-likelihood, for numerical stability. The language model fluency score is then:

$$LM_{Score}(p, q) = \left[1 - \frac{LM(p) - LM(q)}{\lambda} \right]^+ \tag{3.3}$$

λ is a tunable hyper-parameter. If the $LM(q)$ is lower than $LM(p)$ by λ or more, $LM_{Score}(p, q) = 0$. If $LM(q)$ is above or equal to $LM(p)$, then $LM_{Score}(p, q) = 1$, and otherwise, it is a linear interpolation.

We set $\lambda = 1.3$ as it is the value for which the *paired Newsela dataset* achieves an average LM_{Score} of 0.9.

Discriminator Fluency

The LM_{Score} is static and deterministic, which can be limiting, as the generator can learn during training how to adapt and exploit flaws in the language-model (e.g., learning to alter capitalization).

Inspired from the Generative Adversarial Network (GAN) framework [50], we create a dynamic discriminator, trained in conjunction with the generator, dynamically adapting the fluency score during training.

Specifically, we use a RoBERTa model [93] as the basis for the discriminator, a classifier with two labels: 1 for authentic paragraphs, and 0 for generator outputs.

As the generator produces outputs, they are assigned a label of 0 and added to a *training buffer*, while the original paragraphs are assigned a label of 1 and added to the training buffer as well.

Once the training buffer reaches a size of 2,000 samples, the discriminator is trained, using 90% of the training buffer. We train the discriminator for 5 epochs (details of training are in Section 3.5). At the end of each epoch, we checkpoint the discriminator model. We compare the 5 checkpoints in terms of F-1 performance on the remaining 10% of the training buffer, and keep the best checkpoint as the new discriminator.

The discriminator's probability that a paragraph (q) is authentic is the discriminator score:

$$D_{Score}(q) = p_{disc}(Y = 1 | X = q) \tag{3.4}$$

As with GANs, there is an equilibrium between the generator attempting to maximize the probability of generating real outputs ("fooling" the discriminator), and the discriminator succeeding at distinguishing generated and authentic texts.

Salience

For the salience component, we use the *coverage model* introduced in Chapter 2 for the domain of text summarization, and adapt it to the simplification domain.

The coverage model is a Transformer-based model trained to look at generated text and answer fill-in-the-blank questions about the original text. The score is based on model accuracy at filling in the blanks: the more is filled in, the more relevant the generated content is, and the higher the score.

A key element of the coverage model is its masking procedure, which decides which words to mask. In the summary loop, a limited number of extracted keywords (up to 15 words) are masked. By contrast, for simplification, we mask all non-stop words, amounting to a masking rate of about 40%.

This change reflects a difference in expectation between summarization and simplification: in summarization, only key components are expected to be recovered from a summary, whereas in simplification most of the original paragraph should be recoverable. Coverage ranges in $[0, 1]$, and reference simplifications in the *paired Newsela corpus* obtain an average score of 0.76, confirming that manual simplification can achieve high coverage.

Guardrails

We use *guardrails* as simple pattern-based scores to avoid common pathological generation problems that we observed. Unlike the main components, guardrails are binary, giving a score of 1 (pass) unless they trigger (score of 0). We use two guardrails: brevity and inaccuracy.

Brevity guardrail

The brevity guardrail ensures the length of generated paragraph (L_2) falls in a range around the original paragraph's length (L_1). We compute a compression ratio: $C = L_2/L_1$. If $C_{min} \leq C \leq C_{max}$, the guardrail passes, otherwise it triggers.

We set $[C_{min}, C_{max}] = [0.6, 1.5]$, because these values ensure the guardrail is not triggered on 98% of the paired Newsela dataset; this can be adapted depending on the application.

Inaccuracy guardrail

Modern text generation models are known to *hallucinate* facts [63], which has led the community to create models to detect and correct hallucinations [18, 179, 163].

We propose a light-weight inaccuracy detector as a guardrail. We use a Named Entity Recognition (NER) model [62] to extract entities present in the original paragraph (E_1) and the model's output (E_2). We trigger the guardrail if an entity present in E_2 is not in E_1.

Even though human writers can successfully introduce new entities without creating inaccuracies (e.g., replacing the city *La Paz* with the country *Bolivia*), we find that text generators predominantly introduce inaccuracies with novel entities. This simple heuristic can eventually be replaced once inaccuracy detection technology matures.
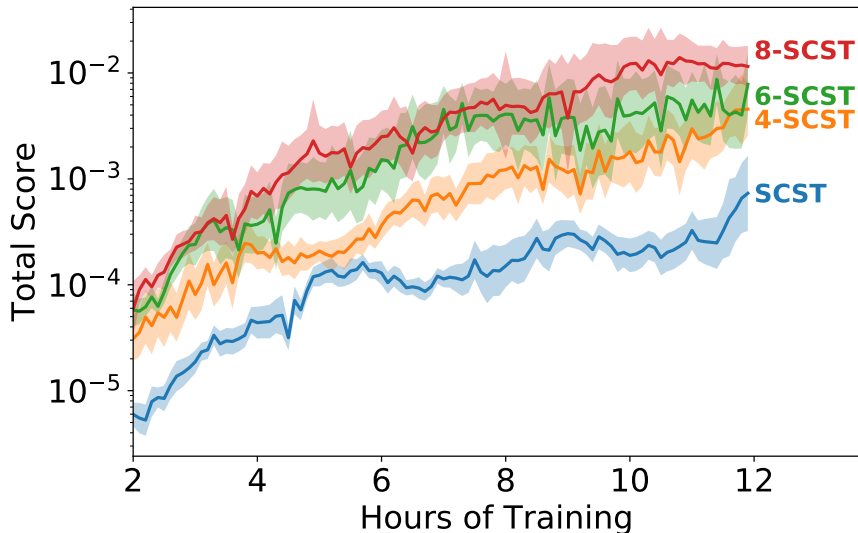
Figure 3.5: Training KiS models comparing SCST with k-SCST. We try 4, 6 and 8 as values for k. Increasing k improves performance and stability.

3.4 KiS Training

Rennie et al. [133] introduced Self-Critical Sequence Training (SCST) as an effective algorithm for reward-based training of text generators, successfully applying it to image captioning. The efficacy of SCST was later confirmed on other text generation tasks such as question generation [176], and summarization [20, 84]. In SCST, a probabilistic model is used to generate two distinct candidates: C^S, a candidate constructed by sampling the word distribution at each step, and $\hat{C}$, by taking the argmax of the word distribution at each step. Each candidate is scored, obtaining rewards of R^S and $\hat{R}$, respectively, and the loss is:

$$L = (\hat{R} - R^S) \sum_{i=0}^{N} \log p(w_i^S | w_1^S ... w_{i-1}^S, P) \tag{3.5}$$

where $p(w_i^S|...)$ represents the probability of the i-th word conditioned on previously generated sampled sequence according to the model, P is the input paragraph, and N the number of words in the generated sequence. Intuitively, minimizing this loss increases the likelihood of the sampled sequence if $R^S > \hat{R}$, and decreases it otherwise, both increasing the expected total reward.

One limitation in SCST occurs when the two sequences achieve comparable rewards ($R^S \simeq \hat{R}$): the loss nears zero, and the model has little to learn, wasting a training sample. In our experiments with SCST, this can occur with 30% of samples.

We propose an extension of SCST, which we call k-SCST. We generate k sampled candidates ($k > 2$), compute the rewards of each candidate $R^{S1}, ..., R^{Sk}$, as well as the mean reward achieved by this sampled population: $\bar{R}^S = (R^{S1} + ... + R^{Sk})/k$, which we use as the baseline, instead of $\hat{R}$.

The loss L becomes:

$$L = \sum_{j=1}^{k} (\bar{R}^{S} - R^{Sj}) \sum_{i=0}^{N} \log p(w_i^{Sj}|w_1^{Sj}...w_{i-1}^{Sj}, P) \tag{3.6}$$

We use a GPT2-medium for the generator, initialized with the released pre-trained checkpoint. Experimental details such as data and optimizer used are provided in Section 3.5.

In Figure 3.5, we show results of a direct comparison of SCST ($k = 2$) with k-SCST varying k in $\{4, 6, 8\}$, while keeping other components of the training fixed. Because of the variance involved in RL training, we recorded six independent training runs for each setting (for a total of 24 runs), and plot the average reward across runs of a setting, as well as the standard error of the mean (SEM).

We observe that increasing k leads to higher average reward, and less variation in the reward. In our setting, k-SCST boosts performance and stabilizes training. We use $k = 8$ in all final models, as increasing k further is impractical due to GPU memory limitations.

We believe k-SCST's advantage stems from two factors: first, obtaining a better estimate of the distribution of rewards by sampling more outputs, second, by using the mean reward as the baseline, saving on computation of a separate baseline generation. We believe k-SCST can also improve learning in other text generation applications and plan to pursue this in future work.

3.5 Training Details

We detail the model architecture size, data, optimizer of the models we train in the paper. All models were trained using Pytorch and HuggingFace's Transformers library[5]. We use the Apex[6] library to enable half-precision training.

The KiS procedure was trained on a single GPU, either an Nvidia V-100 (16Gb memory) or a Quadro RTX 8000 (48 Gb memory). We ran a total of around 200 experiments, with an average run-time of one week.

Because the procedure is unsupervised, the model was trained using a large unreleased corpus of news articles, containing 7 million news articles in English.

KiS Model is initialized with a *GPT2-medium* model. We used the Adam optimizer, with a learning rate of 10^{-6}, a batch-size of 1, using k-SCST with $k = 8$.

Finetune Baseline is initialized with a *GPT2-medium* model. We train using using standard teacher forcing on the 40,000 samples in the *paired Newsela dataset*, reserving 2,000 samples for validation. We use the Adam optimizer, and use the validation set to choose a learning rate of 10^{-5}, and a batch-size of 8, and run for 3 epochs before seeing a plateau in the validation loss.

Discriminator Model is initialized with a **Roberta-base**, and retrained every time the training buffer reaches 2,000 samples. The discriminator is reset to the original *Roberta-base* each time the training buffer is full. We use a standard cross-entropy loss, the ADAM optimizer with a learning rate of 10^{-5} and a batch size of 8. Each time we retrain, we run for 5 epochs, and checkpoint one

---

[5]https://github.com/huggingface/transformers
[6]https://github.com/nvidia/apex

Model	SARI	BLEU	%FKGL	%Lexile	Comp.	Cov.
Newsela	-	-	87	**79**	.918	.754
Finetune Baseline	.470	**.719**	68	52	.903	**.894**
ACCESS Default	.666	.649	86	63	.958	.805
ACCESS 90	.674	.644	93	64	.921	.789
Unsup NTS	.677	.535	48	57	.753	.618
KiS Model	**.709**	.526	**100**	72	.852	.640

Table 3.1: Automatic results on Newsela test-set. *SARI* and *BLEU* are reference-based metrics. *%FKGL* and *%Lexile* are percentages of model outputs lowering the grade level. *Comp.* is the average compression ratio (# words), and *Cov.* the output's average coverage score.

model after each epoch. The checkpoint that achieves the highest performance on a validation set becomes the new discriminator for the next round.

3.6 Experiments

We present results experimentally validating the KiS procedure for text simplification. We give results based on automatic metrics, on a novel human comprehension task, and from an ablation study.

Models Compared

We compare the **KiS Model** to three strong supervised models, and an unsupervised approach.

ACCESS from [98], is a state-of-the-art Transformer model trained on WikiLarge (300,000 pairs of complex/simple sentences). This model uses default parameters ($NBChar$=0.95, $LevSim$=0.75).

ACCESS90 is identical to **ACCESS**, with different parameters ($NBChar$=0.90, $LevSim$=0.75), reducing target compression from 95% to 90%, matching the average compression rate in Newsela.

Finetune Baseline is a GPT2-medium model finetuned on the *paired Newsela dataset*. Large pre-trained models often perform competitively in low-resource environments, making this a strong point of comparison.

Unsup NTS from [151] is an unsupervised approach based on successively encoding and denoising text using a GRU architecture.

Training details for the KiS Model and Finetune Baseline are in Section 3.5.

Automatic Results

We put aside 500 samples from the *paired Newsela dataset* as a test set to compare models on automatic metrics. We compare models on SARI and BLEU, report the percentage when readability

measures see an improvement in readability: %FKGL, and %Lexile and compute the average compression rate (Comp.), and coverage (Cov.). Results are summarized in Table 3.1.

The KiS model achieves the highest SARI score by a margin of 0.04, even though it is an unsupervised approach.

Finetune Baseline achieves the highest BLEU and salience scores, but lowest SARI score. We interpret this as showing the model takes the least risk: high salience, with little simplification.

We observe that all models are able to increase readability in terms of FKGL and Lexile compared to original paragraphs. We note that for almost all models, the percentage is lower for the Lexile measure than for FKGL, showing that an improvement in Lexile score is more difficult to achieve than FKGL. The KiS model achieves an increase in Lexile readability 72% of the time, the closest figure to 79% of the Newsela human-written reference.

We note that the perfect performance of KiS on %FKGL could be explained by the fact that FKGL is a part of a component being optimized (S_{Score}), however Lexile was not.

In terms of compression, the KiS model compresses the second most, most likely hurting its coverage. Adjusting the Brevity guardrail could encourage the model to compress less. ACCESS90 has the compression rate closest to Newsela references, but this only leads to a modest improvement in SARI when compared to ACCESS.

Overall, the Newsela references achieve the best percentage of Lexile readability improvement, while outperforming the KiS model at coverage: there is still a gap between human-written simplifications and model-generated ones.

Human Comprehension Study

We propose a human comprehension study to evaluate the usefulness of simplification results. Simplified text should be easier to read than the original text, while retaining accuracy and understanding. We design a task to evaluate how well both manual and automated simplifications achieve this objective. The main idea is to show readers a text and ask them to answer multiple-choice questions, evaluating the texts based on time and retries needed to select the correct answer.

Study Design

Five different versions of each document were generated as stimuli: the original document, the Newsela reference, and versions from the three best-performing methods from the last section: KiS, Finetune Baseline, and ACCESS. We did not include Unsup NTS in our analysis, because of its low performance on %FKGL and %Lexile metrics. Associated with each document are five manually generated multiple-choice questions, each with one or more correct answers and one to four distractors. The original and the Newsela texts were checked manually by experimenters to ensure that all allow for questions to be answered correctly. Crowd-workers were shown four documents in succession, in a between-participants design. Order of document and stimuli type were randomized. Figure 3.6 shows two stimuli of a document (original and KiS) along with the comprehension questions. The entire set of five stimuli can be found in Figure 3.7.

After several rounds of pilot testing, we arrived at the following design choices:

ORIGINAL [Lexile Grade 11] Each summer, libraries in St. Louis, Missouri, host many types of free camps — yoga, chess and even a Harry Potter "Sorting Hat Camp." In 2020, camp dreams seemed far-fetched given the global coronavirus pandemic. That didn't stop St. Louis libraries, though.

Instead of canceling, they brought camp into kids' homes. So children who signed up for ukulele camp got a beginner's guidebook, instructional DVD and an actual ukulele in the mail. It was all free. In addition, camp sessions still occurred. Advisers met with kids using virtual formats.

Joe Monahan, manager of youth services for the St. Louis library system, says that of the 70 camps originally scheduled, 54 were held virtually.

Paula Langsam, a youth services manager at the soon-to-reopen Martin Luther King Junior Memorial Library in Washington, D.C., says, "In a way, our work has changed a lot. We didn't used to do videos a lot."

KIS MODEL [Lexile Grade 9] In the summer months, St. Louis has many free classes for kids, including yoga, chess and a Harry Potter "Sorting Hat Camp." In 2020, camp dreams again seemed far-fetched given the crisis. That didn't stop St. Louis libraries, though.

They brought camp in. So kids who signed up for ukulele camp got a beginner's guidebook, a lesson DVD and a real ukulele in the mailbox. It was all free. In addition, camp sessions continued. Advisers tried out a virtual format.

Joe Monahan, the manager of youth services for the St. Louis library system, says that of the 70 camps originally scheduled, 54 were held mostly.

Paula Langsam, a youth services manager at the Martin Luther King Junior library, says, "In a way, our work changed a lot. We didn't do videos a lot."

(?) *Who manages the St Louis library kids programs?*
Joe Monahan, Paula Langsam, St. Louis Camp Leaders

(?) *How many camps were scheduled, how many were run?*
54 and 70, **70 and 54**, 70 and 0, 54 and 0

(?) *Were any camps in St. Louis cancelled?*
Yes, No

(?) *How did the Ukulele camp meet?*
In the park, **Virtually**, Did not meet

(?) *What camps did the libraries host?*
Yoga, **Chess**, Pottery, **Ukulele**

Figure 3.6: Example Task (from a Washington Post article [72]) for the Comprehension Study. Shown are two of five stimuli: original document (left), and KiS model output (right). Participants read a text and answered comprehension questions (bottom). Average completion time was 160 seconds (original) and 136 seconds (KiS model output).

Document theme. We chose recent news articles involving complex themes (e.g., trajectory of iceberg) as the source of documents. For news articles, recency seems to engage participants, and technical terms increase the impact of simplification.

Section length. We chose document length of 3-4 paragraphs (or 200 words), and five comprehension questions. Document length should not be too W (makes some questions trivial), or too long (adds a retrieval component to the task).

Selection of questions. Questions were generated via a GPT2 question generation model finetuned on the NewsQA dataset [158]. We select questions answerable by both the original and Newsela references, attempting to have both factoid (answer is entity) and reasoning questions.

Re-submission until correct. When submitting answers, participants received feedback on which were incorrect, and were required to re-submit until all answers were correct. This aligns the objective of the participant (i.e., finishing the task rapidly), with the task's objective (i.e., measuring participant's efficiency at understanding). This also gives a way to discourage participants from "brute-forcing" the task, re-submitting many combinations until one works.

We note that some components of the study such as the choice of document themes and the selection of comprehension questions are elements that create variability in the results. We release the models used in the study, as well all generated texts that were evaluated to enable follow-up research and to aid reproducibility.

ORIGINAL [Lexile Grade 11] Each summer, libraries in St. Louis, Missouri, host many types of free camps — yoga, chess and even a Harry Potter "Sorting Hat Camp." In 2020, camp dreams seemed far-fetched given the global coronavirus pandemic. That didn't stop St. Louis libraries, though.

Instead of canceling, they brought camp into kids' homes. So children who signed up for ukulele camp got a beginner's guidebook, instructional DVD and an actual ukulele in the mail. It was all free. In addition, camp sessions still occurred. Advisers met with kids using virtual formats.

Joe Monahan, manager of youth services for the St. Louis library system, says that of the 70 camps originally scheduled, 54 were held virtually.

Paula Langsam, a youth services manager at the soon-to-reopen Martin Luther King Junior Memorial Library in Washington, D.C., says, "In a way, our work has changed a lot. We didn't used to do videos a lot."

NEWSELA [Lexile Grade 7] In St. Louis, Missouri, libraries hold many types of free camps in the summer. They have yoga, chess and even a Harry Potter camp. In 2020, camp ideas seemed unlikely due to the spread of COVID-19. That did not stop St. Louis libraries, though. They did not cancel their programs. They brought camp into kids' homes.

Children who signed up for ukulele camp got a beginner's book, instructional video and a ukulele in the mail. It was all free. In addition, camp sessions still went on. Camp leaders met with kids over video calls.

Joe Monahan runs the kids programs for the St. Louis libraries. He said 70 camps were scheduled at first. Some were dropped but 54 were still held online.

Paula Langsam works at the Martin Luther King Junior Memorial Library in Washington, D.C. It reopened on September 24. She said, "Our work has changed a lot. We didn't used to do videos a lot."

KIS MODEL [Lexile Grade 9] In the summer months, St. Louis has many free classes for kids, including yoga, chess and a Harry Potter "Sorting Hat Camp." In 2020, camp dreams again seemed far-fetched given the crisis. That didn't stop St. Louis libraries, though. They brought camp in. So kids who signed up for ukulele camp got a beginner's guidebook, a lesson DVD and a real ukulele in the mailbox. It was all free. In addition, camp sessions continued. Advisers tried out a virtual format.

Joe Monahan, the manager of youth services for the St. Louis library system, says that of the 70 camps originally scheduled, 54 were held mostly.

Paula Langsam, a youth services manager at the Martin Luther King Junior library, says, "In a way, our work changed a lot. We didn't do videos a lot."

FINETUNE BASELINE [Lexile Grade 9] Each summer, St. Louis libraries host many types of free camps — yoga, chess and even a Harry Potter "Sorting Hat Camp." In 2020, camp dreams seemed far-fetched given the global cough . However , the future looked bright for libraries in St. Louis.

Instead of canceling, they brought camp into kids' homes. This is called ukulele camp. Kids signed up for it in the mail. They got a beginner's guidebook, video and an actual ukulele to play. It was all free. Advisers met with kids using virtual formats.

Joe Monahan, manager of youth services for the St. Louis library system, says that 70 camps were originally planned. Of those, 54 were held mostly.

Paula Langsam is a youth services manager who works at the Martin Luther King Junior Memorial Library in Washington, D.C. She says, "In a way, our work has changed a lot. We didn't used to do videos a lot."

ACCESS [Lexile Grade 11] Each summer, libraries in St. Louis, Missouri, has many different types of free camps that are yoga, chess and even a Harry Potter gang Sorting Hat Camp. In 2020, camp dreams seemed far-fetched that there was the global coronavirus pandemic. That did not stop St. Louis libraries, though.

Instead of being canceled, they brought camp into children's homes. So children who signed up for ukulele camp got a guidebook. They also had an actual ukulelele in the mail. It was all free. In addition, camp meetings still happened. Advisers met with new children using virtual formats.

Joe Monahan, also known as Joe Monahan, has youth services for the St. Louis library system says that of the 70 camps first started, 54 were held.

Paula Langsam, also known as Paula Langsam, is a youth services manager at the soon-to-reopen Martin Luther King Junior Library in Washington, D. We did not use to do many videos a lot.

Who manages the St Louis library kids programs?	*How many camps were scheduled, how many were run?*
Joe Monahan, Paula Langsam, St. Louis Camp Leaders	54 and 70, **70 and 54**, 70 and 0, 54 and 0

Were any camps in St. Louis cancelled?	*How did the Ukulele camp meet?*	*What camps did the libraries host?*
Yes, No	In the park, **Virtually**, Did not meet	**Yoga**, **Chess**, Pottery, **Ukulele**

Figure 3.7: Complement to Figure 3.6. Example Task for the Comprehension Study. Participants were assigned to one of five settings: original, Newsela, KiS, Finetune Baseline, and ACCESS. Participants were instructed to answer the five comprehension questions.

Study Results

Table 3.2 details the timing and number of participants for each combination of document and stimuli.

	Simplification Model				
Document Id	**Original**	**Newsela**	**Sup. Base.**	**ACCESS**	**KiS**
Marvel Show	152 (12)	209 (11)	140 (11)	209 (14)	126 (13)
Covid Libraries	167 (14)	180 (12)	182 (10)	190 (13)	171 (12)
Sustainable Food	163 (13)	144 (10)	181 (13)	242 (13)	154 (12)
Iceberg Collision	208 (14)	116 (11)	139 (12)	104 (12)	119 (12)
Version Aggregate	174 (53)	163 (44)	161 (46)	188 (52)	143 (49)

Table 3.2: Average time taken and number of participants in each of the document/stimuli combinations. Also shown are aggregates (mean time taken and total number of participants).

We ran the study on Mechanical Turk, accepting crowd-workers with 1700+ completed tasks, and an acceptance rate of 97%+. The study was active for two weeks in December 2020, and remunerated participants completing all four sections at a rate of $10/hour. When removing "brute-forced" submissions (10+ re-submissions), we are left with 244 submissions, used for result analysis reported in Table 3.3. A more detailed results table is included in Table 3.6.

Figure 3.8 shows the instructions given to crowd-worker participants for the manual evaluation.

• The entire HIT should take no more than 15 minutes:
(1) You will answer a pre-questionnaire.
(2) Read 4 short news stories and answer comprehension questions about each.
• If you believe the answer is not in the document, you can select the option "Answer not in document".
• There is no time limit for each individual document or question.
• You can leave at any point but will not complete the HIT.
• You can complete this task at most once.
• If you have a question/problem, contact us at *email*.

Figure 3.8: Instructions given to participants of the comprehension evaluation. Participants were recruited on Amazon Mechanical Turk (MTurk), on which jobs are named "HIT".

We measure two outcomes: question completion time (in seconds), and number of submissions to correctness. We performed a Kruskal-Wallis test [79] with a Dunn post-hoc test [36] for statistical significance between pairs of conditions.

In line with study objectives, simplified texts help participants complete the task faster than reading original texts, with three of the four simplified versions leading to improvements in

Model	Time (sec)	# Subs.	Comp.	CASpeed
♭ Original	174.0	4.23	1.0	1.00
♮ Newsela	163.3	5.10	1.08	**1.15**
∀ ACCESS	188.5	6.69	0.96	0.88
∃ Finetune Baseline	161.0 ∀	4.70	0.97	1.04
∇ KiS Model	**142.6** ♭ ♮ ∀	4.10 ∀	0.87	**1.06**

Table 3.3: Results of the Human Comprehension Study. We measure average completion time (Time), number of submissions (#Subs.), compression ratio (Comp.) and a compression-accounted speed-up (CASpeed). Each text version is assigned a symbol used to indicate statistical significance ($p < 0.05$).

completion times. Participants were fastest with KiS simplifications (18% faster). The KiS model led to a statistically significant speed-up compared to the originals, Newsela references, and ACCESS simplifications. ACCESS simplifications surprisingly led to a non-significant slow-down, which we attribute to a potential loss in fluency that might have confused participants.

One important factor we consider is that shorter passages (i.e., smaller compression) might lead to a speed-up regardless of simplicity. We confirm this by finding a small positive correlation between passage length and completion time of 0.09. We compute a *compression-adjusted speed-up* (*CASpeed*) ratio by: (1) computing the passage length of each simplified version, (2) linearly extrapolating the expected completion time for this passage length for original paragraphs, and (3) computing the ratio of the extrapolation to the observed completion time. If $CASpeed > 1$, participants were faster than expected for the passage length. Newsela reference paragraphs achieve the best *CASpeed*, followed by the KiS model. This suggests that good simplification can involve making texts longer.

Ablation Study

We train three ablated models, each missing a reward component to gain understanding in the value of each component of the KiS procedure.

Figure 3.1 gives a qualitative perspective on each ablation. Without fluency, the generator learns to generate incomplete sentences, without salience, it omits important information, and without simplicity, it can sometimes "complexify".

We computed complete automatic results for the ablated models, and find that each ablation leads to a decrease on an evaluation metric, confirming that all three components are necessary to generate high-quality simplifications (details in Table 3.4).

Table 3.4 details the metric results of the three ablated models, an extension to Table 3.1. An example output of each ablated model, illustrating the limitation when a score component is missing, is given in Figure 3.1.

Model	SARI	BLEU	%FKGL	%Lexile	Comp.	Cov.
KiS Full	0.709	0.526	100	72	0.85	0.636
KiS No Fluency	0.718	0.611	99	95	1.02	0.901
KiS No Salience	0.695	0.591	100	65	1.01	0.701
KiS No Simplicity	0.672	0.617	51	23	0.92	0.809

Table 3.4: Automatic results of the three ablation models. *SARI* and *BLEU* are reference-based metrics. *% FKGL* and *% Lexile* are the percentage of simplified paragraphs with a lower FKGL and Lexile score than the original paragraph. *Comp.* is the average compression ratio (# of words), and *Cov.* is the average coverage score of the simplifications.

