Improving the reliability of language models for summarization

Kundan Krishna

CMU-LTI-24-007

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: summarization, pretraining, factuality of language models, human-in-the-loop techniques, AI safety

Dedicated to my family

Abstract

Abstractive summarization models have made rapid progress since neural networks were first used for the task. We advanced from a situation where models struggled to produce grammatical sentences, to large language models like ChatGPT producing fluent summaries, sometimes rated even better than some human-written summaries. The application of summarization models is expanding beyond the traditionally popular domains of news articles and meeting transcripts, into new niche domains like medical reports, financial articles, social media conversations, product reviews etc.

Despite the progress, the reliability of summarization models is called into question due to rare but catastrophic failure modes. For example, models are known to generate summaries containing statements which are factually incorrect or which are not supported by the input being summarized (called *hallucinations*). Such errors can lead to serious harm if acted upon, in high-risk applications such as healthcare and finance. When deployed in the wild, models might encounter noise in the input, which can significantly reduce summary quality. Finally, while pretraining models greatly improves the quality of its outputs, the web-sourced pretraining data can introduce negative aspects in them. Examples of it include toxic or biased outputs, and verbatim generation of copyrighted content, which has led to multiple recent lawsuits. These problems can dissuade entities from deploying summarization models in the real world.

In this thesis, we contribute methodology and resources to address the aforementioned problems in summarization models. In the first part, we propose methods to generate summaries with improved quality for inputs with challenging characteristics such as long conversations or noisy documents. We introduce a modular summary generation pipeline to handle long sequences, producing better and more factual summaries. We then characterize the impact of input noise on summarization models, and design light-weight probes to detect and remove the noise. In the second part, we introduce approaches to pretraining that shun the use of any upstream pretraining text corpora, but still deliver a large fraction of the performance gains seen by pretraining on giant web corpora. The proposed approaches include creating a pretraining corpus artificially, and re-using unlabeled text from the downstream training examples for pretraining. This part reveals that a large portion of performance gains coming from pretraining are attributable to some unknown mechanism other than knowledge transfer from large external pretraining corpora. In the third and final part, we design methods to facilitate verification of LLM-generated summaries and detect potential factual errors in it. We create a public benchmark dataset to enable training and evaluation of models for multiple tasks useful towards fact-checking summaries. We then design an interactive tool to assist users in verifying LLM-generated summaries against the reference document, and show its effectiveness at highlighting errors generated by a wide variety of LLMs for documents in diverse domains.

Acknowledgments

The work in this thesis would have been impossible to do without the help of so many people in my life. Firstly, I would like to thank my advisors. It was only through the patient and persistent guidance of my advisors, that I was able to develop as a researcher and carry out the work in this thesis. Zack taught me how to recognize experimental results that hold true significance and can lead to promising directions of research. He instilled in me a sense of healthy skepticism for ideas in research that are widely accepted without enough scrutiny. I will cherish his lessons on academic writing, which converted the process of paper writing from a boring task to one of thoughtful creation. Jeff introduced me to thinking deeply from the end-user's perspective while working on problems. In the early years of my PhD, I often ran after research problems that appeared technically *cool* without regard for their usefulness. Later I realized that finding research problems by looking at the user's perspective actually leads to more interesting research, both in term of technique and utility. Jeff helped me greatly improve my user interaction design skills via his insightful feedback, which helped in multiple works in this thesis. Above all, I would like to thank my advisors for believing in me.

I am grateful to my thesis committee members for their contribution in this work. I would like to thank Byron Wallace for his constant guidance and mentorship on our shared projects, some of which also became a part of this thesis. I would like to thank Sherry Wu and Alexander Rush for their insightful feedback which helped me improve the thesis, and especially helped me shape the latter works in it.

Besides my time at CMU, I also had the good fortune to do many internships in the industry. I was fortunate to collaborate with amazing researchers at my internships — Peter Liu, Yao Zhao and Jie Ren at Google, Yaroslav Nechaev and Bill Campbell at Amazon, Anna Von Reden and Davis Liang at Abridge. I sincerely thank them for their mentorship and support. Working with them helped me gain perspective on how to vector my research towards the needs of users and products. I also benefited immensely from learning how to code better and manage large-scale experiments at these places, and these skills continue to pay dividends to this day.

Perhaps the best feature of Carnegie Mellon University is that you get to work with a stellar crowd of intelligent, motivated and kind peers. I was fortunate to collaborate with some of them on a few projects. I would like to thank my collaborators Saurabh Garg, Prakhar Gupta, Sanjana Ramprasad and Amy Pavel. Saurabh's upbeat attitude and keen insight into experimental results were crucial in taking some works in this thesis to the finish line. Prakhar and Sanjana worked tirelessly to execute key parts of multiple projects in this thesis. I would like to thank Amy for guiding a novice me on the very first paper I wrote at CMU.

I was fortunate to always have an amazingly supportive group of friends, who carried me through good and bad times during my PhD. I would like to thank Sanket Mehta who has been my close friend and mentor almost 10 years now, as we went together through one university and two corporate jobs. I would cherish the memorable times we had at our *Chai pe Charcha* sessions. We were also each other's support system during our worst crises in the past 5 years, and I am indebted for that. I would like to thank Prakhar Gupta, for all our adventures together, and for all the useful advice that helped me steer through my PhD. I sincerely thank my buddy Ashwin Jadav. It was great fun being your roommate and doing all kinds of primal ape stuff. I would like to thank Saurabh Garg. I would fondly remember brainstorming deep research ideas while cooking together in the kitchen, and going through the suspenseful twists in our job search together. I would like to thank Jimin Sun for her cheerful support during the final stretch of my PhD, and for teaching me Korean. 많이 고마워 ! I would like to thank Divyansh Kaushik for his friendship and guidance, and for cooking amazing food for me with extra ghee. I would like to thank Danish Pruthi for his mentorship and for the long chats about academia.

I would like to thank my friends in LTI and MLD for their companionship and support throughout the years— Dheeraj Rajagopal, Chaitanya Ahuja, Simran Kaur, Shengyu Feng, Mingkai Deng, Adithya Pratapa, Aman Madaan, Vidhisha Balachandran, Simran Kaur, Jared Fernandez, So Yeon Min, Hwijeen Ahn, Sang Keun Choe, Emmy Liu, Ritam Dutt, Amanda Bertsch and Lindia Tjuatja. Because I have two advisors, I also got to be a part of two amazing labs! I would like to thank members of the ACMI Lab including Tanya Marwah, Pratyush Maini, Jake Tyo, Shantanu, Divyansh Kaushik, Danish Pruthi, Saurabh Garg, Helen Zhou, Emily Byun, Daniel Jeong, Jennifer Hsia, Leqi Liu, Simran Kaur and William Wong. I would like to express my gratitude to members of the Big lab including Amy Pavel, Ahnhong Guo, Prakhar Gupta, Stephanie Valencia, Jason Wu, Yi-Hao Peng, Jessica Huynh, Faria Huq and Yasmine Kotturi. It was humbling to be a part of such a talented group of people and it was a joy to get a close look at the cutting-edge research that they carried out.

I would say that my family has an even bigger contribution in my success than myself. My grandfather, late Ram Krishna Jayaswal had a keen interest in education. He founded the first English-teaching school in our village, and wrote grammar books for me to read. My father, Jitendra Jayaswal and my mother Niru Jayaswal made sure that I got the best education and the freedom to pursue higher education. They always cheered me on without any pressure of expectations. My elder brother Prashant Kumar always stood by me with his love, support and advice. It is difficult to put their contributions and kindness into words. They have my eternal gratitude as always for all their love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A lot of progress has been made on the task of summarization in the past few years. Early works on this task were mostly limited to highlighting the important parts of the input document [Erkan and Radev, 2004, Wong et al., 2008], termed as *extractive* summarization, instead of generating new text paraphrasing the content. This was mostly due to the difficulty of generating coherent text expressing given information. However, with sequence-to-sequence neural architectures such as the LSTM [Hochreiter and Schmidhuber, 1996] coming into popular use, such models were successfully applied to tackle *abstractive* summarization — where a summary text is generated by re-wording the input content [Rush et al., 2015, Nallapati et al., 2016, See et al., 2017]. The quality of the generated summaries improved steadily with the advent of transformer based models [Vaswani et al., 2017] which were pretrained using a variety of different techniques [Liu and Lapata, 2019, Zhang et al., 2020, Lewis et al., 2020b, Raffel et al., 2020a]. Today, as we approach very large language models (LLMs), created using massive pretraining data and also some human feedback signals, we have models like GPT-3 whose summaries are rated very highly by humans [Goyal et al., 2022], sometimes even more than human-written ones [Stiennon et al., 2020].

Simultaneously, we have witnessed an explosion in the domains to which summarization models are being applied. While early datasets were mostly in the domain of news articles [Hermann et al., 2015, Narayan et al., 2018b] and proceedings of meetings [Carletta, 2007], newer datasets explore a lot more domains. Some of these domains include chat conversations [Gliwa et al., 2019], product reviews [Khan et al., 2020, Gerani et al., 2019], medical visits [Krishna et al., 2021b], social media posts [Kim et al., 2019a], bills [Kornilova and Eidelman, 2019] etc. This reflects the growing interest in utilizing summarization models for diverse applications.

While there has been a lot of progress in research on abstractive summarization as evidenced by steadily improving performance on benchmark datasets, actually deploying them in the real-world presents a different set of challenges. For example, one such challenge is much lower tolerance for factual inaccuracies in the generated summaries, which can lead to adverse effects when people actually rely on them for decision making. Another source of difficulty in real world summarization comes from challenging characteristics of input and output sequences to be processed by the model. For example, when summarizing text in the wild, models can encounter noise in the input due to imperfect data acquisition (e.g. ASR errors, incorrect parsing of web pages) which can potentially reduce the summary quality. In some summarization tasks, the input and output sequences may be too long to process end-to-end due to memory constraints. Finally, deployment of models pretrained on a large amount of web-sourced data comes with a new set of risks. The pretraining corpora may contain toxicity and bias, which can get reflected in the model's outputs. Additionally, the web-crawled pretraining data can often contain copyrighted materials, whose use itself can be considered legally contentious, with multiple recent lawsuits challenging it. We elaborate on each of these risks below:

Generation of factually incorrect statements: Abstractive summarization models paraphrase information from the source text to generate a summary, in contrast to extractive summarization which simply *selects* important content from the source text. However, while rephrasing the source text, the model's output doesn't always stay true to the information provided in the source, leading to *factual errors* in the summary. These erroneous facts in the summary can be categorized into two types: (i) claims which contradict with some information in the source text, and (ii) claims for which there is no evidence in the source, although there isn't any direct contradiction either. The latter type of factual errors are popularly known as hallucinations [Maynez et al., 2020] since they consist of made-up information. This is arguably a much worse failure mode than the ones encountered in the extractive summarization paradigm where the worst that could happen is that some important information was not highlighted.

Factual errors can be found in summaries generated from models across different architectures, from the older LSTM models [See et al., 2017] to the more recent transformer architectures [Maynez et al., 2020], and persists even after models are pretrained [Pagnoni et al., 2021]. Even after tremendous increase in model size and pretraining scale, modern large language models (LLMs) such as GPT-3 and GPT-4 also produce factual errors in their summaries, although with a much lesser frequency than their older counterparts [Goyal et al., 2022, Bubeck et al., 2023, Metz, 2023]. Factual inconsistencies in summaries can lead to severe consequences in sensitive application areas where people rely on them for decision-making. For example, in emergency medicine, an incorrect summary of a patient's past health history (such as allergies and diseases) can lead to sub-optimal decisions by the doctors providing care. In finance, real-time news headlines generated via summarization [Bambrick et al., 2020] inform investment decisions, and the presence of incorrect information in them can lead to bad decisions.

Drop in summary quality due to challenging input characteristics: One characteristic of inputs and corresponding summaries which make them challenging for summarization models is their long length. Practically all strong summarization models have an attention component, such as the earlier LSTM-based models [Rush et al., 2015, See et al., 2017] and recent transformer-based models[Raffel et al., 2020a, Zhang et al., 2020]. The memory requirements for these models increase with the length of the sequence processed — linearly in case of LSTMs and quadratically in transformer models. Often it is not possible to fit the entire input sequence representation in the available GPU memory and it becomes necessary to truncate it (e.g. to 400 words in See et al. [2017]). Incidentally, the most popular benchmark datasets in summarization research deal with summarizing news articles [Hermann et al., 2015, Narayan et al., 2018b], where this strategy does not hurt performance. This is because these professionally written articles naturally contain the most important information at the beginning, and so even taking the first three sentences as a summary performs quite well [Grenander et al., 2019]. However, this won't work while summarizing other kinds of media, for example, in meeting transcripts important information may be discussed even at the end. Hence, to ingest the whole sequence while not exceeding memory constraints, one often has to resort to using a smaller models which are generally less capable.