One surprising element is that the model trained without fluency achieves higher scores on almost all metrics, compared to the full model. This surprising fact is due to the fact that without fluency, the model does not learn to generate full sentences (see the example in Figure 3.1). Instead, the model learns to concatenate high-scoring phrases together, which can boost automatic metrics artificially. In fact, the strong performance of a model generating incomplete sentences reveals a limitation of current automatic metrics, such as BLEU and SARI.

3.7   Limitations and Future Work

Improved Accuracy Scoring. The current guardrail for inaccuracy is rudimentary; trained models still generate non-factual simplifications. Recent work in fact-checking for the summarization domain [80, 90] could be adapted to the simplification domain to improve this.

   Inclusion of Supervised Signal. In this work, we establish that text simplification can be approached in an unsupervised manner. In future work, Keep it Simple could be used as a pre-training strategy, or used jointly with supervised training.

   Reproducibility of Human Evaluation. Even though we release the models, stimuli and comprehension questions used in the human evaluation, some elements of the procedure introduce randomness. Participating crowd-workers differ in literacy level which may have an effect on their performance at the task [2].

   New Settings, Domains and Languages. We limited our experiments to the simplification of English news articles following prior work, but plan to pursue other languages in the future. Similarly, because Keep it Simple does not require labeled data, it can be applied to new settings (e.g., rewriting to inverse the effects of simplification), or to new domains (e.g., legal texts).

3.8   Conclusion

We have shown that text simplification can be approached in an unsupervised manner via KiS. By optimizing a reward comprised of simplicity, fluency and salience components, KiS is able to

outperform strong supervised models on automatic metrics (+0.04 in SARI). We propose a human comprehension task to evaluate the usefulness of simplification and show that simplifications tend to lead to a measurable speed-up in task completion, with KiS texts producing the best speed-up of 18% on average. These are first steps for unsupervised text simplification, and we suggest that future work should focus on adapting the methodology to new domains (i.e., legal), non-English languages, and refining optimized rewards to take factuality into account.

Chapter 4

SummaC: Summary Consistency Benchmark

Sentence-Level NLI

$$P(Y = \text{entail} \mid D_i, S_j)$$

Document

Scientists are studying Mars to learn about the Red Planet and find landing sites for future missions. **D1**

One possible site, known as Arcadia Planitia, is covered in strange sinuous features. **D2**

The shapes could be signs that the area is actually made of glaciers, **D3** which are large masses of slow-moving ice.

Arcadia Planitia is in Mars' northern lowlands. **D4**

Summary

S1 There are strange shape patterns on Arcadia Planitia. ✓

S2 The shapes could indicate the area might be made of glaciers. ✓

S3 This makes Arcadia Planitia ideal for future missions. ✗

0.98

0.99

0.02

Document-Level NLI

$$P(Y = \text{entail} \mid \text{document, summary}) = 0.91$$

Figure 4.1: Example document with an inconsistent summary. When running each sentence pair (D_i, S_j) through an NLI model, S_3 is not entailed by any document sentence. However, when running the entire (document, summary) at once, the NLI model incorrectly predicts the document highly entails the entire summary.

4.1 Introduction

Recent progress in text summarization has been remarkable, with ROUGE record-setting models published every few months, and human evaluations indicating that automatically generated summaries are matching human-written summaries in terms of fluency and informativeness [175].

A major limitation of current summarization models is their inability to remain factually consistent with the respective input document. Summary inconsistencies are diverse, from inversions (i.e. negation), the wrong use of an entity (i.e. subject, object swapping), or hallucinations introducing entities not in the original document. Recent studies have shown that in some scenarios, even state-of-the-art pre-trained language models can generate inconsistent summaries in more than 70% of all cases [114]. This has led to accelerated research around summary inconsistency detection.

A closely related task to inconsistency detection is textual entailment, also referred to as Natural Language Inference (NLI) in which a *hypothesis* sentence needs to be classified as either entailed by, neutral or contradicting a *premise* sentence. Due to the crowd-sourcing of large NLI datasets such as SNLI [11] and MNLI [167], modern architectures have been shown to perform close to human performance at the task.

The similarity of NLI to inconsistency detection, as well as the availability of high-performing NLI models led to early attempts at using NLI to detect consistency errors in summaries. These early attempts were unsuccessful, finding that re-ranking summaries according to an NLI model can lead to an increase in consistency errors [43], or that out-of-the-box NLI models obtain 52% accuracy at the binary classification task of inconsistency detection, only slightly above random guessing [80].

In this work, we revisit this approach, showing that NLI models *can* successfully be used for inconsistency detection, as long as they are used at the appropriate *granularity*. Figure 4.1 shows how crucial using the correct granularity as input to NLI models is. An inconsistency checker should flag the last sentence in the summary (shown right) as problematic. When treating the entire document as the premise and the summary as the hypothesis, a competitive NLI model predicts with probability of 0.91 that the summary is entailed by the document. However, when splitting the documents up into premise-hypothesis pairs sentences as (visualized as edges in Figure 4.1) the NLI model correctly determines that S_3 is not supported by any document sentence. This illustrates that working with sentence pairs is a crucial part for making NLI models work for inconsistency detection.

Our contributions are two-fold. First, we introduce a new approach for inconsistency detection based on the aggregation of sentence-level entailment scores for each pair of input document and summary sentences. We present two model variants that differ in the way they aggregate sentence-level scores into a single score. $SC_{ZeroShot}$ performs a zero-shot aggregation by combining sentence-level scores using `max` and `mean` operators. SC_{Conv} is a trained model consisting of a single learned convolution layer compiling the distribution of entailment scores of all document sentences into a single score.

Second, to evaluate our approach, we introduce the SUMMAC Benchmark by standardizing existing datasets. Because the benchmark contains the six largest summary consistency datasets, it is more comprehensive and includes a broader range of inconsistency errors than prior work.

The SUMMAC models outperform existing inconsistency detection models on the benchmark, with the SC_{Conv} obtaining an overall balanced accuracy of 74.5%, 5% above prior work. We publicly release the models and dataset[1].

4.2 Related Work

We briefly survey existing methods and datasets for fact checking, inconsistency detection, and inconsistency correction.

Fact Checking and Verification

Fact checking is a related task in which a model receives an input claim along with a corpus of ground truth information. The model must then retrieve relevant evidence and decide whether the claim is supported, refuted or if there is not enough information in the corpus [157]. A key

---

[1]Code will be released when the work is published.

difference from our task lies in the distinction between consistency and accuracy. If a summary adds novel and accurate information not present in the original document (e.g., adding background information), the summary is accurate but inconsistent. In the summary inconsistency detection domain, the focus is on detecting any inconsistency, regardless of its accuracy, as prior work has shown that current automatic summarizers are predominantly inaccurate when inconsistent [100].

Datasets for Inconsistency Detection

Several datasets have been annotated to evaluate model performance in inconsistency detection, typically comprising up to two thousand annotated summaries. Datasets are most commonly crowd-annotated with three judgements each, despite some work showing that as many as eight annotators are required to achieve high inter-annotator agreement [43].

Reading the entire original document being summarized is time-consuming, and to amortize this cost, consistency datasets often contain multiple summaries, generated by different models, for the same original document.

Some datasets consist of an overall consistency label for a summary (e.g., FactCC [80]), while others propose a finer-grained typology with up to 8 types of consistency errors [63].

We include the six largest summary consistency datasets in the SUMMAC Benchmark, and describe them more in detail in Section 4.4.

Methods for Inconsistency Detection

Due to data limitations, most inconsistency detection methods adapt NLP pipelines from other tasks including QAG models, synthetic classifiers, and parsing-based methods.

QAG methods follow three steps: (1) question generation (QG), (2) question answering (QA) with the document and the summary, (3) matching document and summary answers. A summary is considered consistent if few or no questions have differing answer with the document. A key design choice for these methods lies in the source for question generation. Durmus, He, and Diab [37] generate questions using the summary as a source, making their FEQA method precision-oriented. Scialom et al. [138] generate questions with the document as a source, creating a recall-focused measure. Scialom et al. [139] unite both in QuestEval, by generating two sets of questions, sourced from the summary and document respectively. We include FEQA and QuestEval in our benchmark results.

Synthetic classifiers rely on large, synthetic datasets of summaries with inconsistencies, and use those to train a classifier with the expectation that the model generalizes to non-synthetic summaries. To generate a synthetic dataset, Kryscinski et al. [80] propose a set of semantically invariant (e.g., paraphrasing) and variant (e.g., sentence negation) text transformations that they apply to a large summarization dataset. `FactCC-CLS`, the classifier obtained when training on the synthetic dataset, is included in our benchmark results for comparison.

Parsing-based methods generate relations through parsing and compute the fraction of summary relations that are compatible with document relations as a precision measure of summary factuality. [51] parse extract `(subject, relation, object)` tuples most commonly using OpenIE

[40]. In the recent DAE model, Goyal and Durrett [52] propose to use arc labels from a dependency parser instead of relation triplet. We include the DAE model in our benchmark results.

Methods for Consistency Correction

Complementary to inconsistency detection, some work focused on the task of mitigating inconsistency errors during summarization. Approaches fall in two categories: Reinforcement Learning (RL) methods to improve models and stand-alone re-writing methods.

RL-methods often rely on an out-of-the-box inconsistency detection model and use reinforcement learning to optimize a reward with a consistency component. Arumae and Liu [4] optimize a QA-based consistency reward, and Nan et al. [105] streamline a QAG reward by combining the QG and QA model, making it more efficient for RL training. Pasunuru and Bansal [118] leverage an NLI-based component as part of an overall ROUGE-based reward, and Zhang et al. [179] use a parsing-based measure in the domain of medical report summarization.

Re-writing methods typically operate as a modular component that is applied after an existing summarization model. Cao et al. [18] use a synthetic dataset of rule-corrupted summaries to train a post-corrector model, but find that this model does not transfer well to real summarizer errors. Dong et al. [32] propose to use a QAG model to find erroneous spans, which are then corrected using a post-processing model.

Since all methods discussed above for consistency correction rely on a model to detect inconsistencies, they will naturally benefit from more accurate inconsistency detectors.

4.3 SUMMAC Models

We now introduce the SUMMAC Models for summarization inconsistency detection. We first show how we apply an NLI model out-of-the-box to generate an *NLI Pair Matrix* for a `(document, summary)` pair. We then describe two models that process the Pair Matrix and produce a single consistency score for a given summary. We describe the SUMMAC evaluation benchmark, a set of inconsistency detection datasets in Section 4.4. In Section 4.5, we measure the performance of the SUMMAC models on this benchmark and investigate components of the models, including which NLI model achieves highest performance, which NLI categories should be used, and what textual granularity is most effective.

Generating the NLI Pair Matrix

NLI datasets are predominantly represented at the sentence level. In our pilot experiments, we found that this causes the resulting NLI models to fail in assessing consistency for documents with 50 sentences and more.

This motivates the following approach. We generate a NLI Pair Matrix by splitting a `(document, summary)` pair into sentence blocks. The document is split into M blocks, each considered a

Dataset	Size		%+	IAA	Src	#Sum	#L
	Valid.	**Test**					
CoGenSumm	1281	400	49.8	0.65	C	3	0
XSumFaith	1250	1250	10.2	0.80	X	5	2
Polytope	634	634	6.6	-	C	10	8
FactCC	931	503	85.0	-	C	10	0
SummEval	850	850	90.6	0.7	C	23	4
FRANK	671	1575	33.2	0.53	C+X	9	7

Table 4.1: Statistics of the six datasets in the SUMMAC Benchmark. For each dataset, we report the validation and test set sizes, the percentage of summaries with positive (consistent) labels (**%+**), the inter-annotator agreement (when available, **IAA**), the source of the documents (**Source**: C for CNN/DM, X for XSum), the number of summarizers evaluated (**#Sum**), and the number of sublabels annotated (**#L**).

premise labeled from $D_1, ..., D_M$, and the summary is split into N blocks, each considered a hypothesis labeled from $S_1, ..., S_N$.

Each D_i, S_j combination is run through the NLI model, which produces a probability distribution over the three NLI categories (E_{ij}, C_{ij}, N_{ij}) for entailment, contradiction and neutral, respectively. If not specified otherwise, the pair matrix is an $M \times N$ matrix consisting of the entailment scores E_{ij}. In Section 4.5, we examine the effect of granularity in by splitting texts at the paragraph level or binning two sentences at a time. In Section 4.5, we explore the use of the contradiction and neutral categories in our experiments .

The example in Figure 4.1 has $M = 4$ document sentences, and $N = 3$ summary sentences, and the corresponding NLI Pair Matrix is the following:

$$X_{pair} = \begin{bmatrix} 0.02 & 0.02 & 0.04 \\ 0.98 & 0.00 & 0.00 \\ 0.43 & 0.99 & 0.00 \\ 0.00 & 0.00 & 0.01 \end{bmatrix}$$

The pair matrix can be interpreted as the weights of a bipartite graph, which corresponds to the illustration in Figure 4.1, with the opacity of each edge (i, j) representing the value E_{ij}.

The two SUMMAC models take as input the same NLI Pair Matrix, but differ in the aggregation method to transform the pair matrix into a score. Figure 4.2 presents an overview of $SC_{ZeroShot}$ and SC_{Conv}.

$SC_{ZeroShot}$: **Zero-Shot**

In the $SC_{ZeroShot}$ model, the first step consists of reducing the matrix to a one-dimensional vector by taking the maximum (`max`) value of each column. At a high-level, for each summary sentence, this
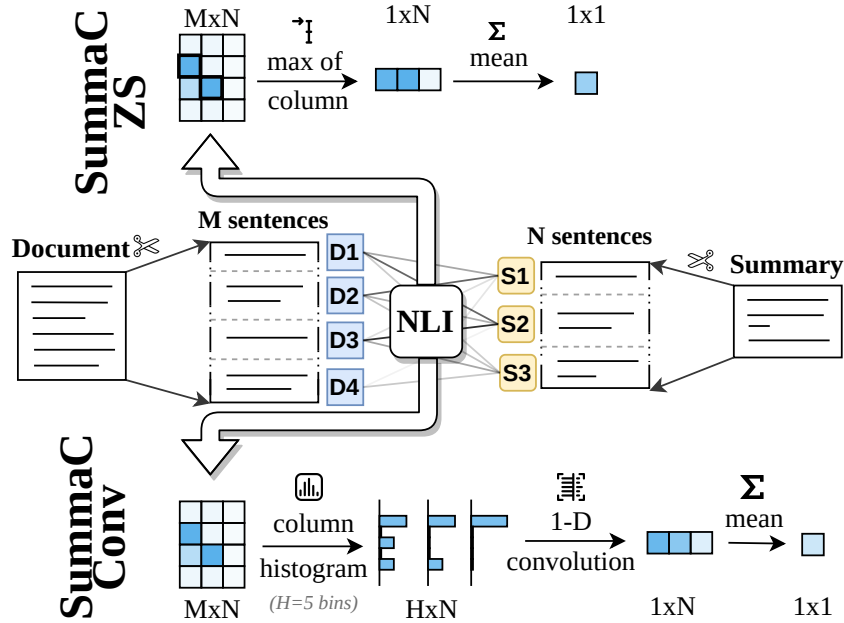
Figure 4.2: Diagram of the $SC_{ZeroShot}$ (top) and SC_{Conv} (bottom) models. Both models utilize the same NLI Pair Matrix (middle) but differ in its processing to obtain a score. The $SC_{ZeroShot}$ is Zero-**S**hot, and does not have trained parameters. SC_{Conv} has a convolutional layer trained on a binned version of the NLI Pair Matrix.

step consists of retaining the score for the document sentence that provides the strongest support for each summary sentence. For the example in Figure 4.1:

$$\max(X_{pair}, \text{axis=`col'}) = \begin{bmatrix} 0.98 & 0.99 & 0.04 \end{bmatrix}$$

The second step consists of taking the `mean` of the produced vector, reducing the vector to a scalar which is used as the final model score. At a high level, this step aggregates sentence-level information into a single score for the entire summary. In the example of Figure 4.1, the score produced by $SC_{ZeroShot}$ would be 0.67, however when removing the third sentence from the summary, the score would increase to 0.985. We experiment with replacing the `max` and `mean` operators with other operators.

Table 4.2 measures the effect of the choice of the two operators in the $SC_{ZeroShot}$ model. We explore three options (`min,` `mean` and `max`) for each operator. We find that the choice of `max` for Operator 1 and `mean` for Operator 2 achieves the highest performance and use these choices in our model.

	Operator 2		
Op. 1	**Min**	**Mean**	**Max**
Min	53.1	55.7	57.4
Mean	60.5	62.8	62.0
Max	68.8	**72.1**	69.1

Table 4.2: Effect of the choice of operator on the performance of the $SC_{ZeroShot}$ model on the SUMMAC Benchmark. `Operator 1` reduces the row dimension of the NLI Pair Matrix, and `Operator 2` reduces the column dimension.

SC_{Conv}: Convolution

One limitation of $SC_{ZeroShot}$ is that it is highly sensitive to extrema, which can be noisy due to the presence of outliers and the imperfect nature of NLI models. In SC_{Conv}, we reduce the reliance on extrema values by instead taking into account the entire distribution of entailment scores for each summary sentence. For each summary sentence, a learned convolutional layer is in charge of converting the entire distribution into a single score.

The first step of the SC_{Conv} algorithm is to turn each column of the NLI Pair Matrix into a fixed-size histogram that represents the distribution of scores for that given summary sentence.

We bin the NLI scores into H evenly spaced bins (e.g., if $H = 5$, the bins are $[0, 0.2), [0.2, 0.4),$ $[0.4, 0.6), [0.6, 0.8), [0.8, 1))$. Thus the first summary sentence of the example in Figure 4.1 would have the following histogram: $[2, 0, 1, 0, 1]$, because there are two values between $[0.0, 0.2]$ in the first column, one in $[0.4, 0.6]$ and one in $[0.8, 1.0]$.

By producing one histogram for each summary sentence, the binning process in the example of Figure 4.1 would produce:

$$\text{bin}(X_{pair}) = \begin{bmatrix} 2 & 3 & 4 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

The binned matrix is then passed through a 1-D convolution layer with a kernel size of H. The convolution layer scans the summary histograms one at a time, and compiles each into a scalar value for each summary. Finally, the scores of each summary sentence are averaged to obtain the final summary-level score.

In order to learn the weights of the convolution layer, we train the SC_{Conv} model end-to-end with the synthetic training data in FactCC [80]. The original training dataset contains one million (`document, summary`) pairs evenly distributed with consistent and inconsistent summaries. Because we are only training a small set of H parameters (we use $H = 50$), we find that using a 10,000 sub-sample is sufficient. We train the model using a cross-entropy loss, the Adam optimizer,

a batch size of 32 and a learning rate of 10^{-2}. We perform hyper-parameter tuning on a validation set from the FactCC dataset.

The number of bins used in the binning process, which corresponds to the number of parameters in the convolution layer, is also a hyper-parameter we tune on the validation set. We find that performance increases until 50 bins (i.e., a bin width of 0.02) and then plateaus. We use 50 bins in all our experiments.

4.4 SUMMAC Benchmark

To rigorously evaluate the SUMMAC models on a diverse set of summaries with consistency judgements, we constructed a new large benchmark dataset, the SUMMAC Benchmark. It comprises the six largest available datasets for summary inconsistency detection, which we standardized to the same classification task.

Benchmark Standardization

We standardize the task of summary inconsistency detection to a binary classification task. Each dataset contains (`document, summary, label`) samples, where the label can either be *consistent* or **inconsistent**.

Each dataset is divided into a validation and test split, with the validation being available for parameter tuning. We used existing validation/test splits created by dataset authors when available. We did not find a split for XSumFaith, Polytope, and SummEval, and created one by putting even-indexed samples in a validation split, and odd-indexed samples in the test split. This method of splitting maintains similar class imbalance and summarizer identity with the entire dataset.

We computed inter-annotator agreement on the dataset as an estimate for dataset quality, skipping datasets for which summaries received a single annotation (Polytope and FactCC). Table 4.1 summarizes dataset statistics and properties.

Benchmark Datasets

We introduce each dataset in the benchmark chronologically, and describe the standardizing procedure.

CoGenSumm (**Co**rrectness of **Gen**erated **Summ**aries, CGS) [43] is the first introduced dataset for summary inconsistency detection, based on models trained on the CNN/DM dataset [104]. The authors proposed that consistency detection should be approached as a ranking problem: given a consistent and inconsistent summary for a common document, a ranking model should score the consistent summary higher. Although innovative, other datasets in the benchmark do not always have positive and negative samples for a given document. We convert the dataset to the classification format by including all inconsistent and consistent summaries as individual samples.

XSumFaith (e**X**treme **Sum**marization **Faith**fulness, XSF) [100] is a dataset with models trained on the XSum dataset [106], which consists of more abstractive summaries than CoGenSumm. The

authors find that standard generators remain consistent for only 20-30% of generated summaries. The authors differentiate between *extrinsic* and *intrinsic* hallucinations (which we call inconsistencies in this work). Extrinsic hallucinations, which involve words or concepts not in the original document can nonetheless be *accurate* or *inaccurate*. In order for a summarizer to generate an accurate extrinsic hallucination, the summarizer must possess external world knowledge. Because the authors found that the models are primarily inaccurate in terms of extrinsic hallucinations, we standardize both extrinsic and intrinsic hallucinations into one inconsistent label.

Polytope [63] introduces a more extensive typology of summarization errors, based on the Multi-dimensional Quality Metric [97]. Each summary is annotated with eight possible errors, as well as a severity level for the error. We standardize this dataset by labeling a summary as inconsistent if it was annotated with any of the five accuracy errors (and disregarded the three fluency errors). Each summary in Polytope was labeled by a single annotator, and it is not possible to measure inter-annotator agreement.

FactCC [80] contains validation and test splits that are entirely annotated by authors of the paper, because attempts at crowd-sourced annotation yielded low inter-annotator agreement. Prior work [48] shows that there can be divergence in annotations between experts and non-experts in summarization, and because the authors of the paper are NLP researchers familiar with the limitations of automatic summarizations, we expect that FactCC annotations differs in quality from other datasets. FactCC also introduces a synthetic dataset by modifying consistent summaries with semantically variant rules. We use a sub-portion of this synthetic dataset to train the SC_{Conv} model.

SummEval [42] contains summarizer outputs from seven extractive models and sixteen abstractive models. Each summary was labeled using a 5-point Likert scale along four categories: coherence, consistency, fluency, and relevance by 3 annotators. We label summaries as consistent if all annotators gave a score of 5 in consistency, and inconsistent otherwise.

FRANK [114] contains annotations for summarizers trained on both CNN/DM and XSum, with each summary annotated by three crowd-workers. The authors propose a new typology with seven error types, organized into semantic frame errors, discourse errors and content verifiability errors. The authors confirm that models trained on the more abstractive XSum dataset generate a larger proportion of inconsistent summaries, compared to models trained on CNN/DM.

Benchmark Evaluation Metrics

With each dataset in the SUMMAC Benchmark converted into a binary classification task, we now discuss the choice of appropriate evaluation metrics for the benchmark. Datasets in the benchmark each proposed an evaluation method, falling in three main categories.

First, CoGenSumm proposes a re-ranking based measure, requiring pairs of consistent and inconsistent summaries for any document evaluated; this information is not available in several datasets in the benchmark.

Second, XSumHallu, SummEval and FRANK report on correlation of various metrics with human annotations. Correlation has some advantages, such as not requiring a threshold and being compatible with the Likert-scale annotations of SummEval, however it is an uncommon choice to measure performance of a classifier due to the discrete and binary label.

Model Type	Model Name	CGS	XSF	Polytope	FactCC	SummEval	FRANK	Benchmark
Baseline	NER-Overlap	53.0	63.3	52.0	55.0	56.8	60.9	56.8
	MNLI-doc	57.6	57.5	61.0	61.3	66.6	63.6	61.3
Classifier	FactCC-CLS	63.1	57.6	61.0	75.9	60.1	59.4	62.8
Parsing	DAE	63.4	50.8	62.8	75.9	70.3	61.7	64.2
QAG	FEQA	61.0	56.0	57.8	53.6	53.8	69.9	58.7
	QuestEval	62.6	62.1	**70.3**	66.6	72.5	**82.1**	69.4
NLI	$SC_{ZeroShot}$	**70.4**	58.4	62.0	83.8	78.7	79.0	72.1
	SC_{Conv}	62.7	**67.3**	63.5	**90.6**	**81.2**	81.5	**74.5**

Table 4.3: Performance of Summary Inconsistency Detection models on the test set of the SUMMAC Benchmark. Balanced accuracy is computed for each model on the six datasets in the benchmark, and the average is computed as the overall performance on the benchmark.

Third, authors of FactCC measured model performance using binary F1 score, and balanced accuracy, which corrects unweighed accuracy with the class imbalance ratio, so that majority class voting obtains a score of 50%.

The datasets have widely varying class imbalances, ranging from 6% to 91% positive samples. Therefore, we select balanced accuracy [14] as the primary evaluation metric for the SUMMAC Benchmark. This choice is based on the fact that accuracy is a conceptually simple, interpretable metric, and that adjusting the class imbalance out of the metric makes the score more uniform across datasets.

The balanced accuracy metric requires models to output a binary label (i.e., not a scalar score), which for most model requires the selection of a threshold in the score. The threshold is selected using the validation set, allowing for a different threshold for each dataset in the benchmark. Performance on the benchmark is the unweighted average of performance on the six datasets.

We choose Area Under the Curve of the Receiver Operating Chart (ROC-AUC) as a secondary evaluation metric, a common metric to summarize a classifier's performance at different threshold levels [12].

4.5   Results

We compared the SUMMAC models against a wide array of baselines and state-of-the-art methods.

Baselines

We evaluated the following models on the SUMMAC Benchmark:

NER Overlap uses an the spaCy named entity recognition (NER) model [62] to detect when an entity present in the summary is not present in the document. This model, adapted from Chapter 3, considers only a subset of entity types as hallucinations (i.e., *PERSON, LOCATION, ORGANIZATION*, etc.)

MNLI-doc is a Roberta [94] model finetuned on the MNLI dataset [167]. The document is used as the premise and the summary as a hypothesis, and we use the predicted probability of entailment as a score, similar to prior work on using NLI models for inconsistency detection [80].

FactCC-CLS is a Roberta-base model finetuned on the synthetic training portion of the FactCC dataset. Although trained solely on artificially created inconsistent summaries, prior work showed the model to be competitive on the FactCC and FRANK datasets.

DAE [52] is a parsing-based model using the default model and hyper-parameters provided by the authors of the paper[2].

FEQA [37] is a QAG method, using the default model and hyper-parameters provided by the authors of the paper[3].

QuestEval [139] is a QAG method taking both precision and recall into account. We use the default model and hyper-parameters provided by the authors of the paper[4]. The model has an option to use an additional question weighter, however experiments revealed that the weighter lowered overall performance on the validation portion of the SUMMAC Benchmark, and we use the model without the weighter.

SUMMAC Benchmark Results

Balanced accuracy results are summarized in Table 4.3. We find that the SUMMAC models achieve the two best performances in the benchmark. SC_{Conv} achieves the best benchmark performance at 74.5%, more than 5 points above QuestEval, the best method not involving NLI.

Looking at the models' ability to generalize across datasets and varying scenarios of inconsistency detection provides interesting insights. For example, the FactCC-CLS model achieves strong performance on the FactCC dataset, but close to lowest performance on FRANK and XSumFaith. In comparison, SUMMAC model performance is strong across the board.

The strong improvement from the $SC_{ZeroShot}$ to SC_{Conv} also shines a light on the importance of considering the entire distribution of document scores for each summary sentence, instead of taking only the maximum score: the SC_{Conv} model learns to look at the distribution and makes more robust decisions, leading to gains in performance.

Table 4.4 presents results with the ROC-AUC metric, the secondary metric of the SUMMAC Benchmark, echoing the trends seen with the balanced accuracy metric.

Table 4.4 details results of models on the benchmark according to the ROC-AUC metric, confirming that the SUMMAC models achieve the two best accuracy results on the benchmark.

Although SUMMAC models require processing all pairs of sentences through an NLI model, we find that the models are computationally tractable when compared to QAG methods. On a single GPU (Titan V100), SUMMAC can score 433 (`document, summary`) pairs per minute, compared to 20-30 for FEQA and QuestEval, due to their requirement to run QG and QA modules.

---

[2]https://github.com/tagoyal/dae-factuality
[3]https://github.com/esdurmus/feqa
[4]https://github.com/ThomasScialom/QuestEval

Model Type	Model Name	CGS	XSF	Polytope	FactCC	SummEval	FRANK	Benchmark
Baseline	NER-Overlap	53.0	61.7	51.6	53.1	56.8	60.9	56.2
	MNLI-doc	59.4	59.4	62.6	62.1	70.0	67.2	63.4
Classifier	FactCC-CLS	65.0	59.2	63.5	79.6	61.4	62.7	65.2
Parsing	DAE	67.8	41.3	64.1	82.7	77.4	64.3	66.3
QAG	FEQA	60.8	53.4	54.6	50.7	52.2	74.8	57.7
	QuestEval	64.4	**66.4**	**72.2**	71.5	79.0	87.9	73.6
NLI	$SC_{ZeroShot}$	**73.1**	58.0	60.3	83.7	85.5	85.3	74.3
	SC_{Conv}	66.9	65.6	62.6	**93.0**	**86.1**	88.7	**77.2**

Table 4.4: Performance of Summary Inconsistency Detection models on the test portion of the SUMMAC Benchmark with the ROC-AUC metric. The metric is computed for each model on the six datasets in the benchmark, and the average is computed as the overall performance on the benchmark.

		Benchmark (B Acc.)	
Architecture	**NLI Dataset**	**ZS**	**Conv**
Dec. Attn	SNLI	56.9	56.4
BERT Base	SNLI	66.6	64.0
	MNLI	69.5	69.8
	MNLI+VitaminC	67.9	71.2
BERT Large	SNLI	66.6	62.4
	SNLI+MNLI+ANLI	69.9	71.7
	VitaminC	71.1	72.8
	MNLI	70.9	73.0
	MNLI+VitaminC	**72.1**	**74.5**

Table 4.5: Effect of NLI model choice on SUMMAC models performance on the benchmark. For each NLI model, we include $SC_{ZeroShot}$ and SC_{Conv} performance. BERT* corresponds to a BERT or other pre-trained models of similar size.

Further Results

We now examine how different components and design choices affect SUMMAC model performance.

Choice of NLI Model

SUMMAC models rely on an NLI model at their core, which consists of choosing two main components: a model architecture, and a dataset to train on. We investigate the effect of both of these choices on the performance of SUMMAC models on the benchmark.

Regarding model architectures, we experiment with the decomposable attention model [117], which is a pre-Transformer architecture model that was shown to achieve high performance on SNLI, as well as Transformer base and Transformer Large architectures.

With respect to datasets, we include models trained on standard NLI datasets such as SNLI [11] and MNLI [167], as well as more recent datasets such as Adversarial NLI [108] and Vitamin C [137].

Results are summarized in Table 4.5, and we emphasize three trends. First, the low performance of the decomposable attention model used in experiments in prior work [43], confirms that less recent NLI models did not transfer well to summary inconsistency detection.

Second, NLI models based on pre-trained Transformer architectures all achieve strong performance on the benchmark, and average increase of 1.3 percentage points when going from a base to a large architecture.

Third, the choice of NLI dataset has a strong influence on overall performance. SNLI leads to lowest performance, which is expected as its textual domain is based on image captions, which are dissimilar to the news domain. MNLI and VitamincC trained models both achieve close to the best performance, and training on both jointly leads to the best model, which we designate as the default NLI model for the SUMMAC models (i.e., the model included in Table 4.3).

The latter two trends point to the fact that improvements in the field of NLI lead to improvements in the SUMMAC models, and we can expect that future progress in the NLI community will translate to gains of performance when integrated into the SUMMAC model.

We did not train the NLI models, and instead relied on models available in HuggingFace's Model Hub [168]. We list the NLI models we used throughout the paper, which can be retrieved on HuggingFace's model hub[5]. BERT stands for any Pre-trained bi-directional Transformer of an equivalent size:

- `boychaboy/SNLI_roberta-base` BERT Base+SNLI
- `microsoft/deberta-base-mnli` BERT Base+MNLI
- `tals/albert-base-vitaminc-mnli` BERT Base + MNLI + VitaminC
- `boychaboy/SNLI_roberta-large` BERT Large+SNLI
- `tals/albert-xlarge-vitaminc` Bert Large+VitaminC
- `roberta-large-mnli` Bert Large+MNLI
- `tals/albert-xlarge-vitaminc-mnli` BERT Large+MNLI+VitaminC

Choice of NLI Category

The NLI task is a three-way classification task, yet most prior work has limited usage of the model to the use of the entailment probability for inconsistency detection [80, 43]. We run a systematic experiment by training multiple SC_{Conv} models which have access to varying subsets of the NLI labels, and measure the impact on overall performance. Results are summarized in Table 4.6. Using solely the entailment category leads to strong performance for all models, however, explicitly including the contradiction label as well leads to small boosts in performance for the ANLI and MNLI models.

With future NLI models being potentially more nuanced and calibrated, it is possible that inconsistency detectors models will be able to rely on scores from several categories.

---

[5]https://huggingface.co/models

			Benchmark (B. Acc.)		
E	N	C	VITC+MNLI	ANLI	MNLI
✓			**74.5**	69.2	72.6
	✓		71.2	55.8	66.4
		✓	72.5	69.2	72.6
✓	✓		73.1	69.6	72.6
✓		✓	74.0	**70.2**	**73.0**
	✓	✓	72.5	69.2	72.6
✓	✓	✓	74.0	69.7	**73.0**

Table 4.6: Effect of NLI categories on the performance of the SC_{Conv} in the benchmark. Models had access to different subsets of the three category predictions (entailment, neutral, contradiction). Experiments were performed with 3 NLI models: Vitamic C + MNLI, ANLI and MNLI.

	MNLI		MNLI + VitC	
Granularity	ZS	Conv.	ZS	Conv.
Sentence	70.3	**73.2**	72.1	**74.5**
Two Sents.	69.7	72.9	71.4	73.2
Paragraph	62.1	62.3	70.6	72.5

Table 4.7: Effect of the granularity on the performance of the SUMMAC models on the benchmark. Experiments include three distinct granularities: sentence, two sentences, and paragraph splitting.

Choice of Granularity

So far, we've conducted experiments primarily with a sentence-level granularity, as it matches the granularity of NLI datasets. One can imagine cases where sentence-level granularity might be limiting. For example, in the case of a summary performing a *sentence fusion* operation, an NLI might not be able to correctly predict entailment of the fused sentence, seeing only one document sentence at a time.

To explore this facet further, we experiment with two new granularities: (1) **paragraph-level** granularity, where both the document and summary are separated into paragraph blocks, (2) **two-sentence** granularity, where both the document and summary are separated into blocks of contiguous sentences of size two (i.e., block 1 contains sentence 1-2, block 2 contains sentence 3-4).

Table 4.7 details results of experiments varying the granularity. We find that for both an NLI model trained on MNLI, as well as the model trained on MNLI and Vitamin C, the best performance is achieved with sentence-level granularity.

Overall, sentence-level granularity outperforms coarser granularities, suggesting that sentence-fusion problems are outweighed by the use of a finer granularity.

4.6 Discussion And Future Work

Improvements on the Benchmark. The models we introduced in this chapter are just a first step at harnessing NLI models to do inconsistency detection. Future work could explore a number improvements: combining multiple NLI models predictions, or multiple granularities, for example using multi-hop reasoning [182].

Interpretability of model output. If a model can pinpoint which portion of a summary is inconsistent, some work has shown that corrector models can effectively re-write the problematic portions and often remove the inconsistency [32]. Furthermore, fine-grained consistency scores can be incorporated into visual analysis tools for summarization such as SummViz [160]. The $SC_{ZeroShot}$ model is directly interpretable, whereas the SC_{Conv} is slightly more opaque, due to the inability to trace back a low score to a single sentence in the document being invalidated. Improving the interpretability of the SC_{Conv} model is therefore of interest.

Beyond news summarization. The six datasets in the SUMMAC Benchmark contain summaries from the news domain, one of the most common application of summarization technology. Recent efforts to expand the application of summarization to new domains such as legal [75] or scholarly text will hopefully lead to the study of inconsistency detection in these novel domains, and perhaps even out of summarization on tasks such as text simplification, or code generation.

Towards Consistent Summarization. Inconsistency detection is but a first step in eliminating inconsistencies from summarization. Future work can include more powerful inconsistency detectors in the training of next generation summarizers to reduce the prevalence of inconsistencies in generated text.

4.7 Conclusion

We introduce $SC_{ZeroShot}$ and SC_{Conv}, two NLI-based models adapted to the task of summary inconsistency detection based on the key insight that NLI models require sentence-level input to work best. Both models achieve strong performance on the SUMMAC Benchmark, a new diverse yet standardized collection of the six largest datasets for inconsistency detection. SC_{Conv} outperforms all prior work with a benchmark performance of 74.5% in terms of balanced accuracy, an improvement of more than 5 percentage points. These are first steps in the adaptation of NLI models for inconsistency detection, and there are many possible avenues for further improvements and applications of our methods.

Part II

NLP-Powered Interfaces for News Readers

Chapter 5

NewsLens: Story-Centered News

5.1 Introduction

Complex news events unfold over months, and the sequence of events over time can be thought of as forming *stories*. Our objective is to generate, from publicly available news articles, story outlines and visualizations that help readers digest and navigate complex, long-lasting stories across a large number of news articles. We attempt this construction of stories by building a dataset with multiple news sources, exploiting the overlap in coverage by different sources. Our contributions include:

1. A method for creating a dataset of articles from multiple sources across a decade from scratch,

2. A topic detection method that handles interruption in topics,

3. A novel way to name stories, and

4. A method for clustering, rating, and displaying quotations associated with the stories.

The demo is available at `https://newslens.berkeley.edu/`.

The remainder of the chapter is organized as follows. Section 5.2 presents current related research work. The aggregation of the dataset and the creation of the timelines is explained in Section 5.3. Section 5.4 presents the interface with the created timelines, as well as the extraction process for the information shown. Finally Section 5.5 concludes the chapter and presents future work directions.

5.2 Related Work

Related work includes prior methods for generating stories from topics, for visualizing stories, and for summarizing news.

Topic Detection and Tracking refers to techniques to automatically process a streamable dataset of text into related groups called topics. In the context of news, the topics detected and tracked are commonly called stories.

Swan and Allan [153] use the Topic Detection and Tracking (TDT) and TDT2 datasets, consisting of 50,000 news articles to produce 146 stories, called clusters. The clustering process is done using named entities and noun phrases, as opposed to unigrams. They report an inability to merge clusters if there are large gaps in time with no articles, and their algorithm does not group documents in an online fashion.

Pouliquen, Steinberger, and Deguernel [125] build a large dataset of news articles, named the Europe Media Monitor (EMM). Their topic detection creates local clusters in each language. The monolingual stories are then linked across languages to form global stories. A reported drawback is the clustering cannot handle merging and splitting between disparate topics and cannot mend gaps between stories that last more than 8 days.

Ahmed et al. [1] propose an online inference model named Storylines that clusters articles into storylines which are assigned to broader topics. Emphasis is put on scalability with a goal of processing one article per second.

Poghosyan and Ifrim [122] leverage keywords in social media to generate storylines, and Vossen, Caselli, and Kontzopoulou [161] propose to use news timelines to build storylines, structured index of events in the timeline, and event relationships.

Visualizing news stories focuses on building a user interface to present a given story to help the user digest a complex story. Using the EMM dataset, Krstajić et al. [78] propose a visual analytics system to represent relationships between news stories and their evolution over time. Each story element is represented as a tile in a vertical list. Over time (x-axis), the placement of story elements is adjusted on the vertical axis according to the level of activity in the story.

Shahaf, Guestrin, and Horvitz [141] propose a "Metro map" view for a given story. Article headlines are selected in the story corpus to maximize coverage of pieces of information. The selected items are put in different "lines" of the metro maps, showing how the story developed. Only headline information is accessible on the produced metro map.

Tannier and Vernier [154] build timelines for journalistic use. Based on a user query, documents are retrieved and dates are extracted from sentences. A timeline is built where peaks represents important dates, and key dates are annotated with representative article headlines and an image from the article when available.

5.3   The newsLens pipeline

In order to build news stories over long time spans based on a variety of news sources, there are two main challenges: an organizational challenge of collecting news articles, and an algorithmic and computational challenge of building the stories. We describe our solutions to both problems.

We first describe how we use the Internet Archive to recover a dataset of news articles. Given an article dataset, we propose a lightweight pipeline to process articles into topics in a streamable fashion, so that timelines can be updated as new articles are added. The pipeline we propose has the following stages: extracting keywords from articles, creating *topics*: local groups of articles in time, solidifying the local topic clusters into *stories*: long-ranging sets of articles that share a common theme, and automatically naming the stories. We then present timing measurements for each step of the pipeline.

Collecting news articles

For each article in our dataset, we require some information, from which we can build the features needed for our processing. The minimum information required is: the publication date, the url, the headline, and the content of the article. Most common news sources build their news websites with specific patterns, to make their articles easier to index. For instance, CNN.com, France24.com and NYTimes.com article urls match the following regular expressions, respectively:

http://cnn.com/yyyy/mm/dd/*
http://france24.com/en/yyyymmdd*
http://nytimes.com/yyyy/mm/dd/*

Table 5.1: Number of news articles collected for each source in the Newslens collection.

Source Name	# articles
reuters.com	1.2 million
allafrica.com	1 million
foxnews.com	475000
washingtonpost.com	440000
telegraph.co.uk	390000
france24.com	250000
nytimes.com	230000
cnn.com	140000
theguardian.com	51000
Other sources	166000

We collected 20 such patterns from globally recognized English-language news sources, and collected all news articles matching these patterns through the Internet Archive's advanced search interface. We start our collection on January 1st 2010 and collect until the present time. The publication date is extracted from the url pattern, and we access the news article's webpage to extract the headline and content.

A somewhat unexpected complication we faced was the process of deduplicating some articles. Some news agencies publish up to 7 different versions of a news article, each with a very minor change (for instance, to the headline, or by adding or removing a single sentence to the content). Because we use counts of articles to measure importance and create stories, it is important to remove duplicate articles. We apply a simple but effective method:

1. For a given source, group articles into small ranges of time (e.g. 1 week),

2. Compute bag of word vectors for each article,

3. Transform the bag of words for each group into a tf-idf matrix,

4. If two articles are above a certain cosine similarity, they are assumed to be duplicates,

5. Retain only the most recent article, as it may have corrected information.

Roughly 10% of articles across all sources are deleted in the deduplication process. After deduplication, our dataset contains 4 million news articles in English, or an average of 1,500 articles per day. The detail of number of articles per source is given in Table 5.1. A study of how article duplicates are created and the types of modifications that news sources create would be interesting.

Figure 5.1: Local topic graph from June 10th 2014 to June 16th 2014. Nodes on the graph are news articles, edges are placed according to our method. Color of the node represents the topic assigned by community detection. Even though the Ferguson and Hong Kong protests form a single connected component, they get assigned to different communities. Keywords are placed on the display for convenience of the reader.

Generating the topics

Extracting article keywords

We use a standard method to extract keywords from an article's content. Given a set of articles with no keywords, we represent each document as a bag of words vector. We apply a tf-idf transform on the bag of words corpus and select a word w_i in document d_j as a keyword for the document if the tf-idf score $S(w_i, d_j) > T$, where T is manually set. If we are trying to extract keywords for a large dataset, we process the articles in batches of a fixed size and randomize the order in which we take the articles. Each article is processed a single time. The keywords are lemmatized and lowercased. Although simple, this approach is effective: for the France24 news article with headline:

Battle to retake Mosul from Islamic State group has begun, says Iraqi

the keywords obtained are:

   shiite, force, abadi, militia, mosul, iraq

Local topic graph

There is not one clear definition of when two articles are about the same "story" in news. Our goal is to cluster articles into local groups we call topics, which are then merged over time into stories. We define two articles to be in the same topic if they share several keywords, and are published in a close range of time.

  We propose to group articles into common local topics by building a graph of articles. The algorithm for building the graph is:

1. For each article a_i over a small range of N days, prepare keyword set kw_i

2. Articles (a_i, a_j) are assigned an edge between them if $\|kw_i \cap kw_j\| \geq T_2$, where T_2 is a manually set threshold.

An example graph obtained over a range of 6 days is shown in Figure 5.1. The graph obtained is not connected and has several components. One can think at first that each component represents a story, however, it is possible for different densely connected topics to erroneously connect over a few edges. This can be seen on Figure 5.1, where two large components: the Ferguson and Hong Kong protests are loosely connected by a single edge. To avoid the problem of merging topics due to erroneous edges, we use a community detection algorithm, whose role is to find correct assignment of the nodes into communities that maximize a quality function on the communities obtained. We use a standard community detection algorithm, the Louvain method [10], which is both lightweight and efficient at finding the correct clusters. It can be seen in Figure 5.1 that the Louvain method correctly assigns the two protests to different communities.

From topics to stories

So far we have presented a method to group articles into topics that are local in time. However, it is not computationally tractable to process the graph for a large number of days, given that we have a total of $N \simeq 3000$ days to process. Apart from the computational complexity, we would like a streamable method where adding new articles updates already existing stories and creates new ones, while avoiding recomputing all stories from scratch. The method we propose to merge topics into long-ranging stories is two-fold: a sliding window to enlarge the topics, and a topic matching process for stories that might be interrupted in time.

  The first step is to run the local topic assignment in chronological order using a sliding window. For instance, if we choose $N = 5$ for the number of days in a local graph, and 50% for the window overlap, the topic assignment is first run for days 1 to 6, and then run for days 3 to 8, etc. This sequence of overlapping graph clustering creates interesting dynamics. Linking, splitting, and merging are three phenomena we believe are important for story generation from topics.

  Linking consists of assigning a topic from a preceding graph to a topic in the current graph: given a cluster in the current graph, if a majority of nodes in the cluster have previously been

assigned another topic (in a previous graph, because of the sliding window), no new topic is created, and the cluster is assigned to the old topic, enabling topics to span more than N days. Linking happens for instance on the story about the French elections, that lasted more than 5 days: the articles from the first 5 days formed a topic, and as later articles appear, they are linked into this topic that already exists. The story experiences no interruption greater than 5 days (the span of the window) from January 15th to May 10th 2017, and linking combines all articles in a single topic.

Splitting occurs when one topic is later on divided into 2 distinct topics: it can happen that a topic's start, a few initial articles are clustered together, and then diverge into clusters that are detected as separate by the community detection. In this case, the smallest cluster gets assigned to a new topic. An example of splitting: the shooting of Jo Cox (Brexit story), and the Orlando Shooting occurred within a few days of each other. The first articles covering each topic were at first assigned in the same topic, due to enough common keywords (shooting, death, killing, etc). However as each story grew with new articles, the topics became more distinct, at which point the topics were split.

Merging is similar to linking: if a current cluster found contains articles that have already been assigned to two distinct old topics, both topics are merged. An example of merging: the "Olympics in Rio" and the topic related to "Athletes worried about the Zika virus" were at first separated, but as the Athletes arrived in Rio, the stories were merged. This does not occur as often as linking and splitting.

A story is what emerges when many local topics are linked or merged. With linking, we see how local topics can be connected into stories with an unbounded time span. As long as a topic has new articles appearing continuously, all articles are linked to the same topic, and the story grows.

The assumption that a story must be uninterrupted is constraining, as some stories can have arbitrarily large gaps in time. Consider the "MH317 Malaysia Airline plane crash" story shown in Figure 5.3b, where new evidence was found a few months after the crash, and then again years after the crash happened. The second step for creating stories is to merge topics into a common story if they do not overlap in time but are similar enough in keyword distribution. We build a vector $v(t_i)$ for topic t_i which contains the counts of keywords in all articles of topic t_i. When a new topic t_j is created, its similarity to old topics is computed using a cosine similarity:

$$sim(t_i, t_j) = \frac{v(t_i) \cdot v(t_j)}{\|v(t_i)\| \|v(t_j)\|}$$

If the similarity is above a threshold T_3, and the two topics do not overlap in time significantly, the topics are merged. The final topics obtained after these two steps represent the stories we will display in our interface. The choice of T_2 and T_3 affect the precision and recall of the algorithm. Increasing T_2 reduces the number of edges on the graph, reducing the number of articles placed in topics. In our implementation, we choose high thresholds ($T_2 = 4$, $T_3 = 0.8$), which limits the number of errors (high precision). The drawback is that only 10% of articles of the overall dataset get assigned to topics. When setting $T_2 = 3$, the number of articles in topics raises to 20%, but we expect more incorrect topics to be created.

Naming stories

Finding a good name to represent the story that can encompass several thousands of articles is challenging. We propose a simple system based on observations of what makes a good title for a topic. Here are examples of good titles we want to be able to pick: *"North Korea nuclear tests"*, *"Ukraine crisis"*, *"Ebola outbreak"*, *"Brexit vote"*, *"Paris attacks"*. The features these names have in common are:

1. A story name is a noun phrase,

2. It contains a proper noun (entity),

3. It contains a common noun or word, and

4. One of the words is abstract (test, crisis, outbreak, ...).
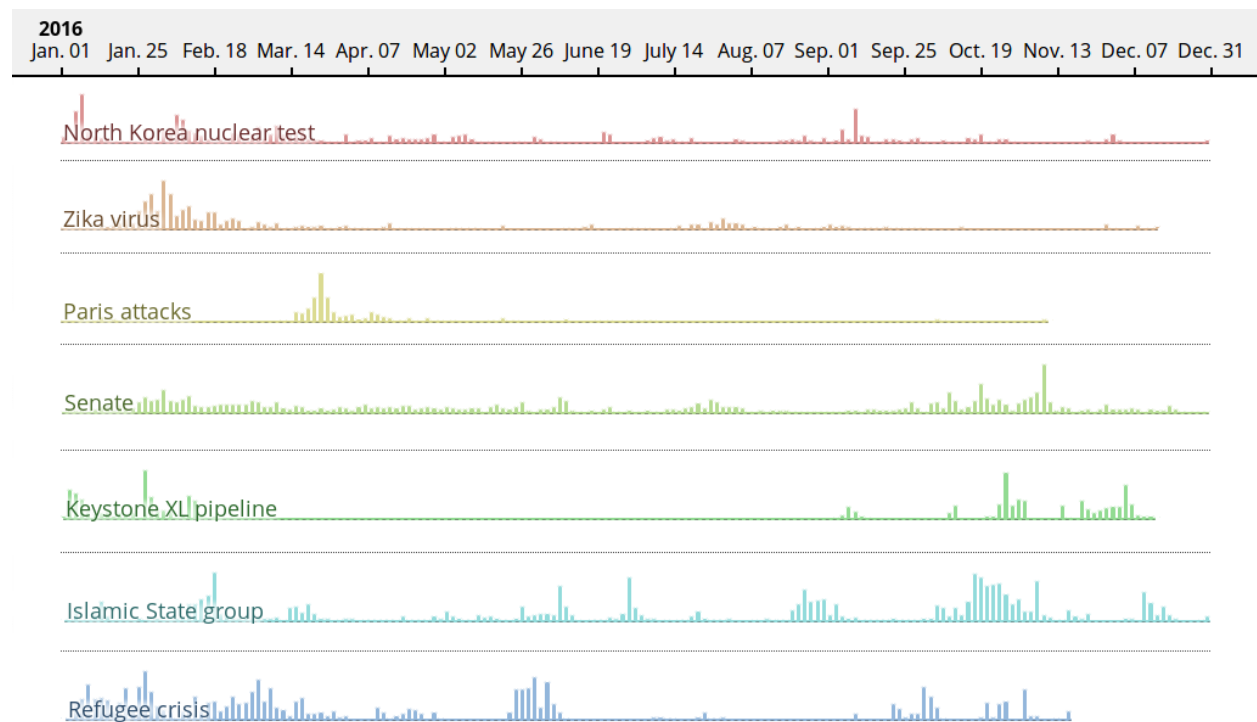
Figure 5.2: "lanes" interface. The 7 stories with most articles in 2016 are shown in timeline format.

For each headline in our story, we extract all maximal noun phrases and assign a score to each. For example, in the headline below (from telegraph.co.uk), noun phrases are underlined:

Pakistan frees Taliban prisoners to help Afghan peace process
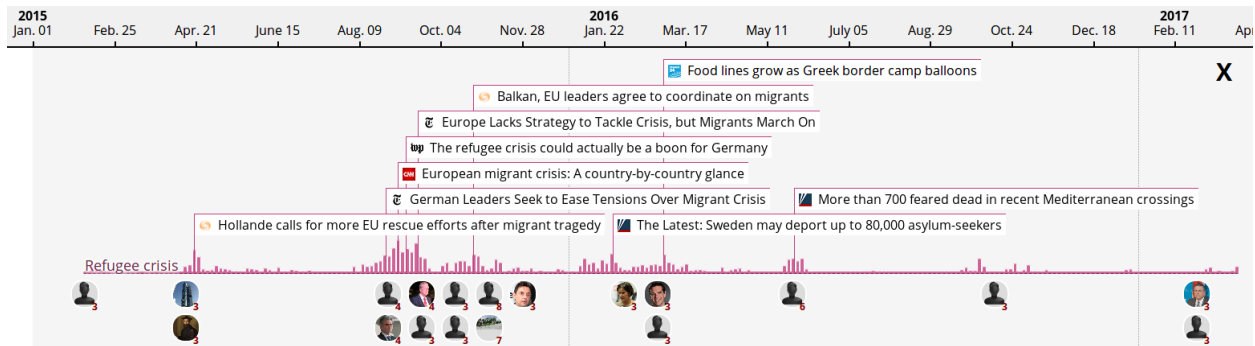
(a) "Refugee crisis" story. Top to bottom: time legend, article headlines, timeline, and quote tiles.
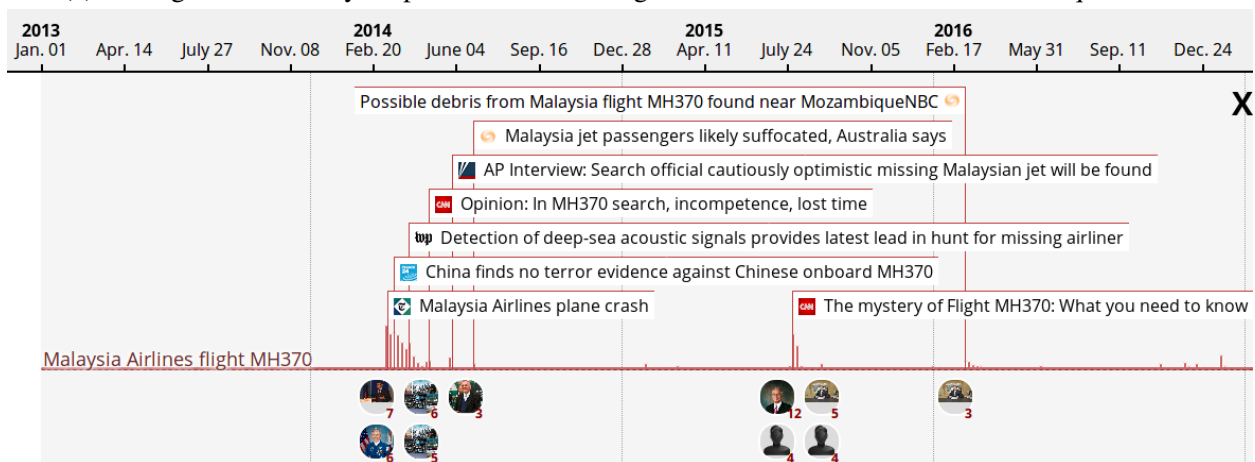
(b) Timeline of the "Malaysia Airline flight MH370", it has large time gaps with no articles.