Virtually all research on summarization models evaluates them on carefully curated benchmark datasets. However, when deployed in real world, they can often encounter noisy data due to text ingestion artifacts. For example, when summarizing a news article on the web, a system would almost always encounter unrelated content embedded in the page such as advertisements, recommendations, and links. During text extraction, such noise may get extracted and fed through the model. Similarly, chat summarization systems might encounter noise in the form of code, URLs, or ASCII art shared in a chatroom. Moreover, it is hard to come up with prior characterization of the kind of noise that could be encountered after the model is deployed. Due to this, it hard to preemptively come up with rules to filter all kinds of noise, or perform adversarial training using synthetic noise [Karpukhin et al., 2019, Vaibhav et al., 2019] to make models robust to them.

Harmful effects stemming from pretraining corpora: The practice of pretraining models before fine-tuning/instruction-tuning them for specific tasks such as summarization has been a monumental success for NLP [Devlin et al., 2019, Raffel et al., 2020a, Brown et al., 2020, Touvron et al., 2023, Achiam et al., 2023]. To pretrain a model, we take a large unlabeled corpus of text and train the model to predict masked out portions of the text, either via infilling [Devlin et al., 2019, Tay et al., 2022] or next-token prediction [Brown et al., 2020]. Pretraining is usually done with a huge upstream corpus with billions of tokens sourced from the internet. Unfortunately, the models can also absorb negative influences from the web-sourced data, which can affect their downstream predictions. For example, the model can learn harmful ethnic and gender biases [Ahn and Oh, 2021, Kaneko et al., 2022] which leads to biased predictions. Pretrained langauge models can generate toxic and abusive text [Gehman et al., 2020, Bender et al., 2021] even when prompted with non-toxic prompts.

The use of huge web-sourced data for pretraining has also raised legal concerns, from an intellectual property standpoint. There are multiple ongoing lawsuits [Metz, 2022, Vincent, 2023, Stempel, 2024] which allege that the use of copyrighted materials to pretrain models without taking permission from creators is not fair use. A more egregious violation of copyright can occur if the model memorizes and generates verbatim some copyrighted content in response to a user query. Indeed, academic research works have demonstrated that models can potentially generate PII or copyrighted text/images [Carlini et al., 2021, 2023]. This issue has captured public interest recently after The New York Times sued OpenAI for serving its articles almost verbatim via its ChatGPT service [Grynbaum and Mac, 2023].

1.1 Thesis Overview

In this thesis, we contribute methods and resources to address the aforementioned limitations of summarization systems, thereby increasing their reliability. In the first part, we propose methods to improve the factual correctness and overall quality of model outputs when summarizing long inputs into structured summaries. We then characterize the impact of input noise on output quality via synthetic addition of noise, and introduce a lightweight method to filter out noise and improve performance. In the second part, we introduce pretraining techniques which provide large performance gains relative to using random initialization, without using any upstream corpora. We create synthetic corpora for pretraining summarization models by using representative skills potentially useful for the task. We then show that for pretraining text encoder models, simply using unlabeled text from the target task's training set often gives comparable gains to pretraining on much larger general corpora (including Wikipedia). In the third and final part, we seek to redress the problem of factual errors in model-generated summaries. We introduce a multi-domain benchmark consisting of tasks related to fact-checking summaries such as identifying hallucinated spans and retrieving supporting evidence for summaries. We then design and release an interactive tool to assist humans in fact-checking model-generated summaries against the corresponding source and demonstrate its effectiveness in a variety of settings. We elaborate on each of these contributions below:

Part I: Generating better summaries for inputs with challenging characteristics

- Chapter 2: In this chapter, we present a method for summarization for dealing with long inputs and outputs via a modular approach. We apply our method to summarize clinical conversations into formal SOAP notes [Podder et al., 2021], and corporate meetings into their summaries. In brief, the modular approach extracts noteworthy sub-parts of the input, clusters them according to their relatedness, and summarizes each cluster into a single summary sentence, while conditioning on the sub-topic within the summary which is currently being generated. This produces better summaries with fewer number of factual errors compared to end-to-end abstractive summarization. Hence, this work shows the benefits of segmenting the summary generation process to deal with smaller contexts and topics, when long text sequences are involved. This work was published at ACL 2021 [Krishna et al., 2021b].
- Chapter 3: Almost all modern works on summarization evaluate models on carefully created datasets, but during deployment these models can sometimes encounter noisy data (e.g. ads embedded in news articles on the web). In this chapter, we study how modern abstractive summarization models perform on noisy input data and provide a way to improve this performance by detecting and filtering out the noise. Crucially, our approach does not require prior knowledge of the kind of noise present. We use out-of-distribution detection techniques on the encoder representation of the input to detect noisy spans and filter them out. Feeding such *cleaned* input to the summarization model produces a better quality summary than the noisy version, recovering a significant proportion of the performance drop caused by noise. This work was published at EMNLP(Findings) 2023 [Krishna et al., 2023c].

Part II: Realizing pretraining's benefits without upstream corpora

- Chapter 4: While pretraining on large-scale web-sourced corpora has produced huge performance gains, it can also cause the model to learn toxic language or harmful biases. In this chapter we explore ways to bypass this problem by pretraining summarization models using only synthetically created nonsense data. We find that transformer models pretrained on multiple kinds of synthetic pretraining corpora, still provide over half of the performance gains (on average across 4 datasets) provided by standard pretraining. Besides performing far better than randomly initialized transformer models, these synthetically pretrained models can also outperform LSTM based models from the pre-transformer era. In summary, this chapter shows that a significant chunk of the benefits of pretraining towards summarization is due to some unexplained mechanism which can be leveraged by creating simple synthetic pretraining corpora. This work was published at EMNLP(Findings) 2021 [Krishna et al., 2021a].

- Chapter 5: In this chapter we propose a way to pretrain text encoder models without any knowledge transfer from external pretraining corpora — by simply pretraining on the downstream training set itself (*self-pretraining*). We show that self-pretraining can provide much of the performance benefit compared to using huge upstream pretraining corpora, suggesting that transfer of knowledge from external data sources is not a necessary prerequisite. We make a broad contribution by showing the generality of this phenomenon across 14 NLP datasets dealing with 6 different types of tasks including structured prediction tasks and even commonsense inference tasks. In summary, this work shows that using standard pretraining objectives, it is possible to get more performance gains out of the exact same data that's used for finetuning, with results that can sometimes even rival pretraining on large upstream corpora such as Bookwiki [Devlin et al., 2019]. This work was published at ACL 2023 [Krishna et al., 2022].

Part III: Fixing factual errors in model-generated summaries

- Chapter 6: In this chapter, we introduce the USB Benchmark consisting of labeled datasets for a variety of tasks which help in improving and verifying the factual correctness of summaries, such as evidence extraction, factual correctness prediction and fixing factually incorrect summaries. The dataset is created from Wikipedia articles, whereby human annotators edit pseudo-summaries to make sure they are fully supported by given reference, and also annotate the evidence for each summary sentence. We demonstrate that for the fact-checking tasks considered, models trained on USB's human-labeled data perform much better than models trained on labels generated synthetically using techniques from prior work. Besides tasks for fact-checking given summaries, the benchmark also provides labeled datasets for 4 tasks related to summary generation itself, thus creating a comprehensive suite of 8 summarization-related tasks. This work was published at EMNLP(Findings) 2023 [Krishna et al., 2023b].
- Chapter 7: In scenarios where the tolerance for factual errors in summaries is extremely low, such as finance and medicine, often it is necessary to get them manually verified by a human. In this chapter, we design an interactive tool called GenAudit to assist users in checking factual correctness of LLM-generated summaries. The tool provides features such as retrieving supporting evidence for the summary, and suggesting edits to fix potentially incorrect claims in it. We designed and released both an interactive interface for

using the tool, as well as trained backend models to power the aforementioned features.[1] We show that GenAudit is able to detect factual errors in outputs generated by a wide variety of advanced LLMs such as Llama-70B [Touvron et al., 2023], Gemini-pro Team et al. [2023] and GPT-4 [Achiam et al., 2023], when summarizing documents taken from multiple domains. Finally, we present an inference algorithm to tune the precision vs recall of error detection by the fact-checking backend model.

[1] Available at `https://genaudit.org`

Chapter 2

Generating SOAP notes using modular summarization techniques

The vast majority of summarization systems are designed with a monolithic approach, where a sequence-to-sequence neural network encodes the entire input at once and then decodes the entire summary in an autoregressive fashion. Such approaches are challenged in situations where the input and output sequences are long, due to factors such as the need to contextualize parts of the input far away from each other, maintaining coherence throughout the generation of a long summary, and requirement of large GPU memory. A representative problem that presents this challenge is the generation of summaries of clinical visits. Following each patient visit, physicians draft long semi-structured clinical summaries called *SOAP notes*. The conversations between the clinician and patient can often be over 2000 words long, and the average SOAP note contains well over 20 sentences which are ordered in a fixed sequence of topics to be addressed (e.g. medications, lab results etc.) While invaluable to clinicians and researchers, creating digital SOAP notes is burdensome, contributing to physician burnout.

In this chapter, we introduce the first complete pipelines to leverage deep summarization models to generate SOAP notes based on transcripts of conversations between physicians and patients. After exploring a spectrum of methods across the extractive-abstractive spectrum, we propose CLUSTER2SENT, an algorithm that (i) extracts important utterances relevant to each summary section; (ii) clusters together related utterances; and then (iii) generates one summary sentence per cluster. CLUSTER2SENT outperforms its purely abstractive counterpart by 8 ROUGE-1 points, and produces significantly more factual and coherent sentences as assessed by expert human evaluators. We demonstrate similar benefits on the AMI dataset which consists of corporate meeting transcripts and their summaries. Our results speak to the benefits of modularizing the summary generation process into multiple smaller steps which deal with individual threads of information discussed in the input and individual topics to be covered in the output.

2.1 Introduction

Electronic health records (EHR) play a crucial role in patient care. However, populating them can take as much time as attending to patients [Sinsky et al., 2016] and constitutes a major cause of

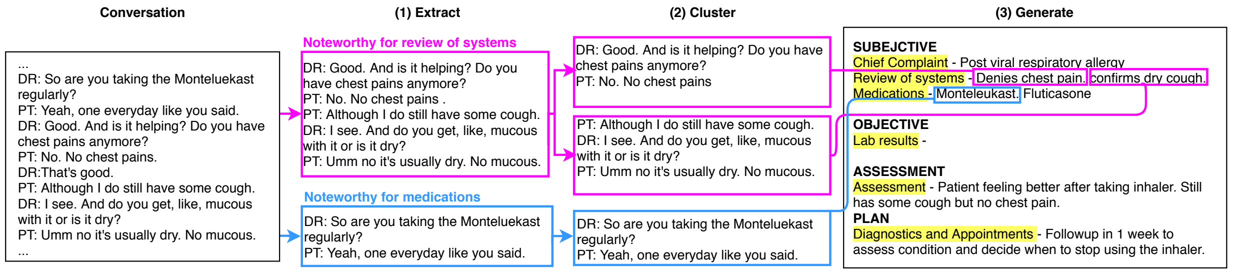

Conversation
...
DR: So are you taking the Monteleukast regularly?
PT: Yeah, one everyday like you said.
DR: Good. And is it helping? Do you have chest pains anymore?
PT: No. No chest pains.
DR:That's good.
PT: Although I do still have some cough.
DR: I see. And do you get, like, mucous with it or is it dry?
PT: Umm no it's usually dry. No mucous.
...
(1) Extract
Noteworthy for review of systems
DR: Good. And is it helping? Do you have chest pains anymore?
PT: No. No chest pains .
PT: Although I do still have some cough.
DR: I see. And do you get, like, mucous with it or is it dry?
PT: Umm no it's usually dry. No mucous.
Noteworthy for medications
DR: So are you taking the Monteleukast regularly?
PT: Yeah, one everyday like you said.
(2) Cluster
DR: Good. And is it helping? Do you have chest pains anymore?
PT: No. No chest pains
PT: Although I do still have some cough.
DR: I see. And do you get, like, mucous with it or is it dry?
PT: Umm no it's usually dry. No mucous.
DR: So are you taking the Monteleukast regularly?
PT: Yeah, one everyday like you said.
(3) Generate
SUBEJCTIVE
Chief Complaint - Post viral respiratory allergy
Review of systems - Denies chest pain. confirms dry cough.
Medications - Monteleukast. Fluticasone
OBJECTIVE
Lab results -
ASSESSMENT
Assessment - Patient feeling better after taking inhaler. Still has some cough but no chest pain.
PLAN
Diagnostics and Appointments - Followup in 1 week to assess condition and decide when to stop using the inhaler.