Figure 5.3: Examples timelines in the NewsLens system.

Notice that noun phrases such as "peace process" and "prisoners" are not proposed as they are enclosed in a larger (maximal) noun phrase. The highest scoring noun phrase is chosen as the name of the story. Here are the features used to score a noun phrase p:

1. $f_1(p) = 1$ if there is a proper noun else 0

2. $f_2(p) = 1$ if there is a common noun else 0

3. $f_3(p) = log_{10}(count(p))$, where $count(p)$ is the number of occurrences of phrase p in all headlines of the story

4. $f_4(p) = \sum_{w \in p} f(w)$, w are the words in p, $f(w)$ is the frequency of w in the titles

5. $f_5(p) = \max_{w \in p} abstractness(w)$, where $abstractness(w)$ is a word abstractness measure [71]

Process name	Time per unit	Total time
Internet Archive	4 min / source	80 min
Populating articles	0.05 sec / article	2.3 days
Extracting keywords	0.01 sec / article	12 hrs
Creating stories	2 sec / day	4 hours
Naming stories	0.02 sec / story	20 min

Table 5.2: Timings of the pipeline. Time per unit, is a time per processed element. Total time is when running the pipeline on the entire dataset.

6. $f_6(p) = length(p)$, number of words in p

The final score is then computed as a linear combination of the features:

$$score(p) = \sum_{i=1}^{6} \lambda_i f_i(p)$$

We choose the λ_i manually, and $p_{final} = \arg\max_p score(p)$. The five titles presented above are results for some of the major stories available in the system.

Processing Times

The processing speed determines the system's capacity, if it is to run in real-time. Table 5.2 presents the speed per unit for each stage of the pipeline, as well as the total time spent when processing 20 sources, with 4 million articles over 7 years.

5.4 Visualizing stories with lanes

We have now presented a method to retrieve 4 million news articles and organize them into more than 80000 stories. Many of these stories have hundreds or thousands of articles. We are posed with the visualization challenge of displaying content in an understandable manner. The following section introduces *lanes*, the interface we propose to represent stories. Lanes is composed of three components: a timeline, article headlines and quotes tiles. Figure 5.3a presents two example lanes generated by our system.

Story timeline

The overall interface is framed on the x-axis representing time, each element added has a given x-position representing its occurrence within the story. We use a timeline as the main visual representation of the topic. The x-axis represents time, and the y-axis represents the number of

articles in a given short period of time. This timeline creates a shape the user can identify the story with. Figure 5.2 shows the timelines of the 7 stories in year 2016 with most news articles. The assumption we follow is that major events in a topic lead to more news articles in a following short period of time, which can be made prominent in the timeline of the overall topic by a peak. For example, in Figure 5.2, it appears that the most active periods for the story "Keystone XL pipeline" are in February, October and November 2016.

The timelines of Figure 5.2 help the user see "when" action occurred in a given story. The following two subsections present the annotations added when a user clicks on a chosen timeline. The annotations help understand the "what" and the "who" of the timeline, respectively.

Headline selection

Because we assume that peaks in the timeline of the story correspond to key times in the story, we propose to annotate these points for the reader. We sample news articles from peak periods of the story and add their headlines as annotation to the timeline. This allows the user to get an idea of what occurred during that period of the topic. The headline is clickable and takes the user to the article's original URL. This enables the user to access articles about a topic that can be several years old. When selecting which article to display for a given peak, we randomly sample an article. Added to the article headline is an image icon representing the logo of the news source, which helps the user know the source of the headline at a glance.

There can be stories where many peaks happen in a short period of time, in which case the visualization would become cluttered. We impose a hard constraint in the visualization: headline annotations cannot overlap, and they are placed on a number of "rows" above the timeline. A maximum number of headline rows is allowed, and if a headline cannot be placed because of a lack of space, it is not displayed. Headlines are placed in decreasing order of their peak heights, so that more "more important" peaks get placed first.

Quote ranking and selection
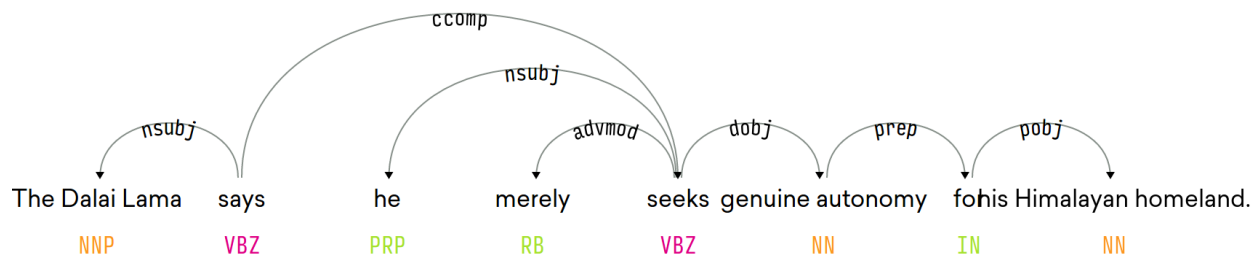
Figure 5.4: Dependency tree of a quote sentence illustrating how extraction process. This figure was generated using a modified version of displaCy.

We assume showing headlines annotations on the timeline helps the news reader answer the "what" of the story. We are experimenting with adding additional kinds of information to the interface. The first of these is quotations extracted from the article that are assumed to be important. Quote extraction is an active field of research [124, 112]. Our objective is to build a simple system to experiment with ranking and displaying the quotes. This process is done in 3 steps: entities are extracted, quotes for these entities are extracted and then grouped and scored for importance.

We extract entities from all articles using an NLP library named spaCy. In order to reduce entity duplication, we proceed with a simple entity linking process leveraging Wikidata [162]. Each entity string is searched through Wikidata's search interface. Wikidata provides unique identifiers that match the search query. The first identifier in the query result is associated with the entity string. This allows us to merge entities such as: "Obama", "Barack Obama", "Mr. Obama", etc

Entity disambiguation is a complex task, and although Wikidata is a first step in resolving entities, it also introduces errors. For instance, many news articles mention "Washington" as the author of a quote. When searching for Washington in Wikidata, the first entry that appears is "George Washington" instead of the city of Washington D.C. Additional patterns verifying the span of life and entity types could be put in place, but overall, this is a complex task and we will introduce more sophisticated entity recognition in future work.

Once entities are extracted, the next step is to attribute quotes to the entities. To extract quotes, we look at each individual sentence in our corpus and determine whether it is a quote by a known entity. The method for quote extraction is the following:

1. The sentence is parsed into a dependency tree

2. Check if the subject (NSUBJ) of the root verb of the sentence is a known entity

3. Check if the lemma of the root verb is in a predefined list (say, tell, state, ...)

4. Check if the root has a complementary clause

5. If all checks are validated, extract the pair (entity, quote)

For example given the sentence from a Reuters article:

> The self-exiled Dalai Lama says he merely seeks genuine autonomy for his Himalayan homeland.

The dependency tree for this sentence is shown in Figure 5.4. We can see that for this sentence, all three conditions are met and the quote pair extracted is: (Dalai Lama, "he merely seeks genuine autonomy for his Himalayan homeland."). The dependency parsing is also achieved with the spaCy library.

This process does not extract all quotes as the pattern recognition we propose is fairly rigid. For now, we accept the low recall for a high precision in the quotes extracted, as we assume users would react more negatively to erroneous quotes than missing quotes. This produces on average 2 quotes per news article, which can represent thousands of quotes for a single story, which is too much to show to users. We propose a simple way to cluster quotes together to find important quotes.

Quotes are transformed into bag of words vectors, and the tf-idf transform is applied to the quote vector corpus. Quotes can then be compared using a cosine similarity measure. Two quotes are judged to be in the same "quote cluster" if they are from articles that are close in time, and they meet a minimum cosine similarity.

Once quote clusters are obtained, the size of the cluster is our measure for the quote cluster's importance. This assumes that a quote that is mentioned by several journalists from various sources has more importance in the story.

We can now rank quotes in order of importance and show a limited number of quotes in the "lanes interface". Each quote cluster is represented by an image tile of the entity speaking. When clicking on a tile, a frame showing the list of quotes in the cluster opens. Figure 5.5 shows one result of opening a quote tile: four quotes from the cluster are displayed, as well as the source from which the quote is extracted. Clicking on the quote opens the article from which the quote was extracted. In this example, we can see that the quote cluster contains quotes from Reuters, CNN
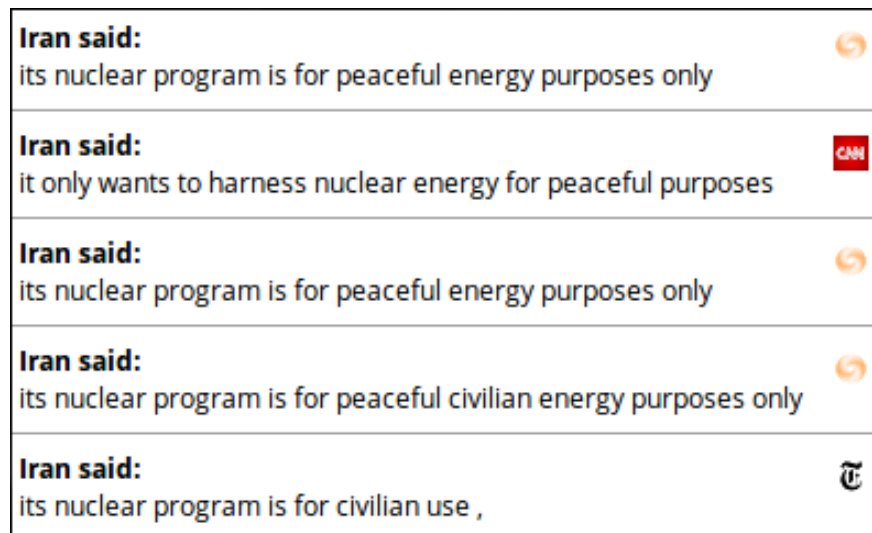
Figure 5.5: Interface that opens upon a user's click on a quote. Quotes shown were assigned to a common cluster in the story named "Iran nuclear talks"

and the NYTimes. The phrasing of each quote is slightly different, showing that sources modify and specify detail in their quote.

The lanes interface presents the stories as timelines annotated both with headlines at key times, as well as quotes representing main actors within the story.

5.5 Conclusion and Future Work

We have presented a method to build a dataset of news articles over a long range of time from several sources and an efficient, novel algorithm for organizing millions of articles into stories that

span long time ranges, despite gaps in coverage. These stories are named with a simple but effective algorithm and visualized using a lanes metaphor, providing the user with a way to view each story in more detail.

Future work includes an assessment of the accuracy of the story creation algorithm: both the accuracy within stories, verifying that articles within a given story are related, and across stories, verifying that story humans would agree with the stories we propose. We also plan to continue refining the user interface and assess it with journalists, media analysts and other relevant end users: we will compare our interface with other news aggregator systems such as Google News, to assess the usability of this approach.

Future work will also leverage the considerable related work on event detection and event pattern understanding, and incorporating that into the story creation process.

Finally, source bias and information validity are important, in the context of alternative news sources and social media. An interface that presents the facts with the source of the information in a transparent way, as well as the results of calculating biases of news sources from a computational perspective is a future direction of interest.

Chapter 6

NewsChat: A Question-Driven News Chatbot

6.1 Introduction

Chatbots offer the ability for interactive information access, which could be of great value in the news domain. As a user reads through news content, interaction could enable them to ask clarifying questions and go in depth on selected subjects. Current news chatbots have minimal capabilities, with content hand-crafted by members of news organizations, and cannot accept free-form questions.

To address this need, we design a new approach to interacting with large news collections. We designed, built, and evaluated a fully automated news chatbot that bases its content on a stream of news articles from a diverse set of English news sources. This in itself is a novel contribution.

Our second contribution is with respect to the scoping of the chatbot conversation. The system organizes the news articles into chatrooms, each revolving around a *story*, which is a set of automatically grouped news articles about a topic (e.g., articles related to Brexit).

The third contribution is a method to keep track of the state of the conversation to avoid repetition of information. For each news story, we first generate a set of essential questions and link each question with content that answers it. The motivating idea is: *two pieces of content are redundant if they answer the same questions.* As the user reads content, the system tracks which questions are answered (directly or indirectly) with the content read so far, and which remain unanswered. We evaluate the system through a usability study.

The remainder of this chapter is structured as follows. Section 6.2 describes the system and the content sources, Section 6.3 describes the algorithm for keeping track of the conversation state, Section 6.4 provides the results of a usability study evaluation and Section 6.5 presents relevant prior work.

The system is publicly available at `https://newslens.berkeley.edu/` and a demonstration video is available at this link: `https://www.youtube.com/watch?v=eze9hpEPUgo`.

6.2 System Description

This section describes the components of the chatbot: the content source, the user interface, the supported user actions and the computed system answers. Section 6.6 lists library and data resources used in the system.

Content Sources

We form the content for the chatbot from a set of news sources. We have collected an average of 2,000 news articles per day from 20 international news sources starting in 2010. The news articles are clustered into *stories*: groups of news articles about a similar evolving topic, and each story is automatically named [83]. Some of the top stories at the time of writing are shown in Figure 6.1a.
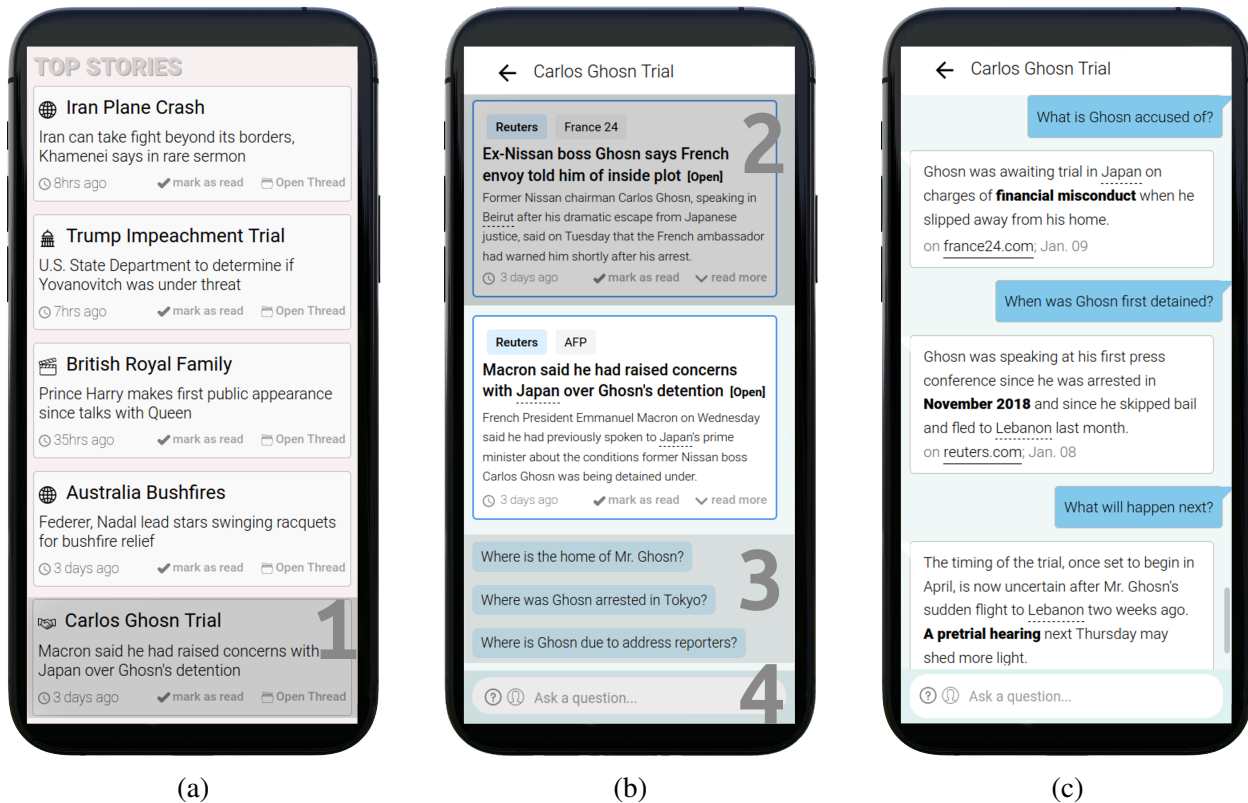
Figure 6.1: **Screenshots of the news chatbot.** (a) Homepage lists most recently active chatrooms (Zone 1 is an example chatroom) (b) Newly opened chatroom: Zone 2 is an event message, Zone 3 the Question Recommendation module, and Zone 4 a text input for user-initiated questions. Event messages are created via abstractive summarization. (c) Conversation continuation with Q&A examples. Sentences shown are extracted from original articles, whose sources are shown. Answers to questions are bolded.

User Interface

The chatbot supports information-seeking: the user is seeking information and the system delivers information in the form of news content.

The homepage (Figure 6.1a) lists the most active stories, and a user can select a story to enter its respective chatroom (Figure 6.1b). The separation into story-specific rooms achieves two objectives: (1) clarity to the user, as the chatrooms allow the user to exit and enter chatrooms to come back to conversations, and (2) limiting the scope of each dialogue is helpful from both a usability and a technical standpoint, as it helps reduce ambiguity and search scope. For example, answering a question like: "What is the total cost to insurers so far?" is easier when knowing the scope is the Australia Fires, compared to all of news.

Articles in a story are grouped into events, corresponding to an action that occurred in a particular time and place. For each event, the system forms an *event message* by combining the event's news article headlines generated by an abstractive summarizer model [85].

Zone 2 in Figure 6.1b gives an example of an event message. The event messages form a chronological timeline in the story.

Because of the difference in respective roles, we expect user messages to be shorter than system responses, which we aim to be around 30 words.

User Actions

During the conversation, the user can choose among different kinds of actions.

Explore the event timeline. A chatroom conversation starts with the system showing the two most recent event messages of the story (Figure 6.1b). These messages give minimal context to the user necessary to start a conversation. When the event timeline holds more than two events, a "See previous events" button is added at the top of the conversation, allowing the user to go further back in the event timeline of the story.

Clarify a concept. The user can ask a clarification question regarding a person or organization (e.g., Who is Dennis Muilenburg?), a place (e.g., Where is Lebanon?) or an acronym (e.g., What does NATO stand for?). For a predetermined list of questions, the system will see if an appropriate Wikipedia entry exists, and will respond with the first two paragraphs of the Wikipedia page. For geographical entities, the system additionally responds with a geographic map when possible.

Ask an open-ended question. A text box (Zone 4 in Figure 6.1b) can be used to ask any free-form question about the story. A Q&A system described in Section 6.3 attempts to find the answer in any paragraph of any news article of the story. If the Q&A system reaches a confidence level about at least one paragraph containing an answer to the question, the chatbot system answers the question using one of the paragraphs. In the system reply the Q&A selected answer is bolded. Figure 6.1c shows several Q&A exchanges.

Select a recommended question. A list of three questions generated by the algorithm described in Section 6.3 is suggested to the user at the bottom of the conversation (Zone 3 in Figure 6.1b). Clicking on a recommended questions corresponds to asking the question in free-form. However,
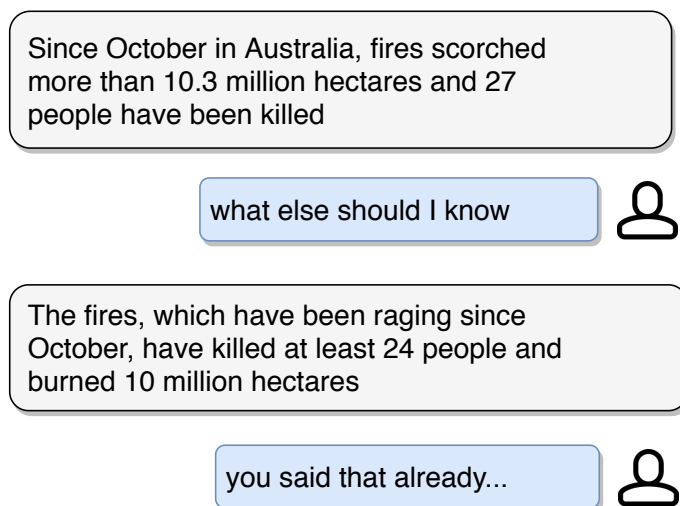
Figure 6.2: Example of repetition from the system. Repeating facts with different language is undesirable in a news chatbot. We introduce a novel question tracking method that attempts to minimize repetition.

because recommended questions are known in advance, we pre-compute their answers to minimize user waiting times.

6.3   Conversation State

One key problem in dialogue systems is that of keeping track of conveyed information, and avoiding repetition in system replies (see example in Figure 6.2). This problem is amplified in the news setting, where different news organizations cover content redundantly.

We propose a solution that takes advantage of a Question and Answer (Q&A) system. As noted above, the motivating idea is that two pieces of content are redundant if they answer the same questions. In the example of Figure 6.2, both system messages answer the same set of questions, namely: "When did the fires start?", "How many people have died?" and "How many hectares have burned?", and can therefore be considered redundant.

Our procedure to track the knowledge state of a news conversation consists of the following steps: (1) generate candidate questions spanning the knowledge in the story, (2) build a graph connecting paragraphs with questions they answer, (3) during a conversation, use the graph to track what questions have been answered already, and avoid using paragraphs that do not answer new questions.

Question Candidate Generation. We fine-tune a GPT2 language model [128] on the task of question generation using the SQuAD 2.0 dataset [129]. At training, the model reads a paragraph from the training set, and learns to generate a question associated with the paragraph. For each paragraph in each article of the story (the paragraph set), we use beam search to generate K
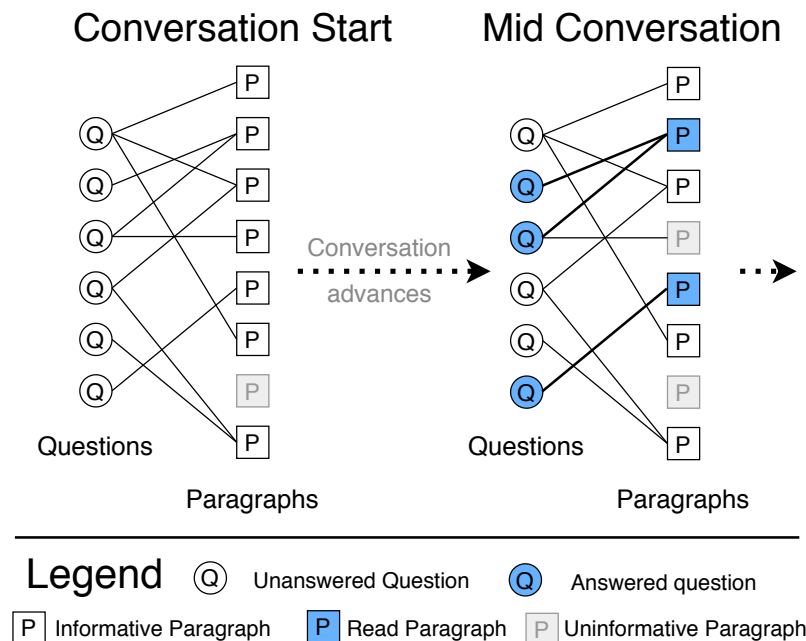
Figure 6.3: Conversation state is tracked with the P/Q graph. As the conversation advances, the system keeps track of answered questions. Any paragraph that does not answer a new question is discarded. Questions that are not answered yet are recommended.

candidate questions. In our experience, using a large beam size (K=20) is important, as one paragraph can yield several valid questions. Beam search enforces exploration, with the first step of beam search often containing several interrogative words (what, where...).

For a given paragraph, we reduce the set of questions by deduplicating questions that are lexically close (differ by at most 2 words), and removing questions that are too long ($>$12 words) or too short ($<$5 words).

Building the P/Q graph. We train a standard Q&A model, a Roberta model [94] finetuned on SQuAD 2.0 [129], and use this model to build a paragraph / question bipartite graph (P/Q graph). In the P/Q graph, we connect any paragraph (P node), with a question (Q node), if the Q&A model is confident that paragraph P answers question Q. An example bipartite graph obtained is illustrated in Figure 6.3, with the question set on the left, the paragraph set on the right, and edges between them representing model confidence about the answer.

Because we used a large beam-size when generating the questions, we perform a pruning step on the questions set. Our pruning procedure is based on the realization that two questions are redundant if they connect to the same subset of paragraphs (they cover the same content). Our objective is to find the smallest set of questions that cover all paragraphs. This problem can be formulated as a standard graph theory problem known as the set cover problem, and we use a standard heuristic algorithm [19]. After pruning, we obtain a final P/Q graph, a subgraph of the original consisting only of the covering set questions.

The P/Q graph embodies interesting properties. First, the degree of a question node measures how often a question is answered by distinct paragraphs, providing a measure of the question's importance to the story. The degree of a paragraph node indicates how many distinct questions it answers, an estimate of its relevance to a potential reader. Finally, the graph can be used to measure question relatedness: if two questions have non-empty neighboring sets (i.e., some paragraphs answer both questions), they are likely to be related questions, which can be used as a way to suggest follow-up questions.

Using the P/Q graph. At the start of a conversation, no question is answered, since no paragraph has been shown to the user. Therefore, the system initializes a blank P/Q graph (left graph in Figure 6.3). As the system reveals paragraphs in the conversation, they are marked as *read* in the P/Q graph (shaded blue paragraphs in the right graph of Figure 6.3). According to our Q&A model, any question connected to a read paragraph is *answered*, so we mark all neighbors of read paragraphs as answered questions (shaded blue questions on the right graph of Figure 6.3). At any stage in the conversation, if a paragraph is connected to only answered questions, it is deemed *uninformative*, as it will not reveal the answer to a new question.

As the conversation moves along, more paragraphs are read, increasing the number of answered questions, which in turn, increases the number of uninformative paragraphs. We program the system to prioritize paragraphs that answer the most unanswered questions, and disregard uninformative paragraphs. We further use the P/Q graph to recommend questions to the user. We select unanswered questions and prioritize questions connected to more unread paragraphs, recommending questions three at a time.

6.4 Study Results

We conducted a usability study in which participants were assigned randomly to one of three configurations:

- TOPQR: the recommended questions are the most informative according to the algorithm in Section 6.3 (N=18),

- RANDQR: the recommended questions are randomly sampled from the questions TOPQR would not select (however, near duplicates will appear in this set) (N=16),

- NOQR: No questions are recommended, and the Question Recommendation module (Zone 3 in Figure 6.1b) is hidden (N=22).

These are contrasted in order to test (a) if showing automatically generated questions is beneficial to news readers, and (b) to assess the question tracking algorithm against a similar question recommendation method with no conversation state.

Measured Value	TOPQR	RANDQR	NOQR
# participants	18	16	22
# chatrooms opened	3.2	2.9	3.1
# msgs. / chatroom	24.9 *	15.3 *	8.1
# rec. questions asked	11.9 *	8.2 *	-
# own questions asked	1.5	1.1	2.2
# total questions asked	13.4 *	9.3 *	2.2
latency (seconds)	1.84 *	1.88 *	4.51

Table 6.1: Usage statistics of the news chatbot during the usability study. Participants either saw most informative recommended questions (TOPQR), randomly selected recommended questions (RANDQR) or no recommended questions (NOQR). ∗ signifies statistical difference with NOQR ($p < 0.05$).

Study Setup

We used Amazon Mechanical Turk to recruit participants, restricting the task to workers in English-speaking countries having previous completed 1500 tasks (HITs) and an acceptance rate of at least 97%. Each participant was paid a flat rate of $2.50 with the study lasting a total of 15 minutes. During the study, the participants first walked through an introduction to the system, then read the news for 8 minutes, and finally completed a short survey.

During the eight minutes of news reading, participants were requested to select at least 2 stories to read from a list of the 20 most recently active news stories.[1] The participants were prompted to choose stories they were interested in.

The survey consisted of two sections: a satisfaction section, and a section for general free-form feedback. The satisfaction of the participants was surveyed using the standard Questionnaire for User Interaction Satisfaction (QUIS) [110]. QUIS is a series of questions about the usability of the system (ease of use, learning curve, error messages clearness, etc.) answered on a 7-point Likert scale. We modify QUIS by adding two questions regarding questions and answers: "Are suggested questions clear?" and "Are answers to questions informative?" A total of fifty-six participants completed the study. We report on the usage of the system, the QUIS Satisfaction results and textual comments from the participants.

Usage statistics

We observed that participants in the QR-enabled interfaces (TOPQR and RANDQR) had longer conversations than the NOQR setting, with an average chatroom conversation length of 24.9

---

[1]We manually removed news stories that were predominantly about politics, to avoid heated political questions, which were not under study here.

messages in the TOPQR setting. Even though the TOPQR setting had average conversation length longer than RANDQR, this was not statistically significant.

This increase in conversation length is mostly due to the use of recommended questions, which are convenient to click on. Indeed, users clicked on 8.2 questions on average in RANDQR and 11.9 in TOPQR. NOQR participants wrote on average 2.2 of their own questions, which was not statistically higher than TOPQR (1.5) and RANDQR (1.1), showing that seeing recommended questions did not prevent participants from asking their own questions.

When measuring the latency of system answers to participant questions, we observe that the average wait time in TOPQR (1.84 seconds) and RANDQR (1.88 seconds) settings is significantly lower than NOQR (4.51 seconds). This speedup is due to our ability to pre-compute answers to recommended questions, an additional benefit of the QR graph pre-computation.

QUIS Satisfaction Scores

Overall, the systems with question recommendation enabled (TOPQR and RANDQR) obtained higher average satisfaction on most measures than the NOQR setting. That said, statistical significance was only observed in 4 cases between TOPQR and NOQR, with participants judging the TOPQR interface to be more stimulating and satisfying.

Although not statistically significant, participants rated the suggested questions for TOPQR almost 1 point higher than RANDQR, providing some evidence that incorporating past viewed information into question selection is beneficial.

Participants judged the answers to be more informative in the TOPQR setting. We interpret this as evidence that the QR module helps teach users what types of questions the system can answer, enabling them to get better answers. Several NOQR participants asked "What can I ask?" or equivalent.

Qualitative Feedback

Thirty-four of the fifty-six participants opted to give general feedback via an open ended text box. We tagged the responses into major themes:

1. 19 participants (7 TOPQR, 7 RANDQR, 5 NOQR) expressed interest in the system (e.g., *I enjoyed trying this system out. I particularly liked that stories are drawn from various sources.*)

2. 11 participants (4, 3, 4) mentioned the system did not correctly reply to questions asked (e.g., *Some of the questions kind of weren't answered exactly, especially in the libya article*),

3. 10 participants (2, 3, 5) found an aspect of the interface confusing (e.g., *This system has potential, but as of right now it seems too overloaded and hard to sort through.*)

4. 6 participants (4, 2, 0) thought the questions were useful (e.g., *I especially like the questions at the bottom. Sometimes it helps to remember some basic facts or deepen your understanding*)

Measured Value	TOPQR	RANDQR	NOQR
(1) dull ... stimulating (7)	5.28 *	5.06	4.20
(1) frustrating ... satisfying (7)	5.00 *	4.43	4.00
(1) rigid ... flexible (7)	4.71	4.66	4.14
(1) terrible ... wonderful (7)	4.79	4.69	4.20
exploring new features	5.80	5.50	5.14
learning to operate	5.40	5.25	5.06
performing task is straightforward	5.40	5.56	5.20
system reliability	5.80	5.19	5.67
system speed	6.20	5.87	5.44
rec. questions are clear	5.78 *	4.87	4.28
answers are informative	5.07 *	4.44	3.64

Table 6.2: QUIS satisfaction results. Likert values on a scale from 1 to 7, higher is better unless stated otherwise. ∗ signifies statistical difference with NOQR ($p < 0.05$).

The most commonly mentioned limitation was Q&A related errors, a limitation we hope to mitigate as automated Q&A continues progressing.

6.5 Related Work

News Chatbots. Several news agencies have ventured in the space of dialogue interfaces as a way to attract new audiences. The chatbots are often manually curated for the dialogue medium and advanced NLP machinery such as a Q&A systems are not incorporated into the chatbot.

On BBC's Messenger chatbot[2], a user can enter search queries, such as "latest news" or "Brexit news" and obtain a list of latest BBC articles matching the search criteria. In the chatbot produced by Quartz[3], journalists hand-craft news stories in the form of pre-written dialogues (aka choose-your-own adventure). At each turn, the user can choose from a list of replies, deciding which track of the dialogue-article is followed. CNN[4] has also experimented with choose-your-own adventure articles, with the added ability for small talk.

Relevant Q&A datasets. NewsQA [158] collected a dataset by having a crowd-worker read the summary of a news article and ask a follow-up question. Subsequent crowd-workers answered the question or marked it as not-answerable. NewsQA's objective was to collect a dataset, and we focus on building a usable dialogue interface for the news with a Q&A component.

CoQA [131] and Quac [24] are two datasets collected for questions answering in the context of a dialogue. For both datasets, two crowd-workers (a student and a teacher) have a conversation

---

[2]https://www.messenger.com/t/BBCPolitics
[3]https://www.messenger.com/t/quartznews
[4]https://www.messenger.com/t/cnn

about a piece of text (hidden to the student in Quac). The student *must ask* questions of the teacher, and the teacher answers using extracts of the document. In our system, the questions asked by the user are answered automatically, introducing potential errors, and the user can choose to ask questions or not.

In this work, the focus is not on the collection of naturally occurring questions, but in putting a Q&A system in use in a news dialogue system, and observing the extent of its use.

Question Generation (QG) has become an active area for text generation. A common approach is to use a sequence to sequence model [34], encoding the paragraph (or context), an optional target answer (answer-aware [150]), and decoding a paired question. This common approach focuses on the generation of a single article, from a single piece of context, often a paragraph. We argue that our framing of the QG problem as the generation of a series of questions spanning several (possibly redundant) documents is a novel task.

Krishna and Iyyer [77] build a hierarchy of questions generated for a single document; the document is then reorganized into a "Squashed" document, where paragraphs and questions are interleaved. Because our approach is based on using multiple documents as the source, compiling all questions into a single document would be long to read, so we opt for a chatbot.

6.6 Resources Used

The libraries and data sources used in the described system are as follows:

Transformers library[5] used to train the GPT2-based Question Generation model and the Roberta-based Q&A model.

spaCy library[6] used to do named-entity extraction, phrase and keyword extraction.

Wikidata[7] for entity linking and collection of textual content of relevant Wikipedia pages used in special case questions.

MongoDB[8] and **Flask**[9] for storing and serving the content to the user.

SetCoverPy[10] for its implementation of standard set cover algorithms in Python.

List of news sources present in the dataset used by the system, in alphabetical order: Aa.com.tr, Afp.com, Aljazeera.com, Allafrica.com, Apnews.com, Bbc.co.uk, Bloomberg.com, Chicagotribune.com, Chinadaily.com.cn, Cnet.com, Cnn.com, Foxnews.com, France24.com, Independent.co.uk, Indiatimes.com, Latimes.com, Mercopress.com, Middleeasteye.net, Nytimes.com, Reuters.com, Rt.com, Techcrunch.com, Telegraph.co.uk, Theguardian.com, Washingtonpost.com

---

[5]https://github.com/huggingface/transformers
[6]https://github.com/explosion/spaCy
[7]https://www.wikidata.org/
[8]https://www.mongodb.com/
[9]https://flask.palletsprojects.com/en/1.1.x/
[10]https://github.com/guangtunbenzhu/SetCoverPy

6.7 Discussion

During the usability study, we obtained direct and indirect feedback from our users, and we summarize limitations that could be addressed in the system.

Inability to Handle Small Talk. 4 participants attempted to have small talk with the chatbot (e.g. asking "how are you"). The system most often responded inadequately, saying it did not understand the request. Future work may include gently directing users who engage in small talk to a chit-chat-style interface.

Inaccurate Q&A system. 32% of the participants mentioned that answers are often off-track or irrelevant. This suggests that further improvements in Q&A systems are needed.

Dealing with errors. Within the current framework, errors are bound to happen, and easing the user's path to recovery could improve the user experience.

6.8 Conclusion

We presented a fully automated news chatbot system, which leverages an average of 2,000 news articles a day from a diverse set of sources to build chatrooms for important news stories. In each room, the system takes note of generated questions that have already been answered, to minimize repetition of information to the news reader.

A usability study reveals that when the chatbot recommends questions, news readers tend to have longer conversations, with an average of 24 messages exchanged. These conversation consist of combination of recommended and user-created questions.

Chapter 7

NewsPod: An Automatic and Interactive News Podcast
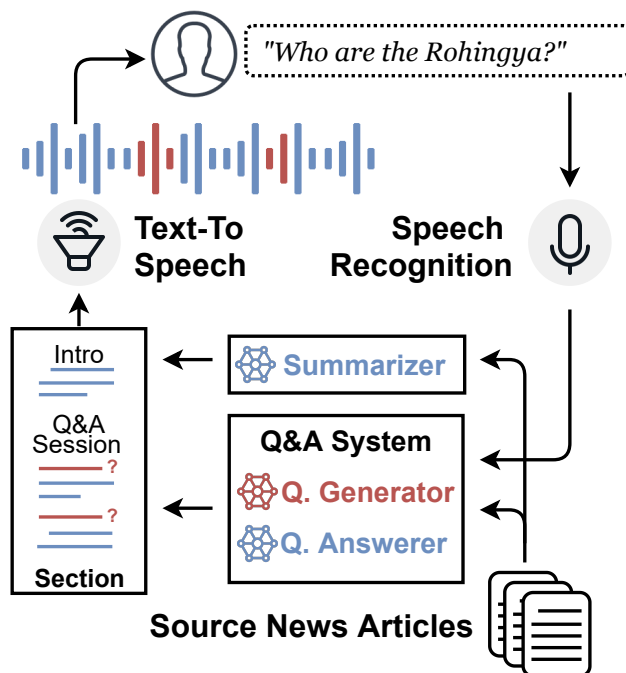
Figure 7.1: In NewsPod, a story is divided into a introductory summary, and an automated Q&A session. The listener can interact with the podcast by asking custom questions. The system integrates answers into the podcast when one is available.

7.1 Introduction

In a given week, only 4 in 10 Americans reported that they delved deeper into a particular story beyond the headlines, according to a report from the American Press Institute [134]. This statistic is partly explained by the fact that social media has become a significant part of news consumption habits, but also because reading long texts is challenging for many, with 21% of Americans aged 16 years or older having low English literacy skills [29].

Audio-based interfaces, such as radio programs or podcasts hold promise to bring in-depth news engagement to a wider audience.

When combining radio and podcast listening, 57% of U.S. adults obtain some of their news through an audio-only platform [7], in diverse settings including at home, at work, in public and private transport, while exercising and outdoors [107]. Today, more than 53 million U.S. adults own a Smart Speaker, with 62% using the speaker one hour a week or more to listen to the news [111].

News podcasts and radio shows are most commonly handcrafted, and curated for the audio platform, often requiring a team of professionals. For instance, a typical episode of *The Daily*, a popular podcast from The New York Times, involves a team with a host, three producers, two editors and an engineer.[1]

---

[1]https://www.nytimes.com/column/the-daily

On the other end of the spectrum, automation in speech synthesis has progressed rapidly in the last decade. Recent advancements in text-to-speech with WaveNet [113] and Tacotron 2 [144] have created an opportunity for a new generation of automated speech interfaces. Automated speech technology has become widely available, for instance through the Voice Browser Working Group of the W3C[2], equipping most modern browsers with standardized text-to-speech and speech recognition interfaces. Natural language understanding (NLU) and generation (NLG) have also seen accelerated development, with the state-of-the-art matching human-performance in fields such as summarization [174], question answering [181] and question generation [126].

Some news publications, such as The Economist and Bloomberg News, have leveraged this new technology to create audio versions of their printed news stories and added *playback* options, to enable reading on the go. To generate a playback, a news article is processed unchanged through a text-to-speech engine to produce an audio article played for its audience.

A playback is inherently limited, because the content is not adapted for the audio format, when compared to a manually recorded podcast. In this chapter, we build the first automatic attempt at bridging the gap between automated news reading and curated podcasts.

We present NewsPod, a system to automatically generate news podcasts. NewsPod uses state-of-the-art text summarization and question generation to create a script consisting of an introductory summary, followed by a Q&A session involving multiple automated voices. The listener can interact with the podcast by selecting stories of interest, listening to segments in a non-linear order, and asking open-ended questions during the podcast, which the system attempts to answer automatically.

We demonstrate the effectiveness of NewsPod through two usability studies. The first study focuses on evaluating the novel narrative style, which is compared to two automated baselines, and a hand-written reference. The second study measures how participants interact with the podcast, and whether including periodic breaks can encourage participant interaction.

The studies found that most participants assessed NewsPod as an efficient and enjoyable way to get informed on the news, with 80% of participants saying they would use NewsPod again.

Our work makes the following contributions:

- A design space for news podcasts, shining a light on the gap between manually curated and automated news podcasts.

- A system called **NewsPod** that automates on-demand news podcasts, featuring multiple voices to simulate a conversation, and allowing the listener to join the conversation by asking questions.

- An evaluation of NewsPod based on two usability studies, finding that the conversational podcast we propose is preferred over a non-conversational baseline, and that a majority of listeners would like to ask questions and interact with podcasts when given the opportunity.

---

[2]https://www.w3.org/Voice/

7.2   Related Work

Automating News Podcasts. To the best of our knowledge, there is no prior work automating the creation of news podcasts. The system of Yoshino and Kawahara [172] is closest conceptually. They present a chat dialogue system using news sources as the subject matter. The system proposes topics to the listener, who can ask questions about the topic. A key conceptual difference with our work is that Yoshino and Kawahara [172]'s system is designed for open-ended information navigation, requiring user input to advance the conversation, while we prepare a script for a podcast targeting a specific duration, allowing for a sequence of passive listening and active question asking, with listener interruptions allowed at any point in the conversation. Another important difference is the use in our work of multiple synthesized voices to simulate a conversation.

Some authors have written position papers and made forecasts about a future that includes automatically generated and personalized podcasts as is done in NewsPod. For instance, Dubiel, Cervone, and Riccardi [35] suggest that news presented in a conversational audio form could increase engagement, allow for the presentation of multiple viewpoints, and allow listeners to ask follow-up questions to expand their knowledge on the subject. The BBC's R&D laboratory has published a series of vision papers on the future of audio broadcasting, including forecasts for a future in which audio news topics and lengths can be personalized [15].

Lochrie et al. [95] manually built prototypes of news podcasts and performed an evaluation, which they used to propose seven recommendations for automating news podcasts. NewsPod follows four of these recommendations, namely: (1) the content should be adapted to the audio format with bite-size chunks of information (in NewsPod: questions and answers), (2) summarize key points, (3) leverage several voices specialized, (4) adapt the narrative style to the context.

Conversational Interfaces for News Recommendations. Several systems have been proposed that retrieve news articles in response to a user's queries within the format of an interactive dialogue. Bechberger et al. [8] retrieve a news event based on a user query and the user's general interests, using text chat rather than audio. Sahijwani, Choi, and Agichtein [135] investigate different approaches for recommending news content as part of an Alexa Prize conversational agent. Their system suggests a news article as part of the conversation, and if the user accepts the recommendation, a search engine is queried for news on an entity or a trending topic until the user rejects the recommendation. They find that reducing the formality of the recommendation and increasing their specificity leads to significantly higher acceptance rate from listeners.

The multiple source effect [60] posits that arguments coming from multiple distinguishable sources increases persuasion when compared to similar arguments made by a single source. Experiments have also validated the multiple source effect for video [59] as well as audio with synthetic speech [87]. Most related to our work, Kang and Gretzel [67] find that when visitors of a national park listen to a podcast tour, including multiple voices in the podcast positively impacts the social presence and enjoyment of the visitor. In NewsPod, we capitalize on the multiple source effect by dividing the content amongst several roles, each synthesized by a distinct automated voice.

Conversational Narration Style. Prior work has shown that modifying the narration style of multimedia content can have an impact on the listener's understanding, retention, and enjoyment. Personalization by altering the point of view from neutral (3rd-person) to conversational (1st- and

2nd-person) has been shown to be effective in many learning domains in reducing cognitive load and improving listener understanding [103, 99, 49]. Kang and Gretzel [67] find that when podcast tours follow a conversation format, it has a positive impact on the listener's enjoyment and social presence. By simulating a conversation in NewsPod, with voices asking questions answered by others, we attempt to create an informal environment that is inviting to the listener.

7.3 Background: Natural Language Processing

We provide relevant background on recent progress in the field of natural language processing (NLP). We leverage several of these components in creating NewsPod, and their capabilities and limitations shape our choices in the design, as well as the interaction possibilities for users of the system.

Question Answering

Recently, large neural-network based models [30, 26] have shown remarkable performance at question-answering, most commonly in the *extractive question answering* setting: given a paragraph and a question, extract a span in the paragraph (e.g., an entity, a phrase, etc.) that answers the question. Additionally, models must provide their confidence in the answer, or else state that the paragraph does not contain an answer to the question [129].

With models outperforming human performance [181] on answer extraction on the standard datasets, the community has proposed more challenging settings for question answering. We introduce two relevant to our work: *conversational question answering* and *open-domain question answering*.

In conversational question answering, introduced by the CoQA [132] and QuAC [24] projects, a model must answer a user's question as part of a longer, contextualized conversation. This is more challenging than answer extraction, because the model must be able to handle phenomena such as co-reference, and follow-up questions, removing the strong assumption that a question can be answered correctly regardless of its context.

In open-domain question answering, a model is given a question and must first retrieve relevant paragraphs from a large corpus of text before attempting to answer the question [102]. Commonly the corpus of text is very large (e.g., all of English Wikipedia [21]), creating an engineering challenge in retrieving the needle of relevant paragraphs from the large haystack of paragraphs in the text corpus. Even though recent work [70] has made great strides in making open-domain question answering computationally tractable, it still requires specialized equipment (i.e., several GPUs) and data-storage software.

In NewsPod, we use a standard extractive Q&A model which can base its answers on a limited set of relevant news articles. This choice's main advantage is its low computational requirement and simplicity in setting up, at the cost of limiting the types of questions the system can answer to factual questions answered in news articles. We discuss this choice further in Section 7.7.

Summarization

Textual summarization has become one of the most active research areas in automatic text generation, and is a common application of large language models such as GPT2 [128]. Typically the models take as input a single text document, and produce a summary by generating it one word at a time from a fixed vocabulary. The models require as training data large datasets of aligned (article, summary) pairs, and so the NLP community has curated a diverse family of datasets, ranging from legal texts [76, 143] to research papers [27] to online forums (i.e., Reddit) [73].

In the news domain, the two most common datasets are CNN/Daily Mail [61] and Newsroom [53]. The latest models, such as PEGASUS [174], have shown strong performance and have been validated through human evaluation to produce fluent and informative summaries on par with human-written summaries.

A known major limitation of automated summarization [64] is the absence of guarantee that generated summaries are factual: the model can *hallucinate* content by not faithfully representing the contents of the document it summarizes. Although the community has proposed methods to optimize factual correctness [180], the absence of a guarantee limits the usability of neural summarization at large.

In NewsPod, we use the PEGASUS model, as it is publicly available [3], and is currently the state-of-the-art for news summarization. In some cases in our dataset, a news source provides a journalist-written summary alongside with the full body content. When a human-written summary is available, we prioritize its use, and produce model-generated summaries in the absence of a human-written one.

Question Generation

In question generation [34, 115], a model is generally given a paragraph and tasked with generating a question answered by the paragraph. Question generation is generally under-constrained, as a paragraph typically answers many valid questions. In under-constrained situations, text-generation models favor most frequent outputs, which can be generic and uninteresting in the case of question generation (e.g., *What happened?*).

A standard strategy to constrain text-generation and enable models to be more specific is to condition the task on a desired first word (i.e., How, Why, etc.), encouraging the model to generate non-generic questions.

Beyond specificity of questions, current question generation models suffer from three main limitations. First, questions can lack naturalness (e.g., *What is the person ...* instead of *Who is ...*). Second, even though a model might be trained only on questions that are answered by the corresponding input paragraph, the model can still generate questions that are un-answered by the input paragraph. Third, questions might focus on un-important aspect of the paragraph (e.g., a specific figure or the name of a person mentioned).

---

[3]https://huggingface.co/models?search=pegasus

Narrative Choices		
Narrative Style	Formal News Text [Automated]	Informal Conversation [Newspod][Manual]	Story-Telling [Manual]
Number of Voices	Single Voice [Automated]	Multiple Voices [Newspod][Manual]	
Format		
Podcast Length	Brief (5-15 min) [Automated][Newspod][Manual]	Commute (20-35 min) [Newspod][Manual]	In Depth (45+ min) [Manual]
Number of Topics	Single Topic [Automated][Newspod][Manual]	Multiple Topic [Automated][Newspod][Manual]	
Podcast Frequency	On Demand [Newspod]	1+ a day [Automated][Manual]	Weekly Monthly [Manual]
Speech Source	Automatic Speech [Automated][Newspod]	Manually Recorded [Manual]	
Features		
Transcript Availability	Available [Automated][Newspod][Manual]	**Not** available [Manual]	
Music / Sounds	Not used [Automated][Newspod]	Mixed in with speech [Manual]	
Interactivity	Pause Play [Automated][Newspod][Manual]	Ask a Question [Newspod]	
Legend	[gray square] Automated Podcasts [yellow diamond] Newspod [blue circle] Manually Recorded Podcasts	

Figure 7.2: A design space of the current landscape of news podcasts. Icons represent three categories of podcasts: manually recorded podcasts, automated podcasts, and the NewsPod system.

In NewsPod: In order to account for these limitations, we generate a large number of questions, generating 7 questions for each paragraph. We obtain many more questions than we require, and use a graph-based method to select questions that are most relevant.

7.4 News Podcast Landscape Analysis

How are news podcasts constructed? In order to gain understanding and draw design principles, we performed a landscape analysis of existing news podcasts.

We first define three criteria that must be satisfied to qualify as a news podcast. First, the source of information should be solely conveyed by audio: TV news broadcasts and newspapers do not

qualify as news podcasts as their primary source of information are video and printed or online text, respectively. Second, the audio podcast must be available on the internet and downloadable on-demand: live radio shows that cover the news do not qualify, unless they are a posteriori uploaded to the internet and available upon user request. Third, a news podcast's primary subject should be about the news: covering information about current events. By contrast, literary audio books (e.g., *Harry Potter* on Kindle), entertainment (e.g.: NPR's *Wait, Wait, Don't Tell Me*) or health (e.g.: Oprah's *SuperSoul Conversations*) podcasts do not qualify as news podcasts.

A news podcast can optionally belong to a *podcast series*, a program producing podcasts of a similar format regularly.

In order to characterize the current state of news podcasts, we compiled a collection of 41 news podcasts, of which 32 are manually recorded, and 9 use automatic speech generation. We selected items for this collection through two avenues: (1) online lists of most popular news podcasts[4], and (2) visiting the website of the 20 most visited news organizations and finding the active podcast series they produce.

Based on an analysis of this news podcast collection, we present a design space [96] for news podcasts in Figure 7.2. This design space delimits the range of possibilities in news podcasting and the gap between automated and manually recorded podcasts. It is organized into three categories: narrative choices, podcast format and podcast features.

Narrative Choices

We found that podcasts use a diversity of narrative devices to convey information and create a relationship with the listener. We distinguished between three major *narrative styles* that represented a majority of podcasts: formal news text, informal conversation, and story-telling. Formal news text is equivalent to a narrator reading aloud a written news story, without modifying the format for the audio interface (e.g., BBC's *Flash Briefing*). An informal conversation narration will present the content in a sequence of conversation turns. Participants in the conversation can assist in highlighting opinions (e.g., an expert's debate on RFI's *International report*) or distinguish between themes (e.g., one journalist per story in NPR's *News Now*). A story-telling narration can be identified by several markers as in the true crime genre, including a first-person narrative (e.g., "I am not an investigative journalist myself, but..." in *Serial*) or the use of suspense. The objective of story-telling narration is often to attract a wide range of recurring listeners.

Another component of the design space is the *number of voices* present in the podcast. Podcasts mainly fall into two categories: a single voice narrating throughout, or multiple voices, usually with a main host and specialized "guest" voices, or a panel of speakers with a moderator.

All automated podcasts we analyzed had a formal news text narrative style, using a single voice. Manually recorded podcasts were more diverse, with a majority following the informal conversation narrative style and the use of multiple voices.

NewsPod Design: Breaking from previous automatic news podcasts, we adopt a conversational narration style when constructing the content, involving three voices with distinct roles.

---

[4]For example: `https://blog.feedspot.com/news_podcasts/`

Podcast Format

The overall format of the podcast is shaped by several decisions regarding the podcast length, number of topics tackled, the frequency of publication, and the type of speech used.

Podcast Length, or duration, of the collection varies widely from five to ninety minutes. Most podcasts fell into one of three ranges. Podcast briefs range from five to fifteen minutes, often giving highlights without going in depth into the topic (e.g., Marketplace's *Morning Report*). The other two length ranges are the 20-35 minute range, targeted at the average daily commute time [107], and the in-depth podcast starting at 50 minutes.