Figure 2.1: Workflow of our best performing approach involving extraction and clustering of noteworthy conversation utterances followed by abstractive summarization of each cluster (fictitious data)

physician burnout [Kumar and Mezoff, 2020]. In particular, doctors document patient encounters with SOAP notes, semi-structured written accounts containing four sections: (S)ubjective information reported by the patient; (O)bjective observations, e.g., lab results; (A)ssessments made by the doctor (typically, the diagnosis); and a (P)lan for future care, including diagnostic tests, medications, and treatments. Sections can be subdivided into 15 subsections.

In a parallel development, patients increasingly record their doctor's visits, either in lieu of taking notes or to share with a family member. A budding line of research has sought to leverage transcripts of these clinical conversations both to provide insights to patients and to extract structured data to be entered into EHRs [Liu et al., 2019c, Schloss and Konam, 2020, Krishna et al., 2021c].

In this paper, we introduce the first end-to-end methods for generating whole SOAP notes based on clinical conversations. Our work builds on a unique corpus, developed in collaboration with *Abridge AI, Inc.*[1]), that consists of thousands of transcripts of recorded clinical conversations together with associated SOAP notes drafted by a work force trained in the official style of SOAP note documentation. On one hand, this task is much harder than traditional summarization benchmarks, in part, because SOAP notes are longer (320 words on average) than summaries in popular datasets like CNN/Dailymail [Nallapati et al., 2016], Newsroom [Grusky et al., 2018], and SamSum [Gliwa et al., 2019] (55, 27, and 24 words on average). On the other hand, our dataset offers useful structure: (i) segmentation of each SOAP note into subsections; and (ii) a set of *supporting utterances* that provide evidence for each sentence in the SOAP note. Exploiting this structure, our methods outperform appropriate baselines.

Our first methodological contribution is to propose a spectrum of methods, for decomposing summarizaton tasks into *extractive* and *abstractive* subtasks. Starting from a straightforward sequence-to-sequence model, our methods shift progressively more work from the abstractive to the extractive component: (i) CONV2NOTE: the *extractive* module does nothing, placing the full burden of summarization on an end-to-end *abstractive* module. (ii) EXT2NOTE: the extractive module selects all utterances that are *noteworthy* (i.e., likely to be marked as supporting utterances for *at least one* SOAP note sentence), and the decoder is conditioned only on these utterances; (iii) EXT2SEC: the extractive module extracts per-subsection *noteworthy* utterances and the decoder generates each subsection, conditioned only on the corresponding utterances;

[1]http://abridge.com

(iv) CLUSTER2SENT: the extractive module not only extracts per-subsection noteworthy utterances but clusters together those likely to support the same SOAP sentence—here, the decoder produces a single sentence at a time, each conditioned upon a single cluster of utterances and a token indicating the SOAP subsection. We see consistent benefits as we move from approach (i) through (iv).

Both to demonstrate the generality of our methods and to provide a reproducible benchmark, we conduct parallel experiments on the (publicly available) AMI corpus [Carletta, 2007][2] Like our medical conversations dataset, the AMI corpus exhibits section-structured summaries and contains annotations that link summary sentences to corresponding supporting utterances. Our experiments with AMI data show the same trends, favoring pipelines that demand more from the extractive component. These results speak to the wider usefulness of our proposed approaches, EXT2SEC and CLUSTER2SENT, whenever section-structured summaries and annotated evidence utterances are available.

Our best performing model, CLUSTER2SENT (Figure 2.1), demands the most of the extractive module, requiring that it both select and group each subsection's noteworthy utterances. Interestingly, we observe that given oracle (per-subsection) noteworthy utterances, a simple proximity-based clustering heuristic leads to similar performance on SOAP note generation as we obtain when using ground-truth clusters—even though the ground truth noteworthy utterances are not always localized. Applied with predicted noteworthy utterances and clusters, this approach achieves the highest ROUGE scores and produces the most useful (factual, coherent, and non-repetitive) sentences as rated by human experts. As an additional benefit of this approach, due to the smaller lengths of the input and output sequences involved, we can feasibly train large transformer-based abstractive summarization models (e.g., T5), whose memory requirements grow quadratically with sequence length. Additionally, our approach localizes the precise utterances upon which each SOAP note sentence depends, enabling physicians to verify the correctness of each sentence and potentially to improve the draft by highlighting the correct noteworthy utterances (versus revising the text directly).

In summary, we contribute the following:

- The first pipeline for drafting entire SOAP notes from doctor-patient conversations.
- A new collection of extractive-abstractive approaches for generating long section-segmented summaries of conversations, including new methods that leverage annotations attributing summary sentences to conversation utterances.
- A rigorous quantitative evaluation of our proposed models and appropriate baselines for both the extractive and abstractive components, including sensitivity of the pipeline to simulated ASR errors.
- A detailed human study to evaluate the factuality and quality of generated SOAP notes, and qualitative error analysis.

[2]Our code and trained models for the AMI dataset: `https://github.com/acmi-lab/modular-summarization`

2.2 Related Work

Summarization is a well-studied problem in NLP [Nenkova et al., 2011]. While early works focused on simply extracting important content from a document [Erkan and Radev, 2004, Wong et al., 2008], later approaches attempted to paraphrase the content into new sentences (abstractive summarization) [Filippova, 2010, Berg-Kirkpatrick et al., 2011, Wang and Cardie, 2013]. Following the development of neural sequence models [Sutskever et al., 2014], more research focuses on neural generation of abstractive summaries [Nallapati et al., 2016, See et al., 2017, Celikyilmaz et al., 2018]. While many papers summarize news articles, others summarize conversations, in business meetings [Wang and Cardie, 2013, Zhu et al., 2020], customer service [Liu et al., 2019a], and tourist information center Yuan and Yu [2019] contexts.

In the space of two-step extractive-abstractive summarization approaches, Subramanian et al. [2019] summarize scientific papers by first extracting sentences from it and then abstractively summarizing them. Chen and Bansal [2018] extract important sentences from the input and then paraphrase each of them to generate the abstractive summary. While they assume that each summary sentence is supported by exactly one source sentence, in our medical conversations, many summary sentences synthesize content spread across multiple dialogue turns (e.g., a series of questions and answers).

Past work on abstractive summarization of medical conversations has focused on summarizing patient-nurse conversations with goals including capturing symptoms of interest [Liu et al., 2019d] and past medical history [Joshi et al., 2020]. These tasks are respectively similar to generating the *review of systems* and *past medical history* subsections of a SOAP note. In contrast, we aim to generate a full-length SOAP note containing up to 15 subsections, and propose methods to address this challenge by extracting supporting context for smaller parts and generating them independently.

2.3 Dataset

We use two different datasets in this work. The primary medical dataset, developed through a collaboration with Abridge AI, consists of doctor-patient conversations with annotated SOAP notes. Additionally, we evaluate our summarization methods on the AMI dataset [Carletta, 2007], comprised of business meeting transcripts and their summaries.

2.3.1 Medical dataset

Our work builds on a unique resource: a corpus consisting of thousands of recorded English-language clinical conversations, with associated SOAP notes created by a work force trained in SOAP note documentation standards. Our dataset consists of transcripts from real-life patient-physician visits from which sensitive information such as names have been de-identified. The full medical dataset consists of 6862 visits consisting of 2732 cardiologist visits, 2731 visits for family medicine, 989 interventional cardiologist visits, and 410 internist visits. Owing to the sensitive nature of the data, we cannot share it publicly (an occupational hazard of research on machine learning for healthcare).

For each visit, our dataset contains a human-generated transcript of the conversation. The transcript is segmented into utterances, each annotated with a timestamp and speaker ID. The average conversation lasts 9.43 minutes and consists of around 1.5k words (Appendix Figure 1). Associated with each conversation, we have a human-drafted SOAP note created by trained, professional annotators. The annotators who created the SOAP notes worked in either clinical transcription, billing, or related documentation-related departments, but were not necessarily professional medical scribes. The dataset is divided into train, validation and test splits of size 5770, 500 and 592, respectively.

Our annotated SOAP notes contain (up to) 15 subsections, each of which may contain multiple sentences. The subsections vary in length. The *Allergies* subsections is most often empty, while the *Assessment* subsection contains 5.16 sentences on average (Table 2.1). The average SOAP note contains 27.47 sentences. The different subsections also differ in the style of writing. The *Medications* subsection usually consists of bulleted names of medicines and their dosages, while the *Assessment* subsection typically contains full sentences. On average, the fraction of novel (i.e., not present in the conversation) unigrams, bigrams, and trigrams, in each SOAP note are 24.09%, 67.79% and 85.22%, respectively.

Subsection	**Mean length**
Family Medical History	0.23
Past Surgical History	0.58
Review of Systems	3.65
Chief Complaint	2.17
Miscellaneous	2.81
Allergies	0.06
Past Medical History	2.93
Social History	0.27
Medications	3.74
Immunizations	0.11
Laboratory and Imaging Results	2.27
Assessment	5.16
Diagnostics and Appointments	1.65
Prescriptions and Therapeutics	1.75
Healthcare Complaints	0.09

Table 2.1: Average number of sentences in different SOAP note subsections grouped by parent sections (*Subjective*, *Objective*, *Assessment*, *Plan*, *Others* resp.)

Each SOAP note sentence is also annotated with utterances from the conversation which provide evidence for that sentence. A SOAP note sentence can have one or more supporting utterances. On average, each SOAP sentence has 3.84 supporting utterances, but the mode is 1 (Appendix Figure 1). We refer to these utterances as *noteworthy utterances* throughout this paper. Throughout this work, we deal with the 15 more granular subsections rather than the 4 coarse sections of SOAP notes, and thus for convenience, all further mentions of *section* technically denote a SOAP *subsection*.

2.3.2 AMI dataset

The AMI dataset is a collection of 138 business meetings, each with 4 participants with various roles (e.g., marketing expert, product manager, etc.). Each meeting transcript comes with an associated abstractive summary that is divided into four sections—*abstract*, *decisions*, *actions*, and *problems*. Each conversation also has an associated extractive summary, and there are additional annotations linking the utterances in the extractive summary to sentences in the abstractive summary. For any given sentence in the abstractive summary, we refer to the linked set of utterances in the extractive summary as its *noteworthy utterances*. We note that 7.9% of the abstractive summary sentences have no annotated noteworthy utterances. To simplify the analysis, we remove these sentences from summaries in the training, validation, and test splits.

2.4 Methods

We investigate the following four decompositions of the summarization problem into extractive and abstractive phases, ordered from abstraction-heavy to extraction-heavy: CONV2NOTE takes an end-to-end approach, generating the entire SOAP note from the entire conversation in one shot. EXT2NOTE first predicts all of the noteworthy utterances in the conversation (without regard to the associated section) and then generates the entire SOAP note in one shot from only those utterances. EXT2SEC extracts noteworthy utterances, while also predicting the section(s) for which they are relevant, and then generates each SOAP section separately using only that section's predicted noteworthy utterances. CLUSTER2SENT attempts to group together the set of noteworthy utterances associated with each summary sentence. Here, we cluster separately among each set of section-specific noteworthy utterances and then generate each section one sentence at a time, conditioning each on the associated cluster of utterances.

Each of these pipelines leaves open many choices for specific models to employ for each subtask. For the abstractive modules of CONV2NOTE and EXT2NOTE, we use a pointer-generator network. The abstractive modules of EXT2SEC and CLUSTER2SENT, which require conditioning on section are modeled using conditioned pointer-generator networks (described in Section 2.5), and fine-tuned T5 models which condition on the section being generated by means of prepending it to the input. T5 models could not be used in the CONV2NOTE and EXT2NOTE settings because their high memory requirement for long inputs could not be accommodated even with 48GB of GPU memory.

For noteworthy utterance extraction, we primarily use a hierarchical LSTM model and a BERT-LSTM model as described in the next section. All models are configured to have a scalar output for binary classification in EXT2NOTE, whereas for EXT2SEC and CLUSTER2SENT, they have multi-label output separately predicting noteworthiness for each section. Note that the same utterance can be noteworthy with respect to multiple sections. We use the same trained utterance extraction models for both EXT2SEC and CLUSTER2SENT.

For the clustering module in CLUSTER2SENT, we propose a heuristic that groups together any two supporting utterances that are *close*, meaning they have less than or equal to τ utterances separating them, where τ is a hyperparameter. This process is iterated, with the clusters growing in size by merging with other singletons or clusters, until every pair of *close* utterances have the same cluster membership. The value of τ is tuned on the validation set. Since each cluster necessarily produces one sentence in the SOAP note, having too many or too few clusters can make the SOAP note too long or too short, respectively. Therefore, for any given value of the hyper-parameter τ and any given section, the prediction thresholds of the extractor are tuned on the validation set to produce approximately the same number of clusters over the entire validation set as present in the ground truth for that section. Among ground truth clusters containing multiple noteworthy utterances, 82% are contiguous. In an experiment where the heuristic is used to cluster the *oracle noteworthy utterances* for each section, and summaries are subsequently generated via the abstractive modules from CLUSTER2SENT, ROUGE-1 and ROUGE-2 metrics deteriorate by less than 1 point as compared to *oracle* clusterings (Appendix Table 3), demonstrating our heuristic's effectiveness.

2.5 Model Architectures

Pointer-Generator Network We use the pointer-generator network introduced by See et al. [2017] for CONV2NOTE and EXT2NOTE. The model is a bidirectional LSTM-based encoder-decoder model with attention. It employs a pointer mechanism to copy tokens directly from the input in addition to generating them by predicting generation probabilities for the entire vocabulary. The model also computes the weights that govern copying versus generating at each decoding timestep.

Section-conditioned Pointer-Generator Network We modify the pointer-generator network for algorithms EXT2SEC and CLUSTER2SENT, to condition on the (sub)section of the summary to be generated. The network uses a new lookup table to embed the section z into an embedding $\boldsymbol{e}^z$. The section embedding is concatenated to each input word embedding fed into the encoder. The section embedding is also appended to the inputs of the decoder LSTM in the same fashion.

T5 We use the recently released T5 model [Raffel et al., 2020a] as an abstractive module. It is an encoder-decoder model, where both encoder and decoder consist of a stack of transformer layers. The T5 model is pre-trained on 5 tasks, including summarization, translation etc. We use the pre-trained T5 model parameters and fine-tune it on our task dataset. For introducing the section-conditioning in EXT2SEC and CLUSTER2SENT, we simply add the name of the section being generated to the beginning of the input.

Hierarchical LSTM classifier(H-LSTM) In this model, we first encode each utterance u_i independently by passing its tokens through a bidirectional LSTM and mean-pooling their encoded representations to get the utterance representation $\boldsymbol{h}_i$. We pass the sequence of utterance representations $\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_n\}$ through another bidirectional LSTM to get new utterance representations which incorporate neighboring contexts. These are then passed through a sigmoid activated linear layer to predict each utterance's probability of noteworthiness with respect to each section.

BERT-LSTM classifier(B-LSTM) In this model, tokens in the utterance u_i are passed through a BERT encoder to obtain their contextualized representations, which are mean-pooled to get the utterance representation $\boldsymbol{h}_i$. The subsequent architecture exactly mirrors hierarchical LSTM, and involves passing utterance representations through a bidirectional LSTM and linear layer to get output probabilities. BERT-LSTM is fine-tuned in an end-to-end manner.

2.6 Experiments

We first establish two baselines. RANDOMNOTE randomly and uniformly samples a SOAP note from the training set and outputs it as the summary for any input conversation. ORACLEEXT presents all the ground truth noteworthy utterances (evidence) from the conversation as the SOAP note without any abstractive summarization. Thus, the ORACLEEXT baseline has the advantage of containing all the desired information (e.g., names of medicines) from the conversation, but the disadvantage of not being expressed in the linguistic style of a SOAP note which leads to lower n-gram overlap. The opposite is true for the RANDOMNOTE baseline. Both baselines give similar performance and are outperformed by the simple CONV2NOTE approach (Table 2.2).

We train the abstractive modules for the 4 approaches described in Section 2.4 with the ground truth noteworthy utterances as inputs. To estimate an upper bound on the performance we can reasonably hope to achieve by improving our noteworthy utterance extractors, we test our models with oracle noteworthy utterances in the test set. All algorithms relying on oracle noteworthy utterances outperform CONV2NOTE, and exhibit a monotonic and significant rise in ROUGE scores as we move towards the extraction-heavy end of the spectrum (Table 2.3)[3].

For predicting noteworthy utterances, we use two baselines: (i) logistic regression on TF-IDF utterance representations; and (ii) a model with a bidirectional LSTM to compute token-averaged utterance representations, followed by a linear classification layer. These two models make the predictions for each utterance independent of others. In contrast, we also use models which incorporate context from neighboring utterances: (a) a hierarchical LSTM; and (b) a BERT-LSTM model as described in Section 2.5. The latter two methods perform much better (Table 2.5), demonstrating the benefit of incorporating neighboring context, with BERT-LSTM performing the best (see Appendix Table 6 for section-wise performance).

Using predicted noteworthy utterances and clusters instead of oracle ones leads to a drop in ROUGE scores, but the performance of EXT2SEC and CLUSTER2SENT is still better than CONV2NOTE (Table 2.2). For the medical dataset, using a BERT-LSTM extractor leads to the best performance, with CLUSTER2SENT outperforming CONV2NOTE by about 8 points in ROUGE-1 (see Appendix Table 5 for section-wise performance). Interestingly, the T5-Small variant achieves similar performance to T5-Base, despite being only about a quarter of the latter's size.

Performance on AMI dataset We see a similar trend in the ROUGE scores when applying these methods on the AMI dataset. One exception is the poor performance of pointer-generator based EXT2NOTE, which excessively repeated sentences despite using a high coverage loss coefficient. There is a larger gap between the performance of the T5-Small and T5-Base abstrac-

[3]The character '-' represents GPU memory overflow

	Medical dataset			AMI corpus		
Method	**R-1**	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
RANDOMNOTE	34.99	12.69	21.37	42.47	11.55	21.47
ORACLEEXT	33.07	12.22	17.42	39.97	11.17	20.91
CONV2NOTE (PG)	49.56	25.68	32.87	39.62	13.16	23.95
EXT2NOTE (PG + HLSTM)	49.58	24.91	31.68	21.28	7.06	15.96
EXT2NOTE (PG + BLSTM)	50.50	25.4	31.93	21.71	6.83	15.69
EXT2NOTE (T5-Small + HLSTM)	-	-	-	40.48	13.82	24.64
EXT2NOTE (T5-Small + BLSTM)	-	-	-	40.36	13.73	24.13
EXT2SEC (PG + HLSTM)	55.23	27.14	35.15	43.75	15.25	23.46
EXT2SEC (PG + BLSTM)	55.74	27.54	36.09	40.48	15.61	23.31
EXT2SEC (T5-Small + HLSTM)	55.77	28.64	37.50	42.45	15.20	23.92
EXT2SEC (T5-Small + BLSTM)	56.00	29.16	**38.38**	45.44	16.59	**26.14**
CLUSTER2SENT (PG + HLSTM)	55.46	27.41	35.81	46.19	16.64	24.29
CLUSTER2SENT (PG + BLSTM)	55.60	27.68	36.29	42.31	15.92	23.51
CLUSTER2SENT (T5-Small + HLSTM)	56.88	28.63	36.78	45.10	15.06	23.52
CLUSTER2SENT (T5-Small + BLSTM)	57.14	29.11	37.43	42.38	15.36	23.9
CLUSTER2SENT (T5-Base + HLSTM)	57.27	29.10	37.38	**50.52**	17.56	24.89
CLUSTER2SENT (T5-Base + BLSTM)	**57.51**	**29.56**	38.06	45.91	**17.70**	25.24

Table 2.2: ROUGE scores achieved by different methods on the two datasets

	Medical dataset		AMI corpus	
Method	PG	T5-Small	PG	T5-Small
EXT2NOTE	52.95	-	20.44	41.10
EXT2SEC	61.00	62.37	43.32	46.85
CLUSTER2SENT	63.63	66.50	51.86	54.23

Table 2.3: ROUGE-1 achieved on test set when using the abstractive models with oracle noteworthy utterances and clusters (more results with oracle in the Appendix)

tive models on this dataset. As an extractor, the performance of BERT-LSTM is again better than HLSTM (Table 2.5), but when used in tandem with the abstractive module, ROUGE scores achieved by the overall pipeline do not always follow the same order. We also observe that the clustering heuristic does not work as well on this dataset. Specifically, tuning the thresholds of the extractive model, while fixing the clustering threshold τ gave worse results on this dataset. Tuning the thresholds independent of the clusters performed better. However, the best method still outperforms CONV2NOTE by about 11 ROUGE-1 points (Table 2.2).

Performance with ASR errors In the absence of human-generated transcripts of conversations, Automatic Speech Recognition (ASR) techniques can be used to transcribe the conversa-

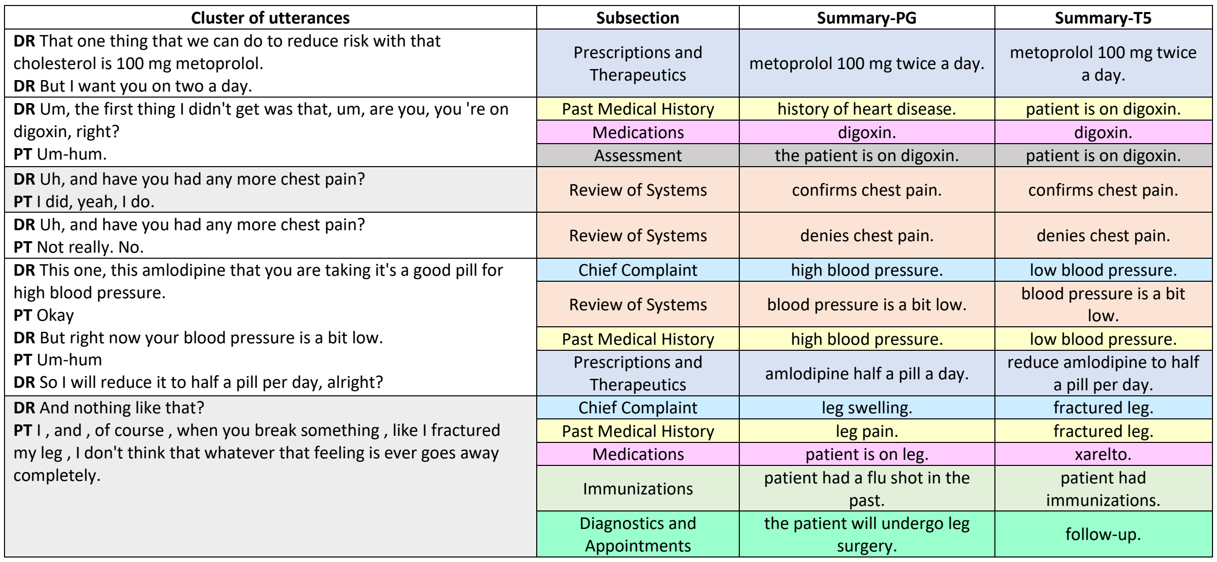
Method	**R-1**	**R-2**	**R-L**
Train on clean data + Test on data with 10% error rate			
CONV2NOTE(PG)	46.52	22.60	30.45
CLUSTER2SENT(PG + BLS)	51.84	23.74	32.94
CLUSTER2SENT(T5-Base+ BLS)	54.88	26.65	35.88
Train and test on data with 10% error rate			
CONV2NOTE(PG)	48.85	24.85	31.27
CLUSTER2SENT(PG + BLS)	54.68	26.59	35.70
CLUSTER2SENT(T5-Base+ BLS)	56.35	28.50	37.04
Train and test on data with 30% error rate			
CONV2NOTE(PG)	45.16	22.26	30.14
CLUSTER2SENT(PG + BLS)	53.69	25.88	35.12
CLUSTER2SENT(T5-Base+ BLS)	55.90	27.73	36.06

Table 2.4: Performance of models trained and tested on data with different simulated ASR error rates. BLS: BERT-LSTM

tions for use by our models. To account for ASR errors, we artificially added errors in transcripts of the medical dataset by randomly selecting some percentage of the words and replacing them with phonetically similar words using RefinedSoundEx [Commons] (details in the Appendix). Models trained on clean dataset perform worse on a 10% corrupted test dataset (Table 2.4). Since ASR errors lead to replacement of a correct word by only a small set of phonetically similar words, there is still some information indicating the original word that can be used by the models. When we train our models on data corrupted at the 10% ASR error rate, our models recover much of the performance drop (Table 2.4). Notably when simulated ASR errors are dialed up to a 30% error rate, (both at train and test time) we see a smaller performance drop for CLUSTER2SENT as compared to CONV2NOTE.