Number of Topics. The podcasts analyzed fell into two categories: single topic podcasts discuss a unique topic for the entire duration of the podcast, typically exploring the topic more in depth (e.g., Vox's *Today Explained*). Multi-topic podcasts, also referred to as briefings, are partitioned in time into several sections each covering a different topic (e.g., Fox News' *Hourly Update*). In the case of multi-topic podcasts, it is possible for a podcast to have a main topic, followed by topics covered with less depth (e.g., Monocle's *The Globalist*).

Podcast Frequency. Podcasts series have varying publication frequencies, most commonly falling into two categories: (1) one or more a day (e.g., *The Daily* by the NYTimes) which are typically shorter and/or have a focus on most recent news, (2) weekly and monthly podcasts (e.g., NPR's *Embedded*). We propose a third frequency type: *on-demand*, in which a podcast is generated at the moment a listener requests it, with the ability to pull customized and recent news. In our landscape analysis, we did not find existing news podcasts produced on-demand.

Speech Source. Speech can be integrated into podcasts in two ways: (1) through a journalist or voice artist's manual recording, or (2) synthetically generated by a text-to-speech system. Typically, synthetically-generated speech is judged to be of lower quality in terms of listening experience, although recent work has shown that synthetic voices are closing the gap with human voices even for long-form content [17].

All of the automatic podcasts we analyzed fell into the brief category (5-15 minutes); most were typically multi-topic briefings, but could also be single topic (e.g., Bloomberg News' *Listen to Article*). All automatic podcasts we analyzed were published at least once a day (with some exceptions on weekends).

Manually recorded podcasts were more diverse, spanning the entire range in podcast length and publication frequency.

NewsPod Design: The podcast is generated on demand, with the listener having control over the podcast format. Specifically, the participant can choose up to five topics to include, and can choose a desired duration.

Podcast Features

Transcript Availability. A transcript is a full-text version of the podcast content displayed in the podcast's graphical interface. It can optionally be synchronized with the audio, with words or sentences appearing as they are spoken. Transcripts are useful for accessibility purposes, as well as enabling listeners to skim and decide which portions to listen to [44]. Automatic podcast systems

are at an advantage with regards to transcript availability, as the speech is generated off of a script which can directly be used as a transcript. On the other hand, manually recorded podcast require costly post-processing in order to obtain a transcript, and we find that only eight of the thirty-two manually recorded podcasts offer a transcript in their interface.

Music and sound use in the podcast can be beneficial, both to create a sense of brand (i.e., NPR's *Short Wave*), or to indicate a transition between sections in multi-topic podcasts (e.g., Monocle 24's *The Globalist*). Some podcasts even use a background music continuously throughout the podcast (e.g., CNN's *Five Things*). In our analysis, none of the automated podcasts made use of any music or sound beyond generated speech, even though it is common practice in the manually recorded podcasts (28 out of the 32).

Interactivity. All podcasts we analyzed allow some form of basic participant interaction, at a minimum enabling participants to pause and play, and jump to a specific time-stamp using a progress bar. We propose that more complex interactions can be conceived, such as the participants asking questions which the podcast attempts to answer.

NewsPod Design: We make the transcript available to the listener. We do not make use of music or sound, but consider it for future work in Section 7.10. We enable participants to interact by asking questions at any point during the podcast, and design a usability study (Section 7.7) to study podcast listeners' interest in such interactions.

7.5   Building the Automatic Podcast

We now present the components used in automating the NewsPod podcast. Figure 7.1 gives a high-level diagram of the components and their relations. We focus on applying our procedure to the English news domain, noting that the approach could be extended to other languages, since the majority of components (such as the text-to-speech or the summarizer) have equivalents in other languages.

We first describe the elements of the NewsPod web interface shown in Figure 7.3. Second, we outline how scripts for podcast sections are automatically generated by identifying groups of news articles that can form the basis of a podcast section (§7.5), and generating a section's Q&A session (§7.5). In §7.5, we describe how the script is transformed into audio through text-to-speech. Finally in §7.5 we detail the live Q&A system we developed to respond to listener-prompted questions.

Podcast Interface

A snapshot of the NewsPod graphical user interface (GUI) is shown in Figure 7.3. The interface contains three main components: the control panel (bottom), the transcript panel (right), and the decorative wave.

The control panel. The listener has the option to play and pause the podcast, as well as skip forward and backward across sections of the podcast. A transcript button toggles the transcript panel to open and close on the right hand side. Finally, a sectioned progress bar indicates the progression through the podcast, with the beginning of each section indicated. A user can click on the progress
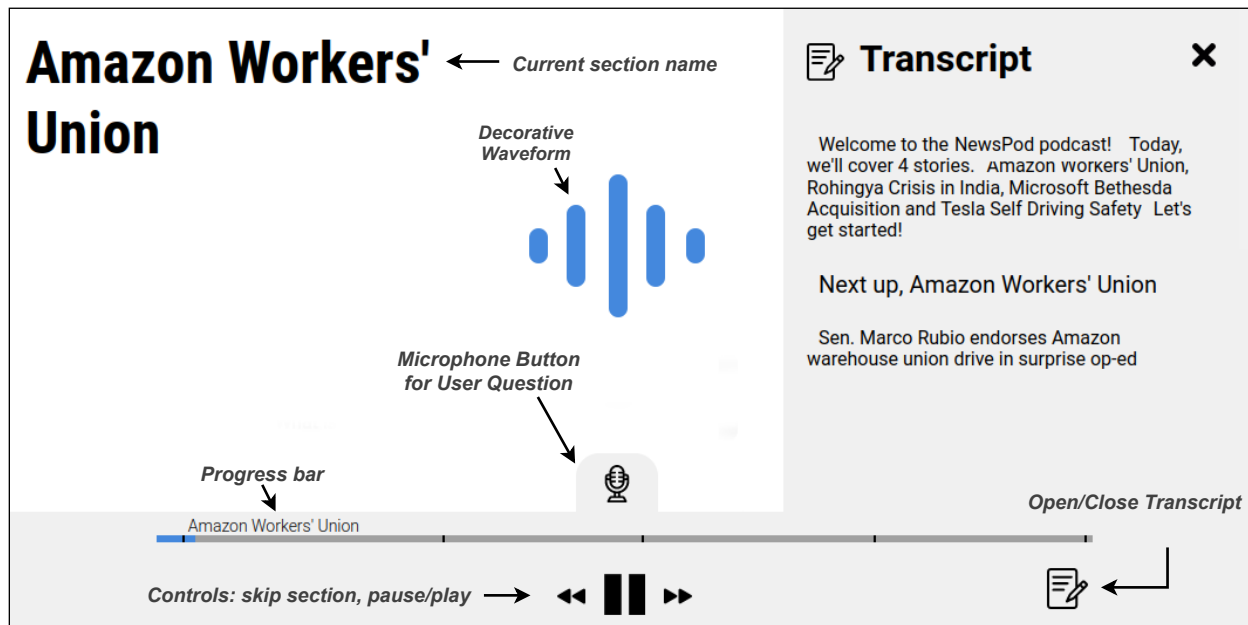
Figure 7.3: NewsPod's interface resembles other audio playback interfaces: a bottom panel contains common user controls such as pause and play and move forward and backward between podcast sections. The user can toggle a Transcript panel that opens on the right-hand side. The listener can interact with the podcast by clicking on the microphone and asking a question.

bar, which will redirect the podcast to the start of the corresponding section. At any time, the listener can click on the microphone button, which will pause the podcast and initiate the live Q&A procedure outlined in Section 7.5.

We considered not requiring the user to click to activate the microphone, and instead continuously monitor the microphone. However, this led to technical difficulty with the microphone sometimes detecting questions from the podcast's voiced text. Continuously monitoring a user's microphone is also concerning from a privacy standpoint [86], and we opted for a button-based microphone activation.

The transcript panel opens on the right-hand side of the interface, with text from the podcast added one sentence at a time, to allow the listener to follow along in real time as the podcast is playing. The transcript panel is initially closed, and in the usability study of Section 7.6, we analyze what fraction of participants opened the transcript once or more.

The decorative wave, an abstract visualization of the generated speech, brings animation to the interface. The wave changes color, turning blue, red and green based on voice identity. When the podcast is paused, the wave flattens and turns grey.

The interface is designed with responsive elements [46] and is compatible with desktop, tablet and mobile environments.

Podcast Organization

In NewsPod, a user can customize the content of the podcast by selecting which topics to include in the podcast and selecting a desired podcast duration. Based on those criteria, a podcast is generated on demand.

The podcast is composed in the following way: (1) a greeting (i.e., "Welcome to NewsPod, today we'll be covering ... "), (2) a sequence of *podcast sections*, and (3) a closing sentence (i.e. "That's it for today, thank you for tuning in").

Each podcast section is centered around a particular news topic selected by the user. Importantly, we construct podcast sections in a way that allows for adaption to user-requested duration. As an example, if a participant selects five sections and a podcast duration of five minutes, each section must have a duration of one minute. However, if the total podcast duration selected is fifteen minutes, podcast sessions can span three minutes each.

To accommodate length variability, we construct podcast sections using an *inverted pyramid* writing style common in the news domain [123]. The section is composed of an introduction consisting of a short headline and a multi-sentence summary, followed by a Q&A session revealing information in order of most important to most specific. An example podcast section with a target of 200 words is given in Figure 7.4. It contains a headline, a summary, four question/answer pairs and a quote. If a user desired a podcast half the length, this section would be truncated after 100 words, revealing only the headline, summary and two question/answer pairs.

Identifying Podcast Sections

In order to create a podcast section, we first collect a group of related articles discussing a news event, which we call the *source articles*. We then construct an introduction to the section consisting of a headline and a short summary, with a length requirement of at least two sentences and 20 words. The introduction is meant to give the listener a high-level understanding of the section before diving into the Q&A session.

The source article groups are collected by clustering a large news dataset[5] based on the live feed of around 20 international news sources in English, using a graph-based clustering algorithm [10].

We choose the introductory headline from the available headlines in the source articles, choosing based on simple rules encouraging shorter headlines, and discouraging the presence of special characters such as ":" or "-", as they tend to create difficulty in the text-to-speech engine.

The introductory summary can either be selected from an article or generated. A subset of the sources in our dataset provide hand-written summaries. If one of the source articles contains a summary that satisfies the length requirement, we select it, otherwise we generate a summary using the PEGASUS model [174], a neural-network based summarizer.

Switzerland Burqa Ban

INTRO

Switzerland votes to ban wearing of the burqa or niqab in public spaces

Referendum denounced by rights group as 'needlessly fueling division and fear', after Swiss right-wing party had framed it as verdict on the role of Islam in public life

Q&A SESSION

What would the ban mean?
The ban will mean that nobody can cover their face completely in public -- whether in shops or the open countryside.

What percentage of the population are Muslims?
Muslims make up around 5 percent of the Swiss population of 8.6 million, or about 390,000 people, most of whom have their roots in Turkey, Bosnia and Kosovo.

When was the veil banned?
France banned the wearing of a full-face veil in public in 2011 and Denmark, Austria, the Netherlands and Bulgaria have full or partial bans on wearing face coverings in public.

Which countries have similar bans?
The so-called anti-burqa vote came after years of debate in Switzerland following similar bans in other European countries -- and in some Muslim-majority states -- despite women in Islamic full-face veils being an exceptionally rare sight in Swiss streets.

QUOTE

Ines Al Shikh, spokeswoman for the Purple Headscarves feminist Muslim women's group said: "This is clearly an attack against the Muslim community in Switzerland. What is aimed here is to stigmatise and marginalise Muslims even more"

Legend: ▮Voice 1: Host ▮Voice 2: Questions ⋮Voice 3: Quotations

Figure 7.4: Example Generated Podcast Section generated based on six source news articles from *France24, Aljazeera, the Guardian, the BBC, the Middle East Eye and the Times of India*. The section is composed of an introductory headline and summary, a Q&A session, and a quote paragraph. Each text color (blue, red, green) indicates a distinct voice.
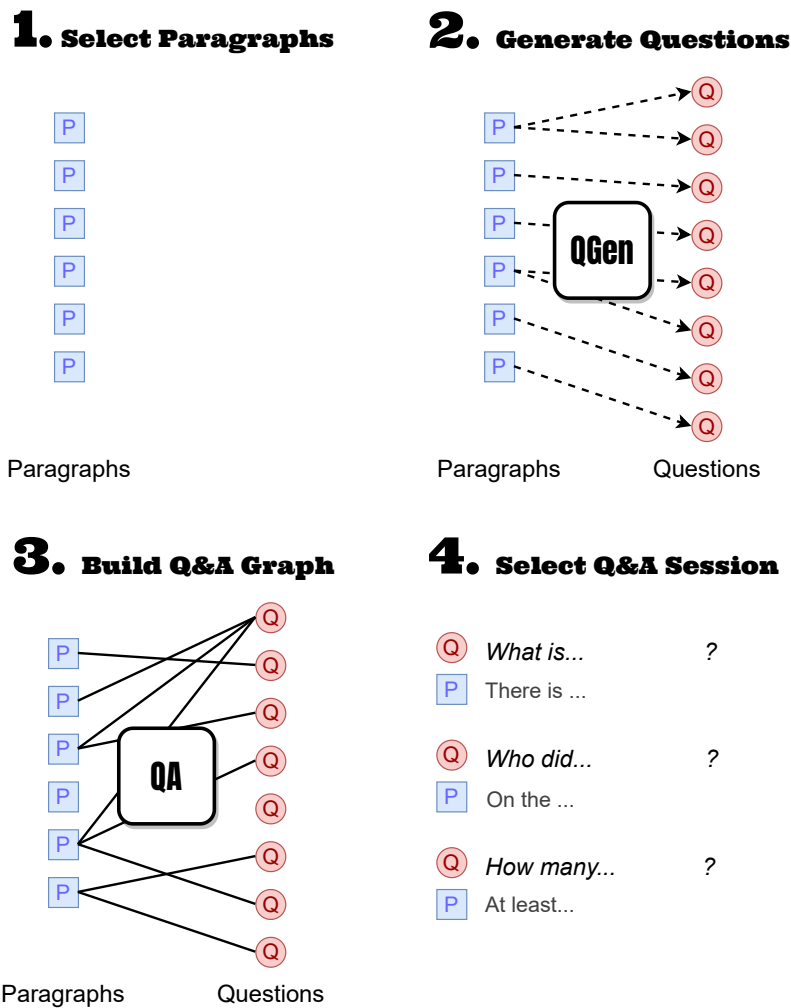
Figure 7.5: The process of generating a Q&A session from raw text follows four steps: (1) selecting paragraphs, (2) generating questions from the paragraphs, (3) using a QA model to detect which paragraphs answer which questions, and (4) finalizing the Q&A session by selecting several question/paragraph pairs.

Generating a Q&A Session

The objective of the Q&A session is to divide the content into modular pieces involving several voices, simulating a conversation and reducing monotony. The questions particularly can serve as a device to focus the listener's attention.

We use two neural network models to process the news articles of a section into a Q&A session: a question generator (*QGen*), and an extractive question-answering model (*QA*).

QGen is a model that takes as input a paragraph of text, and generates a question intended to be answered by the paragraph. We follow current state-of-the-art methodology for QGen models and finetune a large pre-trained language model (a GPT2 model [128]) on a (paragraph, question) pair dataset (in our case, the NewsQA dataset [158]).

The **QA** model is complementary to the QGen model. It takes as input both a paragraph and a question, and produces two outputs: (1) whether or not the answer to the question is in the paragraph, and (2) if it is, what span in the paragraph corresponds to the question. We follow current state-of-the-art methodology and finetune a pre-trained bi-directional Transformer (a RoBERTa-Large [94]) on two common Q&A datasets: SQuAD V2 [129] and NewsQA [158].

We now describe how we combine the QGen and QA model to transform the content of several news articles into a Q&A session. The process, illustrated in Figure 7.5 is divided into four steps: selecting paragraphs, generating questions, building a Q&A graph, and finalizing the Q&A Session.

Selecting Paragraphs. The body content of all news articles in the section serves as the basis for the creation of the Q&A session. Articles are automatically divided into individual paragraphs, further filtered down based on two *filtering criteria*: length (targeting between 10 and 45 words), and whether or not they contain direct quotations, since quotations will be processed separately.

In our dataset, articles have an average of 15 paragraphs, each containing an average of 33 words. Therefore, a Q&A session being composed from 4 source news articles could be based on around 50 paragraphs (if 80% pass the filtering criteria).

Generating Questions. We generate seven questions for each valid paragraph of content, one starting with each of the following words: `Who, What, Why, How, When, Where, Which`. This process often generates on the order of hundreds of questions, which is an order of magnitude above the requirement of a Q&A Section. The next step is to build a graph which can be used to score question relevance to narrow down which questions to include in the Q&A session.

Building the Q&A Graph. Once questions have been generated, we use the QA model to find which paragraphs answer each question. Crucially, the model attempts to answer each question with each of the paragraphs, and we build a bipartite graph between the paragraphs and questions. For each (paragraph, question) pair input to the QA model, if the QA model finds an answer, we add an edge in the graph between that paragraph and question (see Figure 7.5 for a visual illustration of the graph).

The QA graph has mathematical properties of interest: for instance, the degree of the questions indicate how many times a question was answered by distinct paragraphs. The question's degree can therefore indicate whether a question is about a key element of the topic. Vice-versa, the degree of a paragraph roughly indicates how informative it is: how many distinct questions it answers.

---

[5]Reference is omitted for anonymity purposes.

```python
def build_qa_session(QA_graph, Wtgt=200):
    # Initialize output and progress word count
    QA_session, Wsofar = [], 0
    while Wsofar < Wtgt and len(QA_graph.questions) > 0:
        # Find next most answered question
        Qi = QA_graph.select_q_highest_degree()
        # Select most informative answer paragraph
        Pi = Qi.select_answer_p_highest_degree()
        QA_session.append((Qi, Pi))
        # Update word count
        Wsofar += len(Qi.words) + len(Pi.words)
        # Remove all answered questions
        QA_graph.remove_questions(Pi.neighbors())
    return QA_session
```

Figure 7.6: Pseudocode for greedy selection of question-answer pairs to generate a Q&A Session from a Q&A graph.

Prior work [82] has presented a similar paragraph-question graph construction to track the state of conversation in a chatbot. In NewsPod, we use the properties of the graph to rank question and paragraph relevance.

Selecting the Q&A Session. Figure 7.6 provides pseudocode for the procedure to generate the Q&A Session from the Q&A Graph, also taking as an input parameter Wtgt, a target length in number of words. At a high-level, we greedily select questions that are answered the most times (an approximation for relevance). Once the top-ranked question is selected, the algorithm selects a paragraph that answers this question, creating a (question, paragraph) pair. It then removes all questions from the graph that this paragraph answers, to limit repetition of content, and iterates until reaching a target word length.

We expect that some incoherence and repetition will be present in the Q&A Session. In Section 7.6, we perform a usability study to evaluate the viability of our content selection procedure, comparing both automatic and hand-written podcast sessions.

Processing Quotations. Paragraphs that are identified to contain a quote are processed separately. Each quote paragraph is processed to extract three components: the author of the quote, a description of the speaker, and the actual quote. If at least one quote is successfully extracted, a paragraph is added to the end of the podcast section. If multiple quotes are available, the quote from the most mentioned speaker is selected.

An example of a generated section, including the introduction, the Q&A session and an extracted quote is given in Figure 7.4.

```
<speak>
    <emphasis level='none'>Ines Al Shikh,</emphasis>
    <emphasis level='reduced'>spokeswoman for the Purple Headscarves
                        feminist Muslim women's group said:</emphasis>
    <break time='300ms' />
    <emphasis level='strong'>"This is clearly an attack against the Muslim
    community in Switzerland. What is aimed here is to stigmatise and
    marginalise Muslims even more"</emphasis>
</speak>
```

Figure 7.7: We use SSML to explicitly define emphasis and silences in quote paragraphs, which is taken into account by the WaveNet speech engine.

From transcript to speech

Once the podcast's sections are written, we proceed to generating the audio using Google's Text-To-Speech API[6], which gives access to several WaveNet-based voices [113].

Of the 90+ voices available, we choose three based on recommendations from the API's documentation:

- **Voice 1** is assigned to `en-US-Wavenet-J`, a male-identified voice responsible for voicing the introductory summary, and the paragraphs answering questions in the Q&A Session. Voice 1 is the default voice, also responsible for greetings and transitions (e.g. "Next up, ...")

- **Voice 2** is assigned to `en-US-Wavenet-H`, a female-identified voice responsible for voicing questions in the Q&A Session.

- **Voice 3** is assigned to `en-US-Wavenet-D`, a male-identified voice responsible for voicing quotes in the final portion of a section.

In the example podcast section of Figure 7.4, the text is colored according to the assigned voice, with blue, red and green representing voices 1, 2 and 3 respectively. We note that we select WaveNet voices with an American English accent, as our usability studies were performed with crowd-workers mainly from the United States. However, WaveNet supports English accents from other regions, such as Australia, Great Britain and India, which could be used to further customize the experience.

The WaveNet model automatically applies some intonation and emphasis based on patterns learned in its training data (e.g., output a short silence after a comma). On top of that, the WaveNet API supports the Speech Synthesis Markup Language (SSML) [155], allowing the explicit placement of emphasis, prosodic and silence markers. We use regex-based rules to apply SSML markers on a subset of paragraphs, as we found that the markers help reduce monotony. In Figure 7.7, we illustrate how SSML is applied to paragraphs that match the quote format, in which we vary the emphasis between the author's name, their introduction, and the quote itself.

---

[6]https://cloud.google.com/text-to-speech/

Live Q&A Response

Listeners have the option to interact with the podcast by asking free-form questions during a section.

Modern web browsers have an integrated speech recognition engine using a standardized interface[7], which we use to transcribe the user's speech into text.

We use the same QA model used in Section 7.5 to build the Q&A graph to answer user-prompted questions. Specifically, we use the QA model to see if any of the section's paragraphs contain an answer to the question.

Importantly, the QA model is *extractive*, and can only answer questions that have answers in one of the source news articles. The QA model cannot combine information and synthesize text, which could be required in some open-ended questions.

Running the neural network-based QA model introduces a latency of 4 seconds on average. Therefore, when the user submits a question, the system notifies the listener with the following message: "I'll look into that, give me a moment."

If the QA model returns an answer, it is used to fill in a reply template. For instance, one of the participants in Usability Study B asked the question: "Where are Rohingya refugees from?" and received the following answer:

> I think the answer is **Myanmar**, I got it from the following paragraph. **About 170 Rohingya refugees were told they will be forcibly deported back to Myanmar where they had previously fled genocidal human rights abuses.**

On the other hand, if the QA model is not confident that any paragraph contains an answer, the system informs the listener with: "Sorry. I couldn't find the answer. If you rephrase I will try again. Otherwise I'll keep walking you through the story."

The podcast then resumes from the interruption point, skipping any sentence that it was in the midst of.

In order to understand whether a listener would be interested in interrupting the podcast with custom questions, we design a study (Section 7.7) centered around this interaction, evaluating how often interruptions occur, whether participants are satisfied with the returned answers, and analyzing the types of questions participants submit.

Implementation Details

We developed the system back-end in `Python`, and the interface front-end in `HTML`, `CSS`, and `JavaScript`. The front-end communicates with the back-end using an API built with the Flask library[8].

For the summarizer, the question generator, and the question answering models, we used HuggingFace's Transformer library[168], more specifically leveraging the implementation of models for the PyTorch library. For the summarizer, we used a pre-trained model [174], but we

---

[7]https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition
[8]https://github.com/pallets/flask

trained the Question Generator and Question-Answering models ourselves, using GPT-2 [128] and Roberta-Large [94] architectures respectively. We use a Tesla V-100 Nvidia GPU to run the models and to generate summaries and questions.

As news articles are added to our overall collection, we continuously run the pipeline described in §7.5-7.5. A typical section is based on 5-6 source news articles, with 50-100 paragraphs of content. Generating the Q&A Session is the most computationally expensive component of the pipeline, taking on average around 50 seconds. The two most compute-heavy steps are generating the questions (on average 9 seconds), and running the QA model to build the Q&A graph (on average 40 seconds), while all other steps are almost instantaneous.

In order to have granular alignment between the transcript and the audio, we generate an individual audio file for each sentence of the podcast, allowing us to add sentences gradually into the transcript panel of the graphical interface. We store audio files in the standard OGG file format [121], an open-source audio file format compatible with `HTML5` and most modern web-browsers.

7.6   Usability Study A: Narration Style

Study A Design

We perform a usability study focused on evaluating whether the automatic organization and generation of content described in §7.5-7.5 produces coherent and interesting news podcast sections.

Keeping other components fixed (such as the interface, the stories described, the speech engine), we compare four settings: two fully written by hand (Baseline and Reference), and two composed by an automated algorithm (QA Best and QA Rand). In order to ensure a fair comparison, all settings must generate sections of the same target length. For the study, we set the target at 200 words, for an expected section duration of 90 seconds, assuming a speech rate of 120-150 words per minute.

Baseline. We randomly select one of the section's source articles, and select the first k sentences such that there is a total of 200 words. We do not generate questions, and the entire section is voiced by Voice 1. This setting is similar to existing automated podcasts, because the chosen content is read without being adapted for the audio format.

QA Best. We generate the section following the procedure described in Section 7.5. Specifically, questions are automatically generated, and the selection and ordering of questions follows the algorithm illustrated in Figure 7.6.

QA Rand. Similar to QA Best, questions are automatically generated; however, the selection of questions as well as the answering paragraph are randomized (line 6 and 8 in Figure 7.6 are replaced by uniform random sampling).

Reference. One of the authors of the paper, with a background in journalism, wrote podcast sections in the Q&A format. The process involved reading one or more of the source articles, and a combination of copying content and writing novel sentences (e.g., the questions).

Comparing these four settings, we aim to answer the following three research questions:

- **RQ1**: Is the Q&A format adequate for news domain podcasting, compared to top-down article reading? (Baseline vs. Reference)

- **RQ2**: How do automated Q&A Sessions compare to hand-written Q&A Sessions? (QA Best vs. Reference)

- **RQ3**: How effective is the ordering algorithm in Figure 7.6 at creating a relevant and coherent Q&A Session? (QA Best vs. QA Rand)

Study A Specifics

Section Name	#Source	#Part.	♣ Baseline		♢ QA Rand			♡ QA Best			♠ Reference		
			$\star$I	$\star$C	#QA	$\star$I	$\star$C	#QA	$\star$I	$\star$C	#QA	$\star$I	$\star$C
UK Uber Driver Benefits	7	44	3.1	3.8	5	2.8	3.3	4	3.7	4.0	4	4.1	4.7
Emergency Broadband Subsidies	4	46	3.3	4.2	4	3.6	3.8	4	3.8	4.6	3	4.3	4.5
Greek / Turkish Dispute	2	33	3.4	4.1	4	2.8	2.8	3	3.3	3.4	4	3.4	4.4
Instagram Eating Disorders	3	46	3.6	4.2	3	2.5	3.0	3	3.3	3.9	4	3.8	4.6
Jeep Cherokee Name	5	39	3.7	4.4	3	3.3	3.3	3	3.3	4.3	3	4.1	4.7
Khashoggi Murder	18	40	4.3	4.3	3	3.6	3.7	2	4.2	4.3	2	4.3	4.3
Large Iceberg Break-off	3	54	3.6	3.9	5	3.5	3.2	4	3.7	4.5	4	4.2	4.2
Average	6.0	43.6	3.55	4.10	3.85	3.19	3.32	3.14	3.60	4.15	3.42	4.03	4.48
Statistical Significance ($p < 0.05$)						♢			♢	♢		♣♢♡	♣♢♡
Would use for news? (%)			63%		47%			80%			87%		

Table 7.1: Detail of the seven podcast sections of Usability Study A. Participants selected 5 preferred sections (out of 7 possible), and were randomly assigned to a setting: Baseline, QA Rand, QA Best, or Reference. *#Source* is the number of source news articles in the section, *#Part.* the number of participants that selected each section, $\star$*I* the average interestingness rating, $\star$*C* the average coherence rating, and *#QA* the number of question/answers spoken in the section (Baseline did not have Q&A). Methods are each assigned a symbol used to indicate a statistically significant difference ($p < 0.05$) with other methods.