	Medical conversations				AMI corpus	
Metric	**LR**	**LS**	**HLS**	**BLS**	**HLS**	**BLS**
Accuracy	96.0	96.1	96.5	96.5	93.77	94.16
Ma-AUC	78.1	79.3	90.0	90.5	83.81	90.76
Ma-F1	29.5	31.0	38.6	40.9	19.95	33.08
Mi-AUC	87.3	87.6	92.7	93.3	93.21	94.90
Mi-F1	31.2	32.9	39.6	41.1	43.76	49.93

Table 2.5: Performance on multilabel classification of noteworthy utterances with logistic regression(LR), LSTM(LS), Hierarchical-LSTM(HLS) and BERT-LSTM(BLS). Ma:macro-averaged. Mi:micro-averaged



Cluster of utterances	Subsection	Summary-PG	Summary-T5
DR That one thing that we can do to reduce risk with that cholesterol is 100 mg metoprolol. **DR** But I want you on two a day.	Prescriptions and Therapeutics	metoprolol 100 mg twice a day.	metoprolol 100 mg twice a day.
DR Um, the first thing I didn't get was that, um, are you, you 're on digoxin, right? **PT** Um-hum.	Past Medical History	history of heart disease.	patient is on digoxin.
	Medications	digoxin.	digoxin.
	Assessment	the patient is on digoxin.	patient is on digoxin.
DR Uh, and have you had any more chest pain? **PT** I did, yeah, I do.	Review of Systems	confirms chest pain.	confirms chest pain.
DR Uh, and have you had any more chest pain? **PT** Not really. No.	Review of Systems	denies chest pain.	denies chest pain.
DR This one, this amlodipine that you are taking it's a good pill for high blood pressure. **PT** Okay **DR** But right now your blood pressure is a bit low. **PT** Um-hum **DR** So I will reduce it to half a pill per day, alright?	Chief Complaint	high blood pressure.	low blood pressure.
	Review of Systems	blood pressure is a bit low.	blood pressure is a bit low.
	Past Medical History	high blood pressure.	low blood pressure.
	Prescriptions and Therapeutics	amlodipine half a pill a day.	reduce amlodipine to half a pill per day.
DR And nothing like that? **PT** I , and , of course , when you break something , like I fractured my leg , I don't think that whatever that feeling is ever goes away completely.	Chief Complaint	leg swelling.	fractured leg.
	Past Medical History	leg pain.	fractured leg.
	Medications	patient is on leg.	xarelto.
	Immunizations	patient had a flu shot in the past.	patient had immunizations.
	Diagnostics and Appointments	the patient will undergo leg surgery.	follow-up.

Figure 2.2: Noteworthy utterance clusters summarized in different ways for different sections by the abstractive summarization modules of CLUSTER2SENT (utterances were slightly obfuscated for privacy reasons)

2.7 Qualitative Analysis

The conditioned pointer-generator and T5 models used in CLUSTER2SENT learn to place information regarding different topics in appropriate sections. Hence, given a cluster of supporting utterances, the models can generate different summaries for multiple sections (Figure 2.2). For example, given the same supporting utterances discussing the patient's usage of *lisinopril* for low blood pressure, a model generates "low blood pressure" in the *review of systems* section, and "lisinopril" in *medications* section. We direct the reader to the appendix for examples of full-length generated SOAP notes.

Interestingly, when the abstractive model is given a cluster of utterances that are not relevant to the section being generated, the model sometimes outputs fabricated information relevant to that section such as saying the patient is a non-smoker in *social history*, or that the patient has taken a flu shot in *immunizations* . Hence, the quality of produced summaries heavily depends on the ability of the extractive step to classify the extracted utterances to the correct section. Another cause of false information is the usage of pronouns in clusters without a mention of the referred entity. In such situations, T5 models frequently replace the pronoun with some arbitrary entity (e.g. "she" with "daughter", compounds with "haemoglobin", and medicines with "lisinopril").

Occasionally, the abstractive module produces new *inferred* information that is not mentioned explicitly in the conversation. In one instance, the model generated that the patient has a history of heart disease conditioned on a cluster that mentioned he/she takes *digoxin*, a popular medicine for heart disease. Similarly, the model can infer past medical history of "high cholesterol" upon seeing *pravastatin* usage. Such inferences can also lead to incorrect summaries, e.g., when a doctor explained that a patient has leaky heart valves, a model added a sentence to the *diagnostics and appointments* section saying "check valves".

CLUSTER2SENT summarizes localized regions of the conversation *independently*, which may lead to contradictions in the SOAP note. In one visit, the patient was asked about chest pain twice—once in the beginning to get to know his/her current state, and once as a question about how he/she felt just before experiencing a fall in the past. This led to the model generating both that the patient denied chest pain as well as confirmed chest pain, without clarifying that one statement was for the present and another for the past.

2.8 Human evaluation

We asked trained human annotators to evaluate generated SOAP notes for 45 conversations. Every sentence in each SOAP note was labeled according to various quality dimensions such whether it was factually correct, incoherent, irrelevant, redundant, or placed under an inappropriate section. The detailed statistics of annotations received for each quality dimension are provided in the Appendix. We also collected aggregate annotations for the comprehensiveness of each SOAP note and the extent to which it verbatim copied the transcript on a 5-point Likert scale.

Human raters were presented with a web interface showing the conversation, along with a search feature to help them in looking up desired information. The summaries generated by three methods (CONV2NOTE(pointer-generator), CLUSTER2SENT(pointer-generator) and CLUSTER2SENT(T5-base)) were presented in random order to hide their identities. For each sentence, we asked for (i) Factual correctness of the sentence; (ii) If the statement is simply repeating what has already been mentioned before; (iii) If the statement is clinically irrelevant; (iv) If the statement is incoherent (not understandable due to grammatical or semantic errors); and (v) If the statement's topic does not match the section in which it is placed. In addition, we asked two separate questions for rating the overall summary on a scale of 1-5 for its (i) comprehensiveness and (ii) extent of verbatim copying from conversation. The human evaluation of the SOAP notes was done by workers who had also participated in the creation of the dataset of SOAP notes. Hence, they had already been extensively trained in the task of SOAP note creation, which gave them appropriate knowledge to judge the SOAP notes.

To quantify the performance among different methods, we consider a scenario where each generated SOAP note has to be post-edited by discarding undesirable sentences. For a gener-

	Medical conversations			AMI corpus		
Metric	**C2N**	**C2S-P**	**C2S-T**	**C2N**	**C2S-P**	**C2S-T**
Length	21.2	28.2	28.4	20.7	17.9	19.05
%Yield	62.0	69.0	74.7	27.22	30.22	59.45
Comp	2.44	2.42	2.76	2.30	2.55	3.75
Copy	2.18	2.64	2.76	1.80	1.80	1.90

Table 2.6: Averages of different metrics for CONV2NOTE(C2N), CLUSTER2SENT with pointer-generator (C2S-P) and T5-base (C2S-T). Comp:comprehensiveness, Copy:amount of copying. Length: number of sentences generated.

ated SOAP note, we define its *yield* as the fraction of its total sentences that are not discarded. The sentences that are retained are those that are both factually correct and were not labeled as either repetitive or incoherent. The human annotations show that both CLUSTER2SENT-based methods tested produced a higher yield than the CONV2NOTE baseline ($p < 0.02$). T5-base performs better than conditioned pointer-generator as the abstractive module in CLUSTER2SENT setting, producing significantly more yield (Table 2.6). T5 also produces fewer incoherent sentences (Appendix Table 4) likely due to its exposure to a large number of well-formed coherent sentences during pretraining.

We conducted an analogous human evaluation of summaries generated for all 20 conversations in the test set of the AMI corpus, and saw a similar trend in the expected yield for different methods. Notably, for the AMI corpus, CONV2NOTE produced a very high proportion of redundant sentences (> 0.5) despite using the coverage loss, while the pointer-generator based CLUSTER2SENT produced a high proportion of incoherent sentences (Appendix Table 4).

2.9 Conclusion

This paper represents the first attempt at generating full-length SOAP notes by summarizing transcripts of doctor-patient conversations. We proposed a spectrum of extractive-abstractive summarization methods that leverage: (i) section-structured form of the SOAP notes and (ii) linked conversation utterances associated with every SOAP note sentence. The proposed methods perform better than a fully abstractive approach and standard extractive-abstractive approaches that do not take advantage of these annotations. We demonstrate the wider applicability of proposed approaches by showing similar results on the public AMI corpus which has similar annotations and structure. Our work demonstrates the benefits of creating section-structured summaries (when feasible) and collecting evidence for each summary sentence when creating any new summarization dataset.

Chapter 3

Generating better summaries for noisy inputs

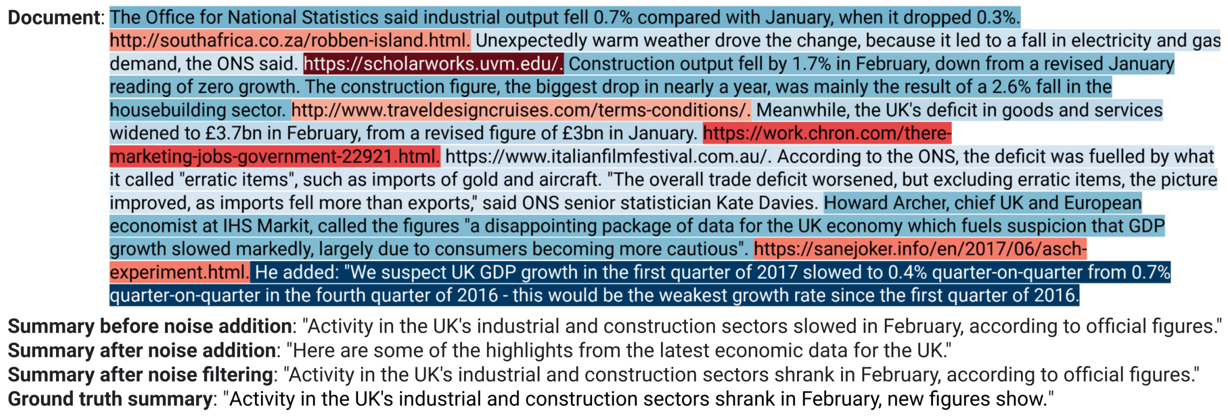
Typically, abstractive summarization models are trained and evaluated on carefully curated datasets which have minimal or no noise. In real-world practice, documents to be summarized may contain input noise caused by text extraction artifacts or data pipeline bugs. For example, in Chapter 2, the dataset of SOAP notes and accompanying conversation transcripts had transcripts created by humans, instead of automatic speech recognition (ASR) methods. In large-scale deployment, such conversations must be transcribed using ASR, and this almost certainly leads to incorrect words in the transcript. The presence of such noise in the test data leads to inferior performance by summarization models trained on clean data, as we saw via experiments simulating ASR errors in Chapter 2. However, a systematic study of the impact of various kinds of such unseen noise on summarization models trained on different kinds of datasets is not available. We also saw that training the models with simulated ASR errors in the training data itself can alleviate this problem to some extent. However, in general scenarios, a great variety of noise types can be encountered during deployment, and it is not always possible to foretell them and train models with their simulated versions. This presents a major obstacle in the deployment of summarization models in real-world scenarios.

In this chapter, we present a large empirical study quantifying the sometimes severe loss in performance (up to 12 ROUGE-1 points) from different types of input noise for a range of datasets and model sizes. We then propose a light-weight method for detecting and removing such noise in the input during model inference without requiring any extra training, auxiliary models, or even prior knowledge of the type of noise. Our proposed approach effectively mitigates the loss in performance, recovering a large fraction of the performance drop, sometimes as large as 11 ROUGE-1 points.

3.1 Introduction

Despite rapid progress in abstractive summarization in recent years [Lewis et al., 2020a, Raffel et al., 2020a, Zhang et al., 2020], virtually all works have tested models using test data which is identically distributed as the training data, and little attention has gone into studying their robustness to input distribution shift caused by input noise. Data from different domains which have been addressed in summarization research, may contain noise of different types. For example, when summarizing a news article on a web page, there can be embedded elements such as ads or tweets which may be included as part of the article due to erroneous text extraction. A system summarizing chatroom conversations might encounter artifacts such as URLs, or sometimes even code shared between participants. If the text to be summarized is acquired by scanning a document, noise can be introduced in the form of OCR errors [Jing et al., 2003]. However, the impact of different kinds of noise on modern abstractive summarization systems, and ways to accurately detect and remove that noise, remain largely unknown.

In this work, we study how noise in the input affects the output generated by summarization models, and propose a method to detect and remove it. We synthetically inject 4 types of noise to 4 abstractive summarization datasets with diverse styles [Narayan et al., 2018a, Kim et al., 2019a, Gliwa et al., 2019, See et al., 2017], and quantify the drop in aggregate metrics for the output summaries (Section 3.3). We also study how the quality of generated summaries varies with factors such as the amount of noise and size of the models. For our experiments, we use PEGASUS [Zhang et al., 2020] models — Transformer-based pre-trained models which deliver competitive performance across abstractive summarization benchmarks.



Document: The Office for National Statistics said industrial output fell 0.7% compared with January, when it dropped 0.3%. http://southafrica.co.za/robben-island.html. Unexpectedly warm weather drove the change, because it led to a fall in electricity and gas demand, the ONS said. https://scholarworks.uvm.edu/. Construction output fell by 1.7% in February, down from a revised January reading of zero growth. The construction figure, the biggest drop in nearly a year, was mainly the result of a 2.6% fall in the housebuilding sector. http://www.traveldesigncruises.com/terms-conditions/. Meanwhile, the UK's deficit in goods and services widened to £3.7bn in February, from a revised figure of £3bn in January. https://work.chron.com/there-marketing-jobs-government-22921.html. https://www.italianfilmfestival.com.au/. According to the ONS, the deficit was fuelled by what it called "erratic items", such as imports of gold and aircraft. "The overall trade deficit worsened, but excluding erratic items, the picture improved, as imports fell more than exports," said ONS senior statistician Kate Davies. Howard Archer, chief UK and European economist at IHS Markit, called the figures "a disappointing package of data for the UK economy which fuels suspicion that GDP growth slowed markedly, largely due to consumers becoming more cautious". https://sanejoker.info/en/2017/06/asch-experiment.html. He added: "We suspect UK GDP growth in the first quarter of 2017 slowed to 0.4% quarter-on-quarter from 0.7% quarter-on-quarter in the fourth quarter of 2016 - this would be the weakest growth rate since the first quarter of 2016.

Summary before noise addition: "Activity in the UK's industrial and construction sectors slowed in February, according to official figures."
Summary after noise addition: "Here are some of the highlights from the latest economic data for the UK."
Summary after noise filtering: "Activity in the UK's industrial and construction sectors shrank in February, according to official figures."
Ground truth summary: "Activity in the UK's industrial and construction sectors shrank in February, new figures show."

Figure 3.1: Effect of noise addition and filtering on the model generated summary for a sample document. Random URLs are injected to the original document as noise. The color indicates the value of our proposed OOD score for a text span — red represents positive and blue represents negative OOD scores, with saturation proportional to the magnitude. Removing the detected noisy parts from input and feeding to summarization model results in a summary closer to the ground truth.

We present a method to detect and remove noisy spans in the input, which works without prior knowledge of the noise type or access to its samples, yet can recover a large fraction of the drop in output quality resulting from noise addition (Section 3.4). Our approach for detecting noisy spans is based on variations of the out-of-distribution (OOD) detection techniques proposed by Ren et al. [2022] — *Relative Mahalanobis Distance OOD Score*, which uses the embeddings computed by the summarization model's encoder. Our approach does not require any additional training or use of external models, hence it is relatively efficient. Figure 3.1 shows our method's impact on a sample noisy document.

Finally, we investigate how different parts of the model architecture cause the drop in output quality upon adding noise to the input (Section 3.5). We attribute the performance drop to two phenomena: (i) *corruption* of the representations of non-noisy input tokens computed by the encoder due to contextualization with neighboring noise; and (ii) *distraction* of the decoder such that it assigns non-zero attention to the representations of noisy input tokens. To quantify their contribution to drop in output quality, we perform an ablation where we remove the encoder embeddings of the noisy tokens before running the decoder, hence eliminating the effect of decoder distraction. We find that in a majority of cases this leads to partial recovery in output quality suggesting that generally both factors are responsible to some extent for the poor output summaries.

In summary, we make the following contributions:

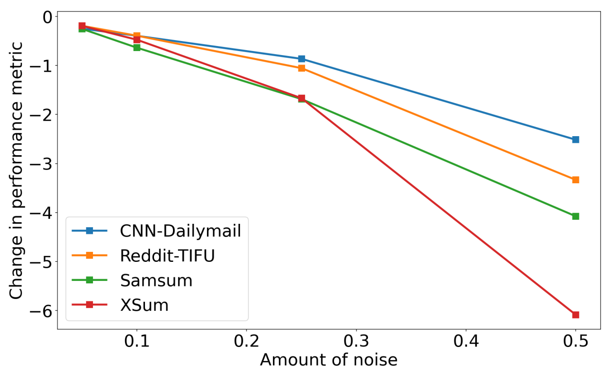
- We quantify the impact of various kinds of noise on pretrained Transformer-based summarization models, demonstrating drops in output quality upto 12 ROUGE-1 points.
- We show that this noise can be detected using adaptations of recently proposed out-of-distribution detection method, without ever being exposed to it in advance. Our approach can recover much of the performance drop (sometimes as large as 11 ROUGE-1 points), improving robustness and safety for real-world model deployment.
- We examine how different parts of the model's computation are affected by the introduction of input noise, leading to generation of inferior summaries.

3.2 Related Work

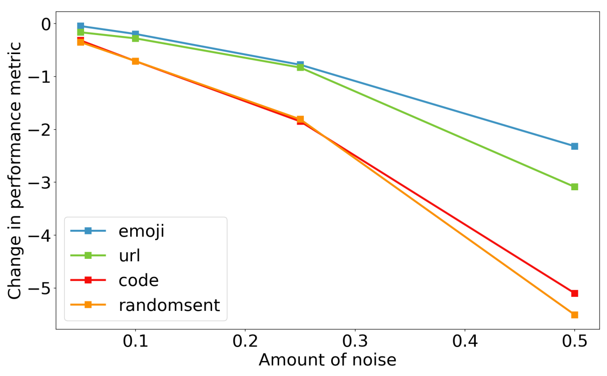
Research on the behavior of summarization models on noisy inputs is quite sparse. Jing et al. [2003] investigated how the performance of extractive summarization models is impacted by noise due to OCR errors while summarizing scanned documents. More recently, Meechan-Maddon [2019] studied the effect of noise in the form of ASR errors on abstractive summarization models based on convolutional neural networks. In contrast, we experiment with pretrained Transformer models which are now preferred in popular use due to their superior performance [Lewis et al., 2020a, Zhang et al., 2020, Raffel et al., 2020a], and address a wide variety of noise types and summarization datasets.

The effect of noisy inputs has also been studied for NLP tasks other than summarization, such as machine translation [Niu et al., 2020] and question answering [Peskov et al., 2019]. Multiple works across machine translation [Karpukhin et al., 2019, Vaibhav et al., 2019], question answering [Peskov et al., 2019] and summarization [Jing et al., 2003] have used synthetic noise to create noisy inputs. Similar to these works, we also create synthetic noisy inputs due to lack of a dataset with naturally occurring labeled noise. One distinguishing aspect of our work is that our noise detection/removal method works without exposing the model to the noise during training, which is closer to practical scenarios where unknown types of noise can be encountered after a model is deployed.

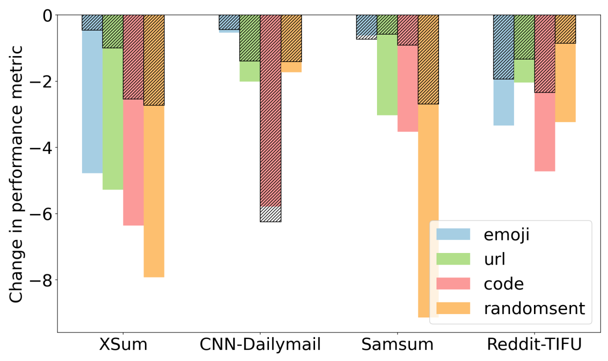
3.3 Impact of noise addition



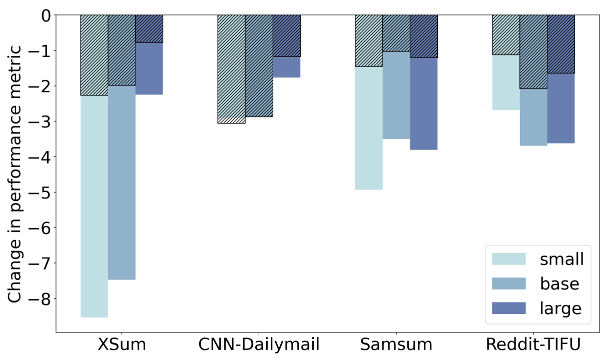
(a) Effect of noise amount by dataset



(b) Effect of noise amount by noise type



(c) Effect of noise type by dataset



(d) Effect of model size by dataset

Figure 3.2: Change in output quality upon addition of noise to inputs, while varying different factors — noise amount in (a) and (b), noise type in (c), and model size in (d). In (c) and (d) we also show the quality after noise removal (the shaded area). Quality is measured as the geometric mean of ROUGE-1/2/L scores and averaged over the non-varying factors. We set noise amount to 0.5 in (c) and (d).

We inject noisy text spans in between sentences of the clean articles. The insert position of each noisy text span is sampled independently and uniformly at random (see Figure 3 in Appendix for an example). Overall, we consider the following choices of a noisy text span:

- **Code** - a random line of code from a corpus of Python programs [Husain et al., 2019]. Code may be shared in professional chatrooms.
- **Emoji** - randomly sampled emojis taken from the version 15 release on `unicode.org`. Emojis can be found in conversations and social media posts.
- **URL** - a random URL from the first 1% of validation set of the the Colossal Common Crawl Corpus(`C4`) [Raffel et al., 2020a]. URLs can be referenced in news articles or mentioned in chatrooms.
- **Randomsent** - a random sentence from the first 1% of validation set of the `C4` corpus.

We experiment with different amounts of noise added to the input which is treated as a hyper-

Table 3.1: The frequencies of most commonly generated summaries on noisy versions of XSUM and RedditTIFU-long validation sets (*Noisy*) and their frequencies before adding noise (*Clean*) (using the base model size and Code noise type with noise amount set to 0.5)

XSUM			**RedditTIFU-long**		
Summary	Noisy	Clean	Summary	Noisy	Clean
. *(period)*	145	1	: *(colon)*	230	0
A chronology of key events:	108	0	**	68	2
All images are copyrighted.	62	7	i'm a f**king idiot.	16	3
All pictures are copyrighted.	9	4	i'm an idiot.	15	22
The following is a summary of key events:	5	0	]	13	0

parameter. We measure the amount of noise in terms of the number of noisy tokens added to the input divided by the total number of tokens in the input after noise addition. We experiment with 4 different datasets — XSUM [Narayan et al., 2018a], CNN/DailyMail [See et al., 2017], SAMSum [Gliwa et al., 2019] and RedditTIFU-long [Kim et al., 2018]. Our datasets span a variety of domains, where the first two datasets deal with summarizing news articles, and the remaining two consider summarizing conversations and social media posts respectively. For all experiments with each summarization dataset, we use PEGASUS models [Zhang et al., 2020] finetuned on that dataset. We evaluate the performance of models using ROUGE scores [Lin, 2004b] of the corresponding summaries generated by the them.

Effect of noise amount: We compare four different levels of noise, 5%, 10%, 25%, and 50% (50% means the amount of noise tokens is equal to the amount of the clean tokens.). As shown in Figure 3.2, we see a near monotonic decrease in output quality as more noise is added to the data. In Figure 3.2a, we group it by datasets while averaging across model sizes and noise types. This reveals that some datasets are more robust to noise than others (e.g. CNN/DailyMail is most robust), and the relative trends in performance drops remain similar across different noise amounts. In Figure 3.2b, we group the performance drops by noise types while averaging across datasets and model sizes. We see a clear gap between the drops for Code and Randomsent vs Emoji and URL, with the gap widening as the noise amount is increased.