We recruited participants through Amazon Mechanical Turk, a crowd-sourcing platform. Crowd-workers participated from their computer, which represents a different setting than an audio-only setting such as listening to a podcast while driving or cooking. This allows us to first evaluate the system with participants whose main focus is on the podcast content, and we leave evaluating the podcast in other settings to future work.

We designed the study to run over fifteen minutes, with the following procedure:

1. **Introduction**: Participants were introduced to the task, gave their consent to participate, and viewed slides outlining the capabilities of the podcast interface (i.e. how to pause, open the transcript, etc.).

2. **Sound Check**: Upon clicking on a button, an audio message would be generated: "Click on button C". The final letter was randomized, and the user had to click on the correct letter's corresponding button (out of four). This was intended to check that participants had sound enabled.

3. **Section Selection**: Participants selected 5 sections out of a total of 7 available sections. The available choices are detailed in Table 7.1. This choice was intended to allow participants to customize podcast content, and reduce the chances they had to listen to topics they were not interested in.

4. **Podcast Listening**: The participant was randomly assigned to one of the four settings for the entire podcast. Without pausing, the podcast had an expected duration of 7 and 8 minutes. After each section, participants had to complete a short section satisfaction form, detailed below.

5. **Post-completion questionnaire**: participants completed the study with a satisfaction questionnaire.

Importantly, in this study's setting, we hid the microphone button in the interface, preventing participants from interrupting with their own questions. This was intended to focus this study on the podcast content itself. The subsequent study described in Section 7.7 focuses on listener interactions.

We created the sections based on news articles from February 27th and 28th, and participants completed the task between February 28th and March 3rd, within five days of the publication of the source news articles. We hand-selected the topics of the seven sections that participants chose from, creating a diverse set ranging from science (e.g., Large Iceberg Break-Off) to international affairs (e.g., Greek / Turkish Dispute). For the study, we specifically avoided sections that were predominantly political.

There were three main sources of signal we recorded:

1. **Interaction Log**: we recorded whether the participant paused the podcast, skipped a section and opened the transcript.

2. **Section Satisfaction Form**: upon completing each section, participants answered three questions: Did you know about this section already? (yes/no) How interesting was the section (1 to 5 stars)? How coherent was the section? (1 to 5 stars)

3. **Post-Completion Questionnaire** consisting of two parts: (1) a binary question: "Would you use this system to get updated on the news?" (yes/no), and (2) an optional free text feedback on the overall experience.

We used Amazon Mechanical Turk to recruit participants, restricting the task to workers in English-speaking countries having previously completed 1600 tasks (HITs) and an acceptance rate of at least 97%. Each participant was paid a flat rate of $2.50 with the study lasting a maximum of 15 minutes.

The following instructions were given to all participants. Participants were expected to accept the terms before moving on to the study:

- The entire HIT should take no more than 15 minutes:

- You must use Google Chrome or Mozilla Firefox for this task.

- You will go through a tutorial about the interface.

- Listen to a news podcast for 7 minutes. You must have sound enabled to hear the podcast.

- You can leave at any point but will not complete the HIT.

- You can complete this task at most once.

- If you have a question/problem, contact us at X@email.com

- By clicking "I Accept", you agree to our terms. Namely: we will use the data will collect for research purposes. We do not collect any personally identifiable information. You agree to performing the task to the best of your ability.

Study A Results

We ran the study with a total of 61 participants; 16 were assigned to the Baseline, and 15 each to QA Best, QA Random and Reference.

Even though section names were listed in shuffled order across participants during section selection, there were some preferences overall for some sections, with the most popular being about a Large Iceberg Break-Off and the least popular the Greek / Turkish Dispute.

With regard to the interaction log, 56% of participants opened the transcript at least once, 22% of participants paused the podcast at least once, and 5% skipped one or more sections, with no significant difference across settings.

The Section Satisfaction Form results are compiled in the Table 7.1. We measure two outcomes: average interestingness and average coherence (one to five stars each). Because we are comparing all pairs of four total settings, we performed a Kruskal-Wallis test [79] with a Dunn post-hoc test [36] for statistical significance between pairs of conditions.

The Reference outperforms all other methods significantly, both in how interesting and how coherent the podcast sections are. QA Best achieved the second-best scores in both metrics, narrowly above the Baseline, without statistical difference between the two settings. Finally, QA Random comes in last for both metrics, with a large average drop of 0.8 stars on coherence compared to QA Best, shining a light on the success of QA Best's selection and ordering of questions.

We measure correlation between *Already Know Story* and *How Interesting*, and find a positive correlation between the two of 0.24: if a listener is already aware of a particular topic, they are likely to rate it as more interesting. The correlation between *Already Know* and *How Coherent* is small (around 0.1), but the correlation between *How Interesting* and *How Coherent* is highest at 0.5: indeed, the podcast's coherence can be seen as a prerequisite for it to be interesting.

Fifty-six of the sixty-one participants opted to give general feedback via an open-ended text box. We tagged the responses into major themes:

- 30 participants expressed positive interaction with the system (e.g., *Nice work, I actually enjoyed listening to this. The female voice with questions was a nice touch. Good job.*, Participant A4 - Reference)

- 11 participants mentioned that the speech quality was problematic, (e.g., *I would like a more natural voice if possible.*, Participant A42 - Baseline), with 4 participants mentioning a voice in particular (e.g. *I liked it [...] but didnt like the guy's voice*, A16 - Baseline)

- 9 participants mentioned the interface was easy to use (e.g., *I think the system here is a very simple but easy to use interface that even less tech-savvy users can easily use to obtain the latest news.*, A13 - QA Best)

- 3 participants mentioned the generated speech was too fast (e.g., *Some of the speech was too fast to understand.*, Participant A28 - Reference), 3 participants wished they could select from more sections (e.g., *their was not to many articles to choose from.*, A38 - Reference), 3 participants mentioned the transcript was useful (e.g., *The live transcript feature worked well, as you'd expect for computer-generated speech)*, A39 - QA Rand).

With regards to the research questions we posed:

- **RQ1**: Yes, the Q&A format is adequate for news domain podcasting, with participants significantly preferring it over a baseline of reading unaltered news articles.

- **RQ2**: Automated Q&A sessions do not perform as well in terms of coherence and interestingness when compared to hand-written references, but 80% of QA Best listeners said they would use the system to listen to the news in the future.

- **RQ3**: The ordering choices based on the greedy traversal of the QA Graph make a significant difference when choosing questions compared to random sampling in terms of coherence and interestingness.

7.7 Usability Study B: Interaction

Study B Design

In the Interaction Study, we set the content for all participants to the QA Best setting, and focus on measuring what kinds of interruptions, if any, listeners of news podcasts might initiate.

Even though there is evidence that using speech recognition can ease user participation [69], there are limitations to speech input interfaces [145], particularly in the case of crowd-workers who may be in environments in which they are not willing to speak out loud. Preliminary experiments with NewsPod's microphone button visible confirmed that a vast majority of crowd-workers might
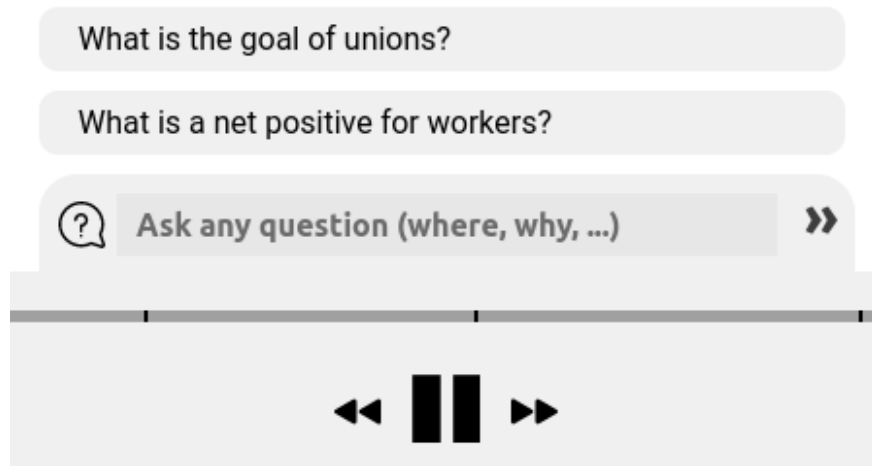
Figure 7.8: For the Interaction Study, we modify the interface, removing the microphone option (shown in Figure 7.3), allowing the participant to either type a question, or click on two recommended questions.

not want to enable their microphone for a 15-minute study. We therefore chose to replace the microphone-enabled interruptions with a text-box for the purpose of the study, allowing participants to type their questions instead of speaking them.

We also added two suggested questions to each of the podcast section. Participants could choose at any point during a podcast section to type their own question, or click on a recommended question. The graphical change to the interface is illustrated in Figure 7.8.

All participants are assigned to the QA Best section content; however, they are randomly split across two settings:

- **With Break**: At the end of each section, the following break paragraph is added: *We're wrapping up this story, if you have a question, now is a good time to ask. Otherwise, we'll be moving on to the next story.* Each of the two sentences is followed by a five second break. This adds a 15-second period dedicated to listeners being able to ask a question at the end of each section.

- **Without Break**: The podcast sections do not include any built-in breaks for participants to ask questions.

In both cases, participants can ask a question or click on a recommended question at any point in the podcast, but in the **With Break** setting, an additional period is added specifically to encourage participants to ask questions.

We aim to answer the following research questions:

- **RQ4**: Are news podcast listeners interested in interacting with the podcast by asking questions?

Section Name	#Part.	Without Break			With Break		
		Own	Rec	Any	Own	Rec	Any
Amazon Union	37	0.4	0.5	47%	0.3	0.8	60%
Bethesda Acquisition	33	0.2	0.4	41%	0.2	0.9	69%
Rohingya Crisis	20	0.4	0.2	33%	0.6	0.6	64%
Senegal Jailed Leader	15	0.0	0.5	33%	0.2	1.0	67%
Swiss Burqa Ban	23	0.2	0.4	46%	0.1	1.0	80%
Tesla Self Driving Ban	32	0.1	0.2	17%	0.2	0.7	57%
Total	160	0.9	1.5	55%	1.1	**3.2***	**85%***

Table 7.2: Results of Usability Study B on the six sections, with participants randomly assigned to Without Break or With Break settings, reporting: *#Part.* the number of participants that selected each section, *Rec* the number of recommended questions clicked, *Own* the number of own questions typed, and *Any* the percentage of participants that asked at least one question. In the total row, * indicates a statistically significant difference between settings ($p<0.05$).

- **RQ5**: Does including breaks specifically designed for listeners to ask questions help increase user interaction?

- **RQ6**: Are current automated question-answering models equipped to answer news listeners' questions?

Study B Specifics

This study was modeled on the previous one, with participants going through an introduction, a sound check, the section selection, podcast listening, and a post-completion questionnaire.

Because we ran this study two weeks after the initial study, we generated a new set of 6 QA Best podcast sections in order to maintain high content relevance for the participants. Details of the six sections options are given in Table 7.2.

In order to keep the study in the fifteen minute window while accommodating for interruptions from the listener, listeners were asked to choose four sections (out of six total) instead of five (out of seven total) as in Study A.

We recruited participants on Amazon Mechanical Turk with the same remuneration and the same selective filters on eligible crowd-workers as in Study A. Crowd-workers that had already completed Study A were ineligible for Study B.

We modified the post-completion questionnaire to focus on the interaction from the participant: for each question asked (recommended and typed), the participant was asked *How was the system's answer?*, and given three options: *I don't remember*, *Irrelevant/Confusing*, and *Good/Relevant*.

Study B Results

Participants completed a Section Satisfaction Form identical to the one described for Study A. Podcast sections in Study B were judged to be roughly as interesting and coherent as in Study A, averaging 3.9 and 4.1 stars respectively (vs. 3.6 and 4.2 in the Study A). This suggests that, even though we hand-selected the pool of sections for participants to choose from, performance of NewsPod is stable across time and does not vary highly with news topics.

Table 7.2 summarizes statistics on questions asked by participants, both clicked recommendations and typed questions. We find that in the condition without breaks, roughly half the participants asked a question, while in the condition with breaks, the proportion increases to 85%.

Including breaks also significantly increases the average number of recommended questions clicked (from 1.5 to 3.2) and in a non-significant way the number of questions typed (0.9 to 1.1).

RQ4, RQ5: When constructing the podcast with breaks, a large majority (85%) of podcast listeners asked at least one question throughout the podcast.

Next, we report on participant satisfaction of the quality of answers for selected questions. For those questions that were generated by the system, participants were generally satisfied with answers, with 79% of the answers rated *Good/Relevant*.

RQ6: On the other hand, participants were mainly dissatisfied with the automatically generated answers to their own typed questions: 76% of the answers were rated *Irrelevant/Confusing*. This result reveals the limitation of applying an extractive question answering system to open question answering.

To understand further what types of questions participants formulated, and why the live Q&A system might be inadequate, we analyzed and categorized the 57 typed questions:

- **Factoid Questions** make up 36% of questions, asking for a specific detail that can be extracted from one of the source articles. (e.g., *what percent of amazon workers belong to a union?*, Participant B18 - Without Break)

- **Synthesis Questions** make up 18% of questions and require compiling and summarizing several elements to answer (e.g., *how have people reacted to the ban?*, B6 - Without Break)

- **Encyclopedic Questions** make up 14% of questions and ask for information most likely present in an encyclopedia such as Wikipedia, or in a knowledge base (e.g., *What type of government does Senegal have?*, B23 - With Break)

- **Clarification Questions** make up 13% of the questions, asking to define a term in the podcast (e.g. *Who is Bethesda, again?* B24 - Without Break)

- **Prediction Questions** make up 9% of questions, asking about hypothetical future events (e.g., *When will Tesla release a new edition?* B12 - With Break)

- **Rhetorical Questions** make up 5% of questions, making a statement not requiring an answer (e.g., *What about driving safety?* B14 - With Break)

- **Self Relevance questions** make up 5% of questions, relating the event to the listener (e.g. *How can I join in this union?*, B36 - With Break)

The QA System we operate is mainly equipped to deal with **Factoid Questions**, as they represent the vast majority of questions present in QA datasets used to train the model. This mismatch between participant expectations and system capabilities leads to the dissatisfaction with most answers to typed questions.

Further analysis reveals that the inadequacy of the QA responses lead to a reduction in participant interaction with the system. Even though the model is able to satisfactorily answer comprehension questions 79% of the time, participants judged that the last question they asked had a satisfying answer only 57% of the time. We interpret this as evidence that inadequate responses from the QA system can discourage participants from asking further questions.

7.8 Discussion

Overall, news podcasts in the conversational format were largely enjoyed, with almost 90% of participants noting they would use our reference system again, compared to 63% for a podcast based on reading unmodified news articles. Currently, automating the podcast by using neural network-based methods to generate summaries, questions and ordering the content causes a loss in quality when compared to manual curation; however, most listeners still enjoy the podcast, with 80% of *QA Best* participants stating they would use the podcast again. Continual improvements in NLP technology will gradually reduce this gap, facilitating the construction of automated audio content for the news and other domains.

Listeners of podcasts are interested in interacting with audio content, with a majority of listeners interrupting the podcast to ask a question when the option is available. Including appropriate silences and invitations to participate in the podcast script is essential to increasing interaction, as in our study this increased the proportion of participants asking questions from 55% to 85%. Though powerful, current question answering systems are still too limited and specific to deal with the diversity of questions posed by listeners, and further engineering is required to enable satisfying interactions with the listener.

The usability studies of our system illustrate the potential of audio interfaces to increase in-depth news engagement: in a few minutes and potentially while performing another activity, a listener can obtain detailed information about topics of their choice, from a potentially diverse set of sources. At any point, the listener can join the conversation and ask questions, clarifying unclear aspects, and steering the discussion dynamically.

7.9 Limitations

At the desk setting. In our current usability studies, our participants were predominantly using laptop computers, most likely at a desk. A desk environment is advantageous, as it provides a better sound system, but is not representative of the large portion of environments for podcast listening

[107]. Further studies of the usability of NewsPod in the car, or while walking, cooking, or doing other everyday activities could shine light on other advantages and limitations of the system.

Amazon Mechanical Turk population bias. Even though recruiting through a crowd-sourcing platform typically leads to more representative participants than on-site recruitment (e.g. undergraduate students at our university) [9], the crowd-sourced population differs from the U.S. population on many metrics (e.g., age, gender, income level) [65] which would likely have an impact on our results.

Imperfect Speech and Q&A systems. Recent work [17] has shown that modern text-to-speech has approached, but not achieved, recorded human speech in quality, and around 15% of our participants stated in their open feedback that low speech quality was noticeable. Similarly, our Q&A system was not able to provide satisfying answers to a majority of the participants' questions. These imperfect components most certainly had a negative impact on participants' evaluation of the system; however, we did not measure the extent of these effect.

7.10   Future Work

Adapting to user interactions. When a listener asks a specific question, the systems attempts to answer, and then immediately resumes the previous planned section. However, it is likely that an interaction should lead to changes later on in the podcast, as it might have answered questions further down in the section, or should lead to follow-ups that would interest the listener more.

Customizing to repeat listeners. A news podcast is an opportunity for a conversational interaction with the listener. The ability to remember prior podcasts, and integrating them seamlessly into the generation of future podcasts could lead to exciting interactions: "*Hey Robin, we talked about Brexit last Wednesday, and here's the update since then...*".

Including Sound and Music. Studies have shown the effectiveness of musical communication in cinema [92], and more recent work [149] has shown that adding music and sounds to audio-books increases transportation and fear effects in listeners, both for human-recorded and automated audio-books. For automated news podcasts, music and sounds could be effective communication tools, and help soften perceptibility of automated speech flaws.

7.11   Conclusion

This chapter introduced NewsPod, an interactive and automated news podcast. NewsPod organizes news content into a simulated conversation in which several automated voices ask and answer questions about a topic. The listener can at any point join the conversation by asking a question, which the system attempts to answer automatically. We evaluated the Q&A narrative style in NewsPod through a usability study and found it led to participants finding the content more enjoyable and less monotonous, with more than 80% of the participants saying they would use the system again to stay informed on the news. A second usability study, centered on participant interactions, reveals that when appropriate breaks are included in the podcast, 85% of listeners ask

questions and interact with the podcast. Participant-prompted questions are found to be diverse, with a majority tripping up current neural network-based question-answering models, reflecting current limitations in natural language processing. This alternative approach to news consumption may eventually serve to broaden the base of users who engage deeply with the news.

Chapter 8

Conclusion

In this thesis, I propose new unsupervised methods to approach problems such as summarization and simplification, circumventing the large data requirements of prior work. I then turn my attention to the use of NLP technology in the practical setting of building news reading interfaces. I first reflect over the contributions so far and then discuss possible directions for future work.

8.1   Reflection

In this section, I reflect on the work completed during the last five years. I go over some of the challenges and the high-level lessons learned.

1. **Approaching text generation tasks such as summarization and simplification in an unsupervised way is possible.** By explicitly defining attributes of a desired generation, models can be trained with reinforcement learning to maximize such attributes. Unsupervised learning is desirable in situations where data is unavailable, but it can also be valuable in cases when data is limited, as we show in Section 2.5 of the Summary Loop chapter. One caveat is that text generators trained with reinforcement learning tend to find shortcuts for high-reward solutions; much of the recipe to success of the Summary Loop and Keep it Simple are in the iterative debugging and improvement of the reward components. One example of such minor improvements is the reward guardrails defined to avoid degenerate solutions, forcing the text generator to stay in an acceptable domain of solutions. I particularly enjoyed training text generators, and learning to out-maneuver the undesirable solutions found by the generators, sometimes being surprised and even amused at the generator's "creativity".

2. **As NLP models become more general, transfer learning becomes within reach, but requires careful attention and tuning.** Many NLP tasks, such as natural language inference, question answering or summarization are framed as very general tasks. Early in the Ph.D. however, it was often the case that models achieving high performance in one domain would see sharp degradation when applied to another domain. This gradually changed, for example with the famously adaptable BERT [30] model. There are many examples of models becoming

more general, for example in summarization with the PEGASUS model [175] setting a new state-of-the-art in ten textual domains. In Chapter 4, I fulfill my objective of using out-of-the-box NLI models for a practical application. Re-purposing models is particularly exciting, as we show in Table 4.5 that progress in NLI leads to progress in Inconsistency Detection. In another project not included in the thesis, I show with my co-authors that a text generator trained to generate news headlines can be re-used in related tasks, such as headline grouping [81].

3. **Evaluating automated text generation models is immensely challenging, and a rapidly evolving topic.** In 2016, most summarization work limited evaluation to ROUGE scores [91], and an occasional manual inspection. Soon after, direct crowd-sourced human evaluation became common, for example measuring summary relevance, fluency, coherence and consistency [42]. Seeing first-hand how these ratings can have large variance, and highly depend on the quality of annotation of the crowd-workers, I got interested in an alternate human evaluation paradigm based on the completion of a realistic tasks using the generated text. The idea is that the quality of the generated text is tied to the ability of completing a task better, faster, or with more ease for the participant. In Chapter 3.3, I successfully built such an evaluation for text simplification, after several iterations on the protocol with my co-authors. This is a step forward in task-based NLP evaluation, but much more remains to be done.

4. **Complex visualizations can be daunting, and simple text-centric interfaces relying on natural language for interaction are powerful.** Early in the Ph.D., I built several interfaces that contained rich visualizations, which were information dense, and could help a potential reader get in-depth information at a glance. I realized that for the news domain, rich visualizations can be limiting, as they require training for proper use, and might not interest more casual news readers. I became interested in building simple interfaces with a limited density of information, and that can assist a person reading news through natural language queries. I find that natural language interfaces allow an interface to be simple and yet interactive, and often two readers learning about the same news story will quickly diverge and be exposed to distinct content based on respective interest. As of now, natural language interfaces remain limited in the queries they can handle, and often do not have appropriate error handling, limitations that will hopefully be alleviated as NLP technology matures further.

8.2 Future Work

There are several directions that are natural extensions to the work presented in the thesis. I outline two ideas of interest in unsupervised text generation, and two ideas focused on improving human evaluation of NLP-based models and interfaces.

Future Work in Unsupervised Text Generation

Towards Consistent Text Generation. The models we obtain by training with the Summary Loop and Keep it Simple procedures are affected by a common limitation in most current models for text generation: they are not factually consistent. In SummaC, we take a first step in solving the problem, by building a model that can detect the generated inconsistencies with high-precision. However, detection of a problem is only a first step, and in order to close the loop and solve the problem, the detection model can be included in the generation and/or training procedures to deter inconsistent generation. I conducted basic experiments in this promising direction by incorporating the SummaC model as a reward component of the KiS procedure to train a model. Trained models tend to achieve a much improved consistency score (from around 0.25 on average to around 0.8 for a score ranging in $[0, 1]$), but inspection reveals this increase in score comes at the expense of abstractiveness, with the text generator taking fewer risks, and not producing edits spanning full sentences. This is a known limitation in text generation: abstractiveness so far comes at the cost of factual consistency. The path towards truly abstractive and consistent text generators therefore remains open.

Beyond the News Domain for Text Generation. Text generations experiments described in this thesis were conducted solely on English news data. This deliberate choice was made for several reasons. First, news is the most developed domain in summarization, due to the existence of large datasets (CNN/DM, Newsroom), which means that aligning to this domain would facilitate comparison with prior work. Second, I have a particular interest in the news domain, and aligning the domains of the text generation portion of the thesis and the interfaces we built created possibilities for synergies: for example some Summary Loop generated summaries were included in the NewsChat user-study interfaces. Nevertheless, the main appeal of the unsupervised methods I describe is that they can be applied to a domain and language without the presence of aligned data. Exploring the performance of the Summary Loop and Keep it Simple on novel domains deserves attention. Examples of particular interest are to apply the Summary Loop to a new language (e.g., Estonian summarization), or applying Keep it Simple to legal text (such as legislative bills). Initial experiments in both reveal that applying the reward functions from the English news setting achieve limited success, and adaptation of the rewards is necessary.

Future Work Human Evaluation and NLP

Task-based Human Evaluation of Text Generation Models. NLP evaluation is still mostly performed through automatic, n-gram overlap based metrics, such as ROUGE, BLEU or SARI. There are known limitations to the metrics, yet they remain ubiquitous (including in our Summary Loop and Keep it Simple chapters). A proposed replacement has been direct human evaluation, by asking crowd-workers to evaluate generated texts with a Likert scale along several dimensions (for instance asses the fluency, relevance and coherence of a summary on a 5-point Likert scale). However, in Clark et al. [25]'s words, "all that's human is not gold", and obtaining high-quality labels from an untrained, heterogeneous crowd is challenging. In Chapter 3, we take the first steps towards a task-based human evaluation for text simplification: crowd participants are given a text

comprehension task to complete, and we measure whether simplified texts lead to a statistically significant speed-up in the completion of the task. After several iterations on the task, we succeeded in finding signal, in that most generated text from simplified models lead to a speedup, with varying effects depending on the model. By directly evaluating whether the generated text can help achieve a task objective in a concrete setting, we can improve the credibility of model performance. Expanding this work to other text generation tasks, such as summarization or question generation, and defining clear, repeatable task protocols merits attention as a viable avenue for text generation evaluation.

Going longitudinal in human evaluation of NLP-powered interfaces. Most of the human evaluation performed in this thesis were short term, often packaged as a 20 minute task or less. This choice enables us to lower the cost paid to each participant, in turn allowing us to increase the participant population size, in order to reduce the variance in results. However this limits the strength of results, as most participants experience a novelty effect when they are exposed to a system. Performing a longitudinal study with the interfaces, for example over a period of two weeks, with required daily use of the interface, might surface unexpected trends, and reveal whether some of the advanced NLP-based functionalities remain in use beyond the first few interactions.

Bibliography

[1]   Amr Ahmed et al. "Unified analysis of streaming news". In: *Proceedings of the 20th international conference on World wide web*. ACM. 2011, pp. 267–276.

[2]   Oliver Alonzo et al. "Comparison of Methods for Evaluating Complexity of Simplified Texts among Deaf and Hard-of-Hearing Adults at Different Literacy Levels". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 2021, pp. 1–12.

[3]   Mandya Angrosh, Tadashi Nomoto, and Advaith Siddharthan. "Lexico-syntactic text simplification and compression with typed dependencies". In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, Aug. 2014, pp. 1996–2006. URL: https://aclanthology.org/C14-1188.

[4]   Kristjan Arumae and Fei Liu. "Guiding Extractive Summarization with Question-Answering Rewards". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 2566–2577.

[5]   Kristjan Arumae and Fei Liu. "Reinforced Extractive Summarization with Question-Focused Rewards". In: *Proceedings of ACL 2018, Student Research Workshop*. 2018, pp. 105–111.