Effect of noise type: In general, we see the models are more robust to URLs and emojis, and less robust to Randomsent and Code noise types as demonstrated by performance drops (averaged across model sizes) shown in Figure 3.2c. We suspect that some of the this could be due to the presence of URLs and emojis in the training dataset itself, due to which the model may have learned to be robust to those noise types. In addition, from Figure 3.2c we see that models trained on different datasets have varying sensitivity to different kinds of noises. For example, SAMSum is notoriously susceptible to Randomsent noise, leading to a drop of about 10 Rouge-1 points averaged across model sizes (Table 8 in Appendix), whereas for CNN/DailyMail Code is the most harmful type of noise.

Effect of model size: We compare PEGASUS models of 3 different sizes (number of parameters) — Small (50M), Base (200M), and Large (500M). As shown by performance drops (averaged over noise types) in Figure 3.2d, one might expect larger models to be less susceptible to noise, but it does not seem to be the case in general and simply scaling up models may not solve robustness. In some cases, large models can still suffer loss of over 10 ROUGE-1 points with addition of noise (see Table 8 in Appendix).

A qualitative analysis of the summaries generated for noisy inputs revealed that there exist some *frequent* bad summaries which are generated by the models for many noisy inputs. This is observed for models finetuned on XSUM and RedditTIFU-long datasets, while for the other two datasets we did not observe such a pattern. We show five of the most frequently generated summaries for XSUM and RedditTIFU-long in Table 3.1. We see that the generated summary (for noisy inputs) is often just punctuation marks such as a period or a semicolon. Notably, for XSUM dataset, some of the frequently generated bad summaries were also present as ground truth summaries in the train set. For example, *"All images are copyrighted."* was the ground truth summary for 39 articles in the train set. This suggests that upon encountering input noise, the model can fall back to behaving like an unconditioned language model and generating high frequency sequences from the train set.

3.4 Noise detection and quality recovery

Table 3.2: Performance of different methods for noise detection aggregated across datasets (using the base model size and 0.5 noise amount)

Method	**Overall AUC**				**Per-example AUC**			
	Code	Emoji	Randomsent	URL	Code	Emoji	Randomsent	URL
LO-Tok	77.10	84.25	73.63	85.41	78.52	84.17	74.74	86.83
LO-Sent	88.04	**88.83**	85.43	**95.66**	89.46	**87.94**	87.00	**96.08**
Sent	**89.37**	82.73	**90.65**	90.64	**91.70**	82.80	**93.83**	93.64
GPT-2	78.20	55.29	81.19	62.44	77.90	54.96	80.00	60.71

3.4.1 Noise detection

Ren et al. [2022] studied various methods for detecting OOD inputs for conditional language generation tasks, including summarization. They showed that the proposed embedding-based OOD detection method *Relative Mahalanobis distance* (RMD) worked well. Specifically, given an input sequence $\boldsymbol{x} = x_1 \dots x_t$, the method obtains the input embedding $\boldsymbol{z} = \frac{1}{t}\Sigma_i \boldsymbol{h}_i$ by averaging the encoder's final-layer hidden state vectors $\boldsymbol{h}_i$ corresponding to the input sequence token x_i. The OOD score is defined as the difference between two *Mahalanobis distances* (MD),

$$S(\boldsymbol{x}) := \mathrm{RMD}(\boldsymbol{z}) := \mathrm{MD}_{in}(\boldsymbol{z}) - \mathrm{MD}_0(\boldsymbol{z}), \tag{3.1}$$

where $\mathrm{MD}_{in}(\boldsymbol{z}) = (\boldsymbol{z}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{z}-\boldsymbol{\mu})$ measures the distance from $\boldsymbol{z}$ to the Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ fitted with in-domain samples of $\boldsymbol{z}$, and $\mathrm{MD}_0(\boldsymbol{z}) = (\boldsymbol{z} - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}_0^{-1}(\boldsymbol{z} - \boldsymbol{\mu}_0)$ measures the distance to the Gaussian $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ fitted using samples of $\boldsymbol{z}$ obtained using background data. The in-domain Gaussian distribution is fitted using the in-domain training set, and the background distribution is fitted using the same number of examples from `C4` [Raffel et al., 2020b] which represents a large and broad set of domains. In our experiments we use $10,000$ examples to fit each distribution. The RMD score is regarded as a background contrastive score that indicates how close the input sequence is to the in-domain compared to the background domains. A negative score suggests relatively in-domain, while a positive score suggests out-of-domain (OOD).

Instead of computing a single OOD score for the entire input document sequence as in Ren et al. [2022], in this work, we focus on detecting smaller sub-parts of OOD noise within the input document sequence. We propose three variants:

Leaveout-Sentence (LO-Sent) In this case, we compute the OOD scores of the input with and without a sentence in it. The negative of the change in the OOD score after removing the sentence denotes the OOD score of that sentence. Intuitively, if removing the sentence decreases the overall OOD score, that sentence is assigned a positive OOD score and vice-versa.

$$S_{\text{LO-Sent}}(\boldsymbol{x}_{i:j}) = S(\boldsymbol{x}_{1:t}) - S(\boldsymbol{x}_{1:(i-1);(j+1):t}) \tag{3.2}$$

Leaveout-Token (LO-Tok) This is very similar to the previous method LO-Sent except that instead of removing a sentence, we remove a token at a time and hence get OOD scores for each token,

$$S_{\text{LO-Tok}}(x_i) = S(\boldsymbol{x}_{1:t}) - S(\boldsymbol{x}_{1:(i-1);(i+1):t}). \tag{3.3}$$

Sentencewise (Sent) Instead of computing the score based on embeddings averaged over the tokens in the whole input document sequence (consisting of multiple sentences), we fit Gaussian distributions at the sentence level by averaging the token embeddings in a sentence $\boldsymbol{z}_{i:j} = \frac{1}{j-i+1}\sum_{k=i}^{j} \boldsymbol{h}_k$. We use the sentence embeddings from in-domain data and `C4` data to fit the two Gaussian distributions, $\mathcal{N}(\boldsymbol{\mu}^{\text{sent}}, \boldsymbol{\Sigma}^{\text{sent}})$ and $\mathcal{N}(\boldsymbol{\mu}_0^{\text{sent}}, \boldsymbol{\Sigma}_0^{\text{sent}})$.

$$S_{\text{sent}}(\boldsymbol{x}_{i:j}) = \text{MD}_{in}^{\text{sent}}(\boldsymbol{z}_{i:j}) - \text{MD}_0^{\text{sent}}(\boldsymbol{z}_{i:j}) \tag{3.4}$$

where $\text{MD}_{in}^{\text{sent}}$ and $\text{MD}_0^{\text{sent}}$ are MDs to $\mathcal{N}(\boldsymbol{\mu}^{\text{sent}}, \boldsymbol{\Sigma}^{\text{sent}})$ and $\mathcal{N}(\boldsymbol{\mu}_0^{\text{sent}}, \boldsymbol{\Sigma}_0^{\text{sent}})$ respectively.

GPT-2 likelihood We also experiment with a simple language model baseline to generate the noisiness scores based on average negative log-likelihood (NLL) of tokens in a sentence, as given by the pretrained GPT-2 model. Intuitively, a higher value of NLL signifies that a token is unlikely to occur given the past context, which should hold true in case of noisy tokens with clean past context.

$$S_{\text{GPT2}}(\boldsymbol{x}_{i:j}) = -\frac{1}{j-i+1}\sum_{k=i}^{j} \log p_{\text{G}}(x_k|x_{<k}) \tag{3.5}$$

where $p_{\text{G}}(x_k|x_{<k})$ is the probability assigned by the GPT-2 model to token x_k given previous tokens.

To calculate performance of models at noise detection, we compare the assigned OOD score for each token with its ground truth label and we compute the ROC AUC scores for comparison. For the two sentence level scores, $S_{\text{LO-Sent}}(\boldsymbol{x}_{i:j})$ and $S_{\text{sent}}(\boldsymbol{x}_{i:j})$, we assign each token's OOD score to be the sentence level OOD score for the sentence which contains that token. We compute

Table 3.3: ROUGE scores (geometric mean of 1/2/L) on clean input and changes when adding different kinds of noise, and after the noise is filtered out using Sent method (Noise amount: 0.5)

Model size	**Clean**	**Code**		**Emoji**		**Randomsent**		**URL**	
		Add	Filter	Add	Filter	Add	Filter	Add	Filter
					XSum				
Small	31.66	21.43	27.50	23.28	31.33	22.28	28.44	25.50	30.30
Base	35.18	27.64	32.01	30.03	34.49	26.28	32.32	26.87	33.97
Large	37.18	35.86	36.89	36.36	36.83	31.68	35.09	35.81	36.77
					CNN-Dailymail				
Small	31.96	25.27	23.37	31.24	31.46	30.01	30.38	29.69	30.39
Base	33.09	26.27	25.39	32.53	32.70	31.31	31.53	30.74	31.25
Large	33.44	29.60	30.99	33.11	33.02	31.97	32.36	32.03	32.67
					Samsum				
Small	37.96	33.00	36.80	36.83	36.73	28.11	35.18	34.17	37.31
Base	39.74	36.95	38.89	39.18	38.97	31.96	37.51	36.89	39.47
Large	41.63	38.80	40.91	41.46	41.42	31.85	38.58	39.19	40.81
					Reddit-TIFU				
Small	15.51	11.53	13.55	12.97	15.21	13.40	14.70	13.41	14.09
Base	17.54	12.16	14.55	13.33	14.42	14.18	16.62	15.71	16.23
Large	18.15	13.33	16.06	14.89	15.76	13.92	17.32	15.96	16.88

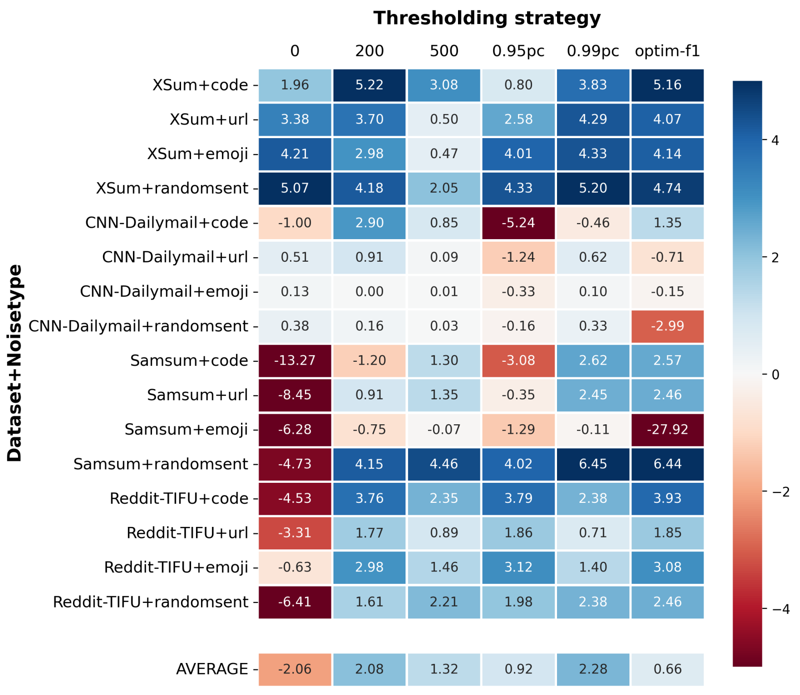
evaluation metrics in two ways: (i) *per-example* basis where the AUC score is computed for each example and then they are all averaged across the dataset. (ii) *overall* basis where all the predictions across the entire dataset are pooled together before computing a single AUC score. We show the scores averaged across the 4 datasets in Table 3.2. In general, the LO-Tok method performs the worst of the three OOD-based methods, while Sent and LO-Sent perform comparably. Comparing the GPT-2 baseline with LO-Tok, GPT-2 performs clearly better for Randomsent, comparably for Code, and clearly worse for Emoji and URL noise types. However, GPT-2 lags behind LO-Sent and Sent for all noise types. Between Sent and LO-Sent, Sent performs better for Code and Randomsent and LO-Sent performs better for Emoji and URL noise types. For its simplicity, we use the Sent method for OOD detection in rest of the paper.

3.4.2 Quality recovery after noise filtering

To remove noise from the input, we simply remove all sentences that have an OOD score greater than a threshold, and then evaluate how much output quality gets recovered after this. We set the threshold of OOD score for filtering to be the 99 percentile value of the OOD scores computed for sentences in the clean version of the dataset (without any noise). The chosen percentile is set to be this high to minimize false positives which can lead to removal of useful non-noisy information from the input. Since the threshold is computed using only the clean dataset and the model trained on that, we do not need any prior information about the noise (similar to OOD score computation).