[6]   Federico Barrios et al. "Variations of the Similarity Function of TextRank for Automated Summarization". In: *Argentine Symposium on Artificial Intelligence (ASAI 2015)-JAIIO 44 (Rosario, 2015)*. 2015.

[7]   Michael Barthel et al. "Measuring News Consumption in a Digital Era". In: *Pew Research Center's Journalism Project* (2020). URL: https://www.journalism.org/2020/12/08/measuring-news-consumption-in-a-digital-era/.

[8]   Lucas Bechberger et al. "Personalized news event retrieval for small talk in social dialog systems". In: *Speech Communication; 12. ITG Symposium*. VDE. 2016, pp. 1–5.

[9]   Adam J. Berinsky, G. Huber, and G. Lenz. "Evaluating Online Labor Markets for Experimental Research: Amazon.com's Mechanical Turk". In: *Political Analysis* 20 (2012), pp. 351–368.

[10] Vincent D Blondel et al. "Fast unfolding of communities in large networks". In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008.

[11] Samuel Bowman et al. "A large annotated corpus for learning natural language inference". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015, pp. 632–642.

[12] Andrew P Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms". In: *Pattern recognition* 30.7 (1997), pp. 1145–1159.

[13] H. Breland. "Word Frequency and Word Difficulty: A Comparison of Counts in Four Corpora". In: *Psychological Science* 7 (1996), pp. 96–99.

[14] Kay Henning Brodersen et al. "The balanced accuracy and its posterior distribution". In: *2010 20th international conference on pattern recognition*. IEEE. 2010, pp. 3121–3124.

[15] Matthew Brooks. *Future Content Experiences: The First Steps For Object-Based Broadcasting*. 2019. URL: `https://www.bbc.co.uk/rd/blog/2015-03-future-content-experiences-the-first-steps-for-object-based-broadcasting`.

[16] M. Brysbaert and B. New. "Moving beyond Kucera and Francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American English". In: *Behavior Research Methods* 41 (2009), pp. 977–990.

[17] Julia Cambre et al. "Choice of Voices: A Large-Scale Evaluation of Text-to-Speech Voice Quality for Long-Form Content". In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (2020).

[18] Meng Cao et al. "Factual Error Correction for Abstractive Summarization Models". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 6251–6258.

[19] Alberto Caprara, Matteo Fischetti, and Paolo Toth. "A heuristic method for the set covering problem". In: *Operations research* 47.5 (1999), pp. 730–743.

[20] Asli Celikyilmaz et al. "Deep Communicating Agents for Abstractive Summarization". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 2018, pp. 1662–1675.

[21] Danqi Chen et al. "Reading Wikipedia to Answer Open-Domain Questions". In: *ACL*. 2017.

[22] Yen-Chun Chen and Mohit Bansal. "Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 675–686.

[23] Zewen Chi et al. "Cross-Lingual Natural Language Generation via Pre-Training". In: *arXiv preprint arXiv:1909.10481* (2019).

[24] Eunsol Choi et al. "QuAC: Question Answering in Context". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 2174–2184.

[25] Elizabeth Clark et al. "All That's 'Human' Is Not Gold: Evaluating Human Evaluation of Generated Text". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 7282–7296. DOI: 10.18653/v1/2021.acl-long.565. URL: https://aclanthology.org/2021.acl-long.565.

[26] Kevin Clark et al. "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators". In: *International Conference on Learning Representations*. 2019.

[27] Arman Cohan et al. "A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents". In: *NAACL-HLT (Short Papers)*. 2018, pp. 615–621.

[28] Edgar Dale and Jeanne S Chall. "A formula for predicting readability: Instructions". In: *Educational research bulletin* (1948), pp. 37–54.

[29] Richard Desjardins et al. "OECD skills outlook 2013: First results from the survey of adult skills". In: *Journal of Applied Econometrics* 30.7 (2013), pp. 1144–1168.

[30] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 4171–4186.

[31] Yue Dong et al. "EditNTS: An Neural Programmer-Interpreter Model for Sentence Simplification through Explicit Editing". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 3393–3402.

[32] Yue Dong et al. "Multi-Fact Correction in Abstractive Text Summarization". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 9320–9331.

[33] Wenwen Dou et al. "Leadline: Interactive visual analysis of text data through event identification and exploration". In: *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*. IEEE. 2012, pp. 93–102.

[34] Xinya Du, Junru Shao, and Claire Cardie. "Learning to Ask: Neural Question Generation for Reading Comprehension". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2017, pp. 1342–1352.

[35] M. Dubiel, Alessandra Cervone, and G. Riccardi. "Inquisitive mind: a conversational news companion". In: *Proceedings of the 1st International Conference on Conversational User Interfaces* (2019).

[36] Olive Jean Dunn. "Multiple comparisons using rank sums". In: *Technometrics* 6.3 (1964), pp. 241–252.

[37]   Esin Durmus, He He, and Mona Diab. "FEQA: A Question Answering Evaluation Framework for Faithfulness Assessment in Abstractive Summarization". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 5055–5070.

[38]   Ullrich KH Ecker et al. "The effects of subtle misinformation in news headlines." In: *Journal of experimental psychology: applied* 20.4 (2014), p. 323.

[39]   Sergey Edunov, Alexei Baevski, and Michael Auli. "Pre-trained language model representations for language generation". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 4052–4059.

[40]   Oren Etzioni et al. "Open information extraction from the web". In: *Communications of the ACM* 51.12 (2008), pp. 68–74.

[41]   Matan Eyal, Tal Baumel, and Michael Elhadad. "Question Answering as an Automatic Evaluation Metric for News Article Summarization". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 3938–3948.

[42]   Alexander R Fabbri et al. "Summeval: Re-evaluating summarization evaluation". In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 391–409.

[43]   Tobias Falke et al. "Ranking generated summaries by correctness: An interesting but challenging application for natural language inference". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 2214–2220.

[44]   Maria Federico and M. Furini. "Enhancing learning accessibility through fully automatic captioning". In: *W4A*. 2012.

[45]   Daniel Ferrés, Montserrat Marimon, Horacio Saggion, et al. "YATS: yet another text simplifier". In: *International Conference on Applications of Natural Language to Information Systems*. Springer. 2016, pp. 335–342.

[46]   Ben Frain. *Responsive web design with HTML5 and CSS3*. Packt Publishing Ltd, 2012.

[47]   Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. "Bottom-Up Abstractive Summarization". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 4098–4109.

[48]   Dan Gillick and Yang Liu. "Non-expert evaluation of summarization systems is risky". In: *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. 2010, pp. 148–151.

[49]   Paul Ginns and J. Fraser. "Personalization enhances learning anatomy terms". In: *Medical Teacher* 32 (2010), pp. 776–778.

[50]   Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. 2014.

[51] Ben Goodrich et al. "Assessing the factual accuracy of generated text". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 166–175.

[52] Tanya Goyal and Greg Durrett. "Evaluating factuality in generation with dependency-level entailment". In: *arXiv preprint arXiv:2010.05478* (2020).

[53] Max Grusky, Mor Naaman, and Yoav Artzi. "Newsroom: A Dataset of 1.3 Million Summaries with Diverse Extractive Strategies". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 2018, pp. 708–719.

[54] Min Gui et al. "Attention Optimization for Abstractive Document Summarization". 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)*. 2019, pp. 1222–1228.

[55] Robert Gunning. "The fog index after twenty years". In: *Journal of Business Communication* 6.2 (1969), pp. 3–13.

[56] Han Guo, Ramakanth Pasunuru, and Mohit Bansal. "Dynamic Multi-Level Multi-Task Learning for Sentence Simplification". In: *Proceedings of the 27th International Conference on Computational Linguistics*. 2018, pp. 462–476.

[57] Han Guo, Ramakanth Pasunuru, and Mohit Bansal. "Soft Layer-Specific Multi-Task Summarization with Entailment and Question Generation". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 687–697.

[58] Brian Hamman. *From Designing Boxes to Designing Algorithms: How Programming the News Has Evolved at the New York Times*. 2019. URL: http://cplusj.org/keynote-speaker-brian-hamman.

[59] Steven G. Harkins and Richard E. Petty. "Effects of source magnification of cognitive effort on attitudes: An information-processing view." In: *Journal of Personality and Social Psychology* 40 (1981), pp. 401–413.

[60] Steven G. Harkins and Richard E. Petty. "The Multiple Source Effect in Persuasion". In: *Personality and Social Psychology Bulletin* 7 (1981), pp. 627–635.

[61] Karl Hermann et al. "Teaching Machines to Read and Comprehend". In: *NIPS*. 2015.

[62] Matthew Honnibal et al. *spaCy: Industrial-strength Natural Language Processing in Python*. doi.org/10.5281/zenodo.1212303. 2020. DOI: 10.5281/zenodo.1212303.

[63] Dandan Huang et al. "What Have We Achieved on Text Summarization?" In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 446–469.

[64] Dandan Huang et al. "What Have We Achieved on Text Summarization?" In: *EMNLP*. 2020.

[65] Panagiotis G. Ipeirotis. "Demographics of Mechanical Turk". In: *Labor: Supply & Demand eJournal* (2010).

[66] Google News Jigsaw. *Unfiltered.news*. `http://unfiltered.news/`. Accessed October 26, 2018.

[67] M. Kang and U. Gretzel. "Effects of podcast tours on tourist experiences in a national park". In: *Tourism Management* 33 (2012), pp. 440–455.

[68] Katharina Kann, Sascha Rothe, and Katja Filippova. "Sentence-Level Fluency Evaluation: References Help, But Can Be Spared!" In: *Proceedings of the 22nd Conference on Computational Natural Language Learning*. 2018, pp. 313–323.

[69] Lewis R. Karl, Michael E. Pettey, and Ben Shneiderman. "Speech versus Mouse Commands for Word Processing: An Empirical Evaluation". In: *Int. J. Man Mach. Stud.* 39 (1993), pp. 667–687.

[70] Vladimir Karpukhin et al. "Dense Passage Retrieval for Open-Domain Question Answering". In: *EMNLP*. 2020.

[71] Makoto P. Kato et al. "Can Social Tagging Improve Web Image Search?" In: *International Conference on Web Information Systems Engineering (WISE)*. Springer, 2008, pp. 235–249.

[72] Haben Kelati. "Librarians find creative ways to serve kids when buildings are closed for browsing". In: *The Washington Post* (2020). URL: `https://www.washingtonpost.com/lifestyle/kidspost/librarians-find-creative-ways-to-serve-kids-when-buildings-are-closed-for-browsing/2020/09/22/fd6f2db4-f9b6-11ea-a275-1a2c2d36e1f1_story.html`.

[73] Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. "Abstractive summarization of Teddit posts with multi-level memory networks". In: *NAACL-HLT*. 2019.

[74] J. Peter Kincaid et al. *Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel*. Tech. rep. Naval Technical Training Command Millington TN Research Branch, 1975.

[75] Anastassia Kornilova and Vladimir Eidelman. "BillSum: A Corpus for Automatic Summarization of US Legislation". In: *Proceedings of the 2nd Workshop on New Frontiers in Summarization*. 2019, pp. 48–56.

[76] Anastassia Kornilova and Vladimir Eidelman. "BillSum: A Corpus for Automatic Summarization of US Legislation". In: *Proceedings of the 2nd Workshop on New Frontiers in Summarization, EMNLP*. 2019.

[77] Kalpesh Krishna and Mohit Iyyer. "Generating Question-Answer Hierarchies". In: *ACL*. 2019.

[78] Miloš Krstajić et al. "Story Tracker: Incremental visual text analytics of news story development". In: *Information Visualization* 12.3-4 (2013), pp. 308–323.

[79] William H Kruskal and W Allen Wallis. "Use of ranks in one-criterion variance analysis". In: *Journal of the American statistical Association* 47.260 (1952), pp. 583–621.

[80] Wojciech Kryscinski et al. "Evaluating the Factual Consistency of Abstractive Text Summarization". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 9332–9346.

[81] Philippe Laban, Lucas Bandarkar, and Marti A Hearst. "News Headline Grouping as a Challenging NLU Task". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2021, pp. 3186–3198.

[82] Philippe Laban, John Canny, and Marti A Hearst. "What's The Latest? A Question-driven News Chatbot". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 2020, pp. 380–387.

[83] Philippe Laban and Marti A Hearst. "newsLens: building and visualizing long-ranging news stories". In: *Proceedings of the Events and Stories in the News Workshop*. 2017, pp. 1–9.

[84] Philippe Laban et al. "The Summary Loop: Learning to Write Abstractive Summaries Without Examples". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, July 2020, pp. 5135–5150. DOI: 10.18653/v1/2020.acl-main.460. URL: https://www.aclweb.org/anthology/2020.acl-main.460.

[85] Philippe Laban et al. "The Summary Loop: Learning to Write Abstractive Summaries Without Examples". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. to appear. 2020.

[86] Josephine Lau, Benjamin Zimmerman, and Florian Schaub. "Alexa, are you listening? privacy perceptions, concerns and privacy-seeking behaviors with smart speakers". In: *Proceedings of the ACM on Human-Computer Interaction* 2.CSCW (2018), pp. 1–31.

[87] Kwan Min Lee and Clifford Nass. "The multiple source effect and synthesized speech: Doubly-disembodied language as a conceptual framework". In: *Human communication research* 30.2 (2004), pp. 182–207.

[88] Mike Lewis et al. "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension". In: *arXiv preprint arXiv:1910.13461* (2019).

[89] Sophie Lewis. "NASA Curiosity rover celebrates 3,000th day on Mars with stunning panorama of planet". In: *CBS News* (2021). URL: https://www.cbsnews.com/news/nasa-curiosity-rover-celebrates-3000-day-mars-panorama/.

[90] Haoran Li et al. "Ensure the correctness of the summary: Incorporate entailment knowledge into abstractive sentence summarization". In: *Proceedings of the 27th International Conference on Computational Linguistics*. 2018, pp. 1430–1441.

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

[92] Scott D. Lipscomb and David E. Tolchinsky. "The role of music communication in cinema". In: *Musical communication* (2005), pp. 383–404.

[93] Y. Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *ArXiv* abs/1907.11692 (2019).

[94] Yinhan Liu et al. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[95] Mark Lochrie et al. "Designing immersive audio experiences for news and information in the internet of things using text-to-speech objects". In: *Proceedings of the 32nd International BCS Human Computer Interaction Conference 32*. 2018, pp. 1–5.

[96] Allan MacLean et al. "Questions, options, and criteria: Elements of design space analysis". In: *Human–computer interaction* 6.3-4 (1991), pp. 201–250.

[97] Valerie R Mariana. *The Multidimensional Quality Metric (MQM) framework: A new framework for translation quality assessment*. Brigham Young University, 2014.

[98] Louis Martin et al. "Controllable Sentence Simplification". In: *Proceedings of The 12th Language Resources and Evaluation Conference*. 2020, pp. 4689–4698.

[99] R. Mayer et al. "A Personalization Effect in Multimedia Learning: Students Learn Better When Words Are in Conversational Style Rather Than Formal Style." In: *Journal of Educational Psychology* 96 (2004), pp. 389–395.

[100] Joshua Maynez et al. "On faithfulness and factuality in abstractive summarization". In: *arXiv preprint arXiv:2005.00661* (2020).

[101] Rada Mihalcea and Paul Tarau. "Textrank: Bringing order into text". In: *Proceedings of the 2004 conference on empirical methods in natural language processing*. 2004, pp. 404–411.

[102] D. Moldovan et al. "The Structure and Performance of an Open-Domain Question Answering System". In: *ACL*. 2000.

[103] R. Moreno and R. Mayer. "Engaging students in active learning: The case for personalized multimedia messages." In: *Journal of Educational Psychology* 92 (2000), pp. 724–733.

[104] Ramesh Nallapati et al. "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond". In: *CoNLL 2016* (2016), p. 280.

[105] Feng Nan et al. "Improving Factual Consistency of Abstractive Summarization via Question Answering". In: *arXiv preprint arXiv:2105.04623* (2021).

[106] Shashi Narayan, Shay B Cohen, and Mirella Lapata. "Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 1797–1807.

[107] Nic Newman. *Podcasts: Who, Why, What, and Where?* 2020. URL: `https://www.digitalnewsreport.org/survey/2019/podcasts-who-why-what-and-where/`.

[108] Yixin Nie et al. "Adversarial NLI: A new benchmark for natural language understanding". In: *arXiv preprint arXiv:1910.14599* (2019).

[109] Nikola I Nikolov and Richard HR Hahnloser. "Abstractive Document Summarization without Parallel Data". In: *arXiv preprint arXiv:1907.12951* (2019).

[110] Kent L Norman et al. "Questionnaire for user interaction satisfaction". In: *University of Maryland (Norman, 1989) Disponıével em* (1998).

[111] NPR and Edison Research. *The Smart Audio Report*. Tech. rep. National Public Media, 2020. URL: `https://www.nationalpublicmedia.com/insights/reports/smart-audio-report/`.

[112] Tim O'Keefe et al. "A sequence labelling approach to quote attribution". In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics. 2012, pp. 790–799.

[113] Aaron van den Oord et al. "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499* (2016).

[114] Artidoro Pagnoni, Vidhisha Balachandran, and Yulia Tsvetkov. "Understanding Factuality in Abstractive Summarization with FRANK: A Benchmark for Factuality Metrics". In: *NAACL*. 2021.

[115] Liangming Pan et al. "Recent Advances in Neural Question Generation". In: *ArXiv* abs/1905.08949 (2019).

[116] Kishore Papineni et al. "BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002, pp. 311–318.

[117] Ankur Parikh et al. "A Decomposable Attention Model for Natural Language Inference". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 2249–2255.

[118] Ramakanth Pasunuru and Mohit Bansal. "Multi-Reward Reinforced Summarization with Saliency and Entailment". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 2018, pp. 646–653.

[119] Romain Paulus, Caiming Xiong, and Richard Socher. "A Deep Reinforced Model for Abstractive Summarization". In: *Proceedings of ICLR*. 2018.

[120] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[121] Silvia Pfeiffer. "The Ogg Encapsulation Format Version 0". In: *RFC 3533 (Informational) (2003). URL http://www.ietf.org/rfc/rfc3533.txt*. 2003.

[122] Gevorg Poghosyan and Georgiana Ifrim. "Real time News Story Detection and Tracking with Hashtags". In: *Computing News Storylines Workshop at EMNLP 2016, Austin, Texas*. Nov. 2016.

[123] Horst Pöttker. "News and its communicative quality: The inverted pyramid—when and why did it appear?" In: *Journalism Studies* 4.4 (2003), pp. 501–511.

[124] Bruno Pouliquen, Ralf Steinberger, and Clive Best. "Automatic detection of quotations in multilingual news". In: *Proceedings of Recent Advances in Natural Language Processing*. 2007, pp. 487–492.

[125] Bruno Pouliquen, Ralf Steinberger, and Olivier Deguernel. "Story tracking: linking similar news over time and across languages". In: *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization*. Association for Computational Linguistics. 2008, pp. 49–56.

[126] Weizhen Qi et al. "ProphetNet: Predicting Future N-gram for Sequence-to-Sequence Pre-training". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. 2020, pp. 2401–2410.

[127] Jipeng Qiang et al. "Lexical simplification with pretrained encoders". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. 2020, pp. 8649–8656.

[128] Alec Radford et al. "Language models are unsupervised multitask learners". In: (2019).

[129] Pranav Rajpurkar, Robin Jia, and Percy Liang. "Know What You Don't Know: Unanswerable Questions for SQuAD". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2018, pp. 784–789.

[130] Juan Enrique Ramos. "Using TF-IDF to Determine Word Relevance in Document Queries". In: 2003.

[131] Siva Reddy, Danqi Chen, and Christopher D Manning. "Coqa: A conversational question answering challenge". In: *Transactions of the Association for Computational Linguistics* 7 (2019), pp. 249–266.

[132] Siva Reddy, Danqi Chen, and Christopher D. Manning. "CoQA: A Conversational Question Answering Challenge". In: *Transactions of the Association for Computational Linguistics* 7 (2019), pp. 249–266.

[133] Steven J Rennie et al. "Self-critical sequence training for image captioning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 7008–7024.

[134] Tom Rosenstiel et al. "How Americans get their news". In: *American Press Institute* (2014). URL: https://www.americanpressinstitute.org/publications/reports/survey-research/how-americans-get-news/.

[135] Harshita Sahijwani, Jason Ingyu Choi, and Eugene Agichtein. "Would You Like to Hear the News?: Investigating Voice-Based Suggestions for Conversational News Recommendation". In: *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval* (2020).

[136] Julian Salazar et al. "Masked Language Model Scoring". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 2699–2712.

[137]   Tal Schuster, Adam Fisch, and Regina Barzilay. "Get Your Vitamin C! Robust Fact Verification with Contrastive Evidence". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2021, pp. 624–643.

[138]   Thomas Scialom et al. "Answers Unite! Unsupervised Metrics for Reinforced Summarization Models". 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)*. 2019, pp. 3237–3247.

[139]   Thomas Scialom et al. "Questeval: Summarization asks for fact-based evaluation". In: *arXiv preprint arXiv:2103.12693* (2021).

[140]   Abigail See, Peter J Liu, and Christopher D Manning. "Get To The Point: Summarization with Pointer-Generator Networks". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1. 2017, pp. 1073–1083.

[141]   Dafna Shahaf, Carlos Guestrin, and Eric Horvitz. "Trains of thought: Generating information maps". In: *Proceedings of the 21st international conference on World Wide Web*. ACM. 2012, pp. 899–908.

[142]   Dafna Shahaf et al. "Information cartography: creating zoomable, large-scale maps of information". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2013, pp. 1097–1105.

[143]   Eva Sharma, Chen Li, and Lu Wang. "BIGPATENT: A Large-Scale Dataset for Abstractive and Coherent Summarization". In: *ACL*. 2019.

[144]   Jonathan Shen et al. "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 4779–4783.

[145]   Ben Shneiderman. "The limits of speech recognition". In: *Communication ACM* 43 (2000), pp. 63–65.

[146]   Malbert Smith et al. "The Lexile Framework for Reading: An Introduction to What It Is and How to Use It". In: 2016.

[147]   Robyn Speer et al. *LuminosoInsight / wordfreq: v2.2*. doi.org/10.5281/zenodo.1443582. Oct. 2018. DOI: `10.5281/zenodo.1443582`.

[148]   Felix Stahlberg and Shankar Kumar. "Seq2Edits: Sequence Transduction Using Span-level Edit Operations". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 5147–5159.

[149]   Sophia C. Steinhaeusser, Philipp Schaper, and Birgit Lugrin. "Comparing a Robotic Storyteller versus Audio Book with Integration of Sound Effects and Background Music". In: *Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction* (2021).

[150] Xingwu Sun et al. "Answer-focused and position-aware neural question generation". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 3930–3939.

[151] Sai Surya et al. "Unsupervised Neural Text Simplification". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 2058–2068.

[152] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[153] Russell Swan and James Allan. "Automatic generation of overview timelines". In: *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2000, pp. 49–56.

[154] Xavier Tannier and Frédéric Vernier. "Creation, Visualization and Edition of Timelines for Journalistic Use". In: *Proceedings of Natural Language meets Journalism Workshop at IJCAI 2016*. New York, USA, July 2016.

[155] Paul Taylor and Amy Isard. "SSML: A speech synthesis markup language". In: *Speech communication* 21.1-2 (1997), pp. 123–133.

[156] S Rebecca Thomas and Sven Anderson. "WordNet-based lexical simplification of a document." In: *KONVENS*. 2012, pp. 80–88.

[157] James Thorne et al. "FEVER: a Large-scale Dataset for Fact Extraction and VERification". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 2018, pp. 809–819.

[158] Adam Trischler et al. "NewsQA: A Machine Comprehension Dataset". In: *Proceedings of the 2nd Workshop on Representation Learning for NLP*. 2017, pp. 191–200.

[159] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[160] Jesse Vig et al. "SummVis: Interactive Visual Analysis of Models, Data, and Evaluation for Text Summarization". In: *arXiv preprint arXiv:2104.07605* (2021).

[161] Piek Vossen, Tommaso Caselli, and Yiota Kontzopoulou. "Storylines for structuring massive streams of news". In: *Proceedings of the First Workshop on Computing News Storylines*. 2015, pp. 40–49.

[162] Denny Vrandečić and Markus Krötzsch. "Wikidata: a free collaborative knowledgebase". In: *Communications of the ACM* 57.10 (2014), pp. 78–85.

[163] Alex Wang, Kyunghyun Cho, and Mike Lewis. "Asking and Answering Questions to Evaluate the Factual Consistency of Summaries". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 5008–5020.

[164] Wenbo Wang et al. "Concept Pointer Network for Abstractive Summarization". 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)*. 2019, pp. 3067–3076.

[165] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. "Neural network acceptability judgments". In: *Transactions of the Association for Computational Linguistics* 7 (2019), pp. 625–641.

[166] Peter West et al. "BottleSum: Unsupervised and Self-supervised Sentence Summarization using the Information Bottleneck Principle". 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)*. 2019, pp. 3743–3752.

[167] Adina Williams, Nikita Nangia, and Samuel Bowman. "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 2018, pp. 1112–1122.

[168] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[169] Sander Wubben, Antal van den Bosch, and Emiel Krahmer. "Sentence Simplification by Monolingual Machine Translation". In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2012, pp. 1015–1024.

[170] W. Xu, Chris Callison-Burch, and Courtney Napoles. "Problems in Current Text Simplification Research: New Data Can Help". In: *Transactions of the Association for Computational Linguistics* 3 (2015), pp. 283–297.

[171] Wei Xu et al. "Optimizing statistical machine translation for text simplification". In: *Transactions of the Association for Computational Linguistics* 4 (2016), pp. 401–415.

[172] Koichiro Yoshino and Tatsuya Kawahara. "Conversational system for information navigation based on POMDP with user focus tracking". In: *Computer Speech & Language* 34.1 (2015), pp. 275–291.

[173] Jingqing Zhang et al. "PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization". In: *arXiv preprint arXiv:1912.08777* (2019).

[174] Jingqing Zhang et al. "PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization". In: *ICML*. 2020.

[175] Jingqing Zhang et al. "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 11328–11339.

[176] Shiyue Zhang and Mohit Bansal. "Addressing Semantic Drift in Question Generation for Semi-Supervised Question Answering". 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)*. 2019.

[177] Xingxing Zhang and Mirella Lapata. "Sentence Simplification with Deep Reinforcement Learning". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 584–594.

[178] Yuhao Zhang et al. "Optimizing the Factual Correctness of a Summary: A Study of Summarizing Radiology Reports". In: *arXiv preprint arXiv:1911.02541* (2019).

[179] Yuhao Zhang et al. "Optimizing the Factual Correctness of a Summary: A Study of Summarizing Radiology Reports". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 5108–5120.

[180] Yuhao Zhang et al. "Optimizing the Factual Correctness of a Summary: A Study of Summarizing Radiology Reports". In: *ACL*. 2020.

[181] Zhuosheng Zhang, Junjie Yang, and Hai Zhao. "Retrospective reader for machine reading comprehension". In: 2021.

[182] Chen Zhao et al. "Transformer-XH: Multi-Evidence Reasoning with eXtra Hop Attention". In: *International Conference on Learning Representations*. 2019.

[183] Yang Zhong et al. "Discourse level factors for sentence deletion in text simplification". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 2020, pp. 9709–9716.

[184] Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. "A monolingual tree-based translation model for sentence simplification". In: *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. 2010, pp. 1353–1361.