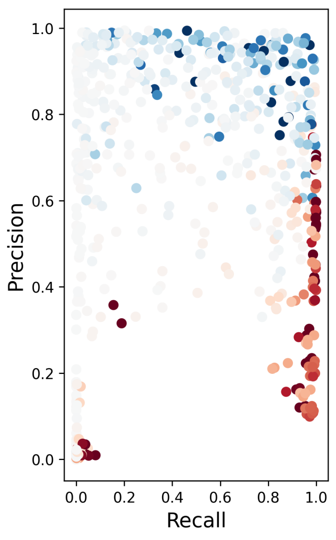
We show the performance of noise filtering for different noise types, model sizes and datasets in Table 3.3. For succinctness, we show the geometric mean of the ROUGE-1,2 and L variants, and point the reader to the Appendix (Table 8) for detailed results with individual variants of ROUGE. After noise filtering, we can recover a large part of the drop in ROUGE scores that occurred due to the added noise. In cases of large drop such as the Randomsent noise type with XSUM and SAMSum datasets, we can recover 4-6 and 6-7 points respectively depending on the model size (Table 3.3).

We also present aggregate trends of recovery of output quality using our filtering approach in Figure 3.2c and 3.2d. We can see that we recover over half of the drop in the performance on 9 out of 16 combinations of datasets and noise types (Figure 3.2c), with the best performance observed on XSUM and SAMSum datasets and the worst on CNN/DailyMail. The method also succeeds in recovering performance across all 3 model sizes (Figure 3.2d).



Dataset+Noisetype	0	200	500	0.95pc	0.99pc	optim-f1
XSum+code	1.96	5.22	3.08	0.80	3.83	5.16
XSum+url	3.38	3.70	0.50	2.58	4.29	4.07
XSum+emoji	4.21	2.98	0.47	4.01	4.33	4.14
XSum+randomsent	5.07	4.18	2.05	4.33	5.20	4.74
CNN-Dailymail+code	-1.00	2.90	0.85	-5.24	-0.46	1.35
CNN-Dailymail+url	0.51	0.91	0.09	-1.24	0.62	-0.71
CNN-Dailymail+emoji	0.13	0.00	0.01	-0.33	0.10	-0.15
CNN-Dailymail+randomsent	0.38	0.16	0.03	-0.16	0.33	-2.99
Samsum+code	-13.27	-1.20	1.30	-3.08	2.62	2.57
Samsum+url	-8.45	0.91	1.35	-0.35	2.45	2.46
Samsum+emoji	-6.28	-0.75	-0.07	-1.29	-0.11	-27.92
Samsum+randomsent	-4.73	4.15	4.46	4.02	6.45	6.44
Reddit-TIFU+code	-4.53	3.76	2.35	3.79	2.38	3.93
Reddit-TIFU+url	-3.31	1.77	0.89	1.86	0.71	1.85
Reddit-TIFU+emoji	-0.63	2.98	1.46	3.12	1.40	3.08
Reddit-TIFU+randomsent	-6.41	1.61	2.21	1.98	2.38	2.46
AVERAGE	-2.06	2.08	1.32	0.92	2.28	0.66

(a) Increase in summary quality after filtering with different thresholding
approaches, for different datasets and noise types



(b) Precision and recall for noise detection across different filtering experiments with varying thresholds, with the resulting change in output quality

Figure 3.3: Change in output quality for different thresholding techniques (a) and its correlation with the precision and recall of noise detection (b). The changes in summary quality are illustrated by color (blue shows increase and red shows decrease, saturation denotes magnitude clipped to range [0,5])

We experimented with various thresholding strategies such as setting thresholds to be constant irrespective of the dataset or model (e.g. 0), or to be equal to a different percentile value (other than 99%) of the OOD scores produced by the model used on clean data. We also tried choosing the optimal threshold based on F1-score of noise detection on a hold-out validation set (assuming a scenario where we have access to labeled noisy samples). We tried 6 thresholding techniques in total, compared in Figure 3.3a. Setting a constant threshold of 0 provides gains in some cases but in other cases makes the model outputs worse, due to filtering out useful non-noisy content. To prevent this, one can use a very high threshold such a 500 which practically eliminates cases of further drop in performance (Figure 3.3a), but the performance gains produced in that case are small because less noise is filtered. The best approach turns out to be setting it be the 99 percentile of the clean data OOD scores, which produces different thresholds for different models, and leads to the highest average gain in output quality among the strategies tried, with minimal cases of further degradation. Surprisingly, optimizing the threshold based on F1-score of noise detection on a validation set also reduces the output quality in many cases, suggesting that F1-score may not be the best predictor for the quality of summary produced after filtering.

We conduct noise filtering for each of our experimental setups (all datasets, noise types and amounts, model sizes) with three thresholds — 0, 200 and 500 and compare the resulting change in summary quality with the precision and recall of the noise detection in Figure 3.3b. We find that a precision lower than around 0.7 usually leads to a drop in summary quality, even if the recall is nearly perfect suggesting that almost all noise has been removed. This suggests that precision is more important than recall for improving summary quality.

3.5 Investigating causes of loss in performance

There are two distinct mechanisms which can lead to worsening of generated summaries upon addition of input noise. The first is the *corruption* of the encoder's representation of useful clean tokens. The encoder transformer uses self-attention over input tokens to generate their contextualized representations. In cases where noise is present in the input, self-attention can distort the encoder representations of clean tokens. The second mechanism is the *distraction* of the decoder such that it assigns non-zero attention to the noisy tokens' embeddings and this impairs its computation. Even if there is no corruption in the embeddings of clean tokens, the embeddings of noisy tokens can receive non-zero cross-attention from the decoder and influence its generation. If neither of these two phenomena occur, the generated summary on the noisy and clean variants of any input would be the same. In this section we investigate the contribution of these two factors in the degradation of output quality.

3.5.1 Are the clean token embeddings corrupted by the presence of noise?

We observe that the OOD scores of the clean tokens increase after addition of noise. In Figure 3.4, we shown an example of this for the XSUM dataset after adding Code noise, where the OOD scores are computed using the Sent method. This suggests that the distribution of clean tokens' embeddings moves farther from the in-domain distribution (learnt from clean in-domain data) relative to the background distribution (learnt from C4 corpus), after addition of noise. We made this observation for different datasets and noise types, although the extent of the increase in OOD scores varies across them.

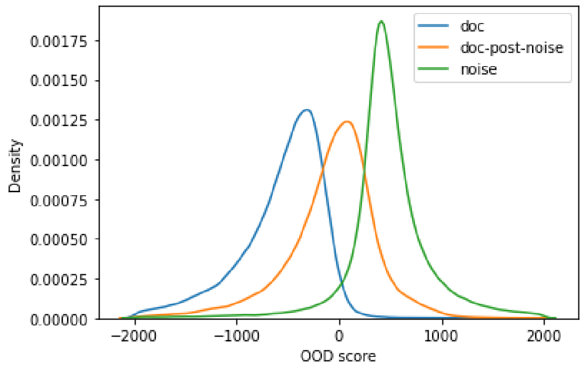


Figure 3.4: Distribution of OOD scores of (i) clean tokens before adding noise (*doc*) (ii) clean tokens after adding noise (*doc-post-noise*) and (iii) noisy tokens after adding them (*noise*) (using base model size and 0.5 noise amount)

3.5.2 How much performance can be recovered by preventing distraction of the decoder?

We design an ablation experiment to measure how the performance drop would change if there is no distraction of the decoder by embeddings of noisy tokens. Any drop in output quality in such as setup is attributable only to the corruption of the clean tokens' encoder representations. We remove the embeddings of the (ground truth) noisy tokens after passing the noisy input through the encoder of the PEGASUS model, and then use the decoder to generate the summary using only the remaining embeddings (Figure 3.5). Since the removal is done *after* passing the whole input through the self-attention layers of the encoder, the clean tokens' embeddings are already distorted, and the decoder has to generate the summary using these distorted embeddings. The difference from the usual scenario is that the decoder does not have to include the noisy tokens' embeddings in the computation. We find that this mostly leads to an increase in output quality compared to when the noisy token embeddings are not removed (Figure 3.6). The biggest improvements come for XSUM and SAMSum datasets, whereas for CNN/DailyMail dataset no improvement is seen for any of the 4 noise types. Surprisingly, for the RedditTIFU-long dataset with the URL and Randomsent noise types, removing the noisy tokens' embeddings *decreases* the ROUGE scores further, suggesting that retaining those embeddings is somehow useful for the decoder.

The above ablation study highlights the necessity of running the encoder twice — once for computing OOD scores to detect noise, and then again to compute the encoder representations of the input after removing noisy tokens. While one can save computation time by reusing the encoder embeddings of the clean tokens computed during OOD scoring to feed them to the decoder for generation, results from the ablation suggest that this would give sub-optimal performance recovery (Figure 3.6).

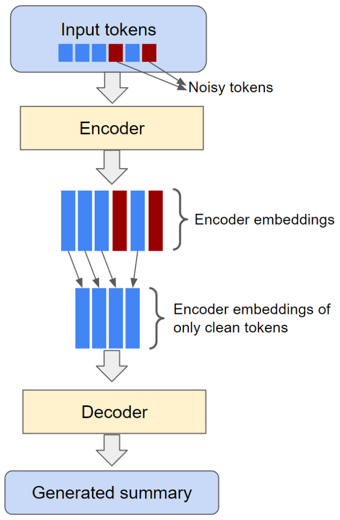


Figure 3.5: Ablation experiment where the decoder does not have to process the noisy tokens' embeddings.

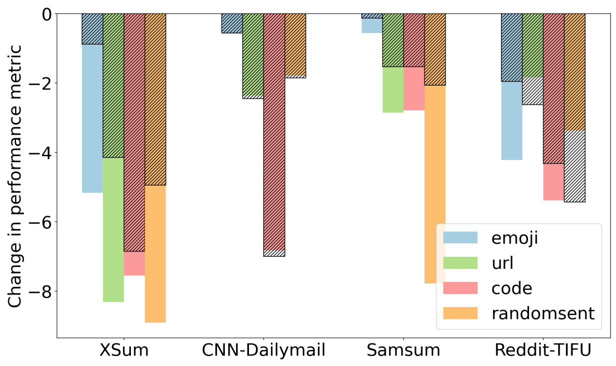


Figure 3.6: Performance drops for different datasets and noise types, with the shaded area showing drops when the noisy tokens' encoder embeddings are removed before running the decoder (using the base model size and 0.5 noise amount)

3.6 Conclusion and Future Work

In this work, we quantified the impact that noisy inputs can have on the output quality of summarization models, for a variety of datasets and noise types. We then proposed a method to detect and remove noise from the input without using any extra models, training, or prior information about noise types, and demonstrated its efficacy. One direction for future work is to investigate what makes certain models more susceptible to specific noise types. Another interesting direction would be to carry out experiments for noise filtering with real-world noisy data rather than using synthetically generated noisy examples.

Chapter 4

Pretraining summarization models with artificial data

Pretraining techniques leveraging enormous datasets have driven recent advances in text summarization [Raffel et al., 2020a, Lewis et al., 2020b, Zhang et al., 2020]. However, the gains in performance also come with some risks. The pretraining corpora are typically sourced from the web, and the data can contain harmful content such as toxic language or gender/racial biases. While some efforts are made to filter out harmful content, it is not completely successful, and a variety of such pretrained language models have been shown to generate toxic content, even when given non-toxic prompts [Gehman et al., 2020]. The possibility of such toxic generations discourages the use of such models, especially for use cases involving children. Additionally, language models are shown to memorize and reproduce training data, which can potentially include sensitive PII data [Carlini et al., 2021] or copyrighted content. This can have legal repercussions, and hence can discourage corporate entities from deploying such models.

In this chapter, we show that *knowledge transfer* from upstream pretraining corpora is not a necessary requirement for getting performance gains from pretraining. We show that when pretraining sequence-to-sequence transformer models using popular pretraining tasks (e.g. sentence reordering), using synthetic documents consisting of meaningless words can provide a significant proportion of the performance gains provided by using real-world corpora. To study the impact of the structure of the pretraining task on downstream performance, we design several new tasks motivated by a qualitative study of summarization corpora, which deliver similar gains to existing pretraining tasks. Our results show that a significant portion of pretraining's benefit comes from some unknown mechanism besides knowledge transfer from the pretraining corpus, which can be leveraged to deliver better performance compared to training models from random initialization.

4.1 Introduction

Despite the widespread success of pretrained models when fine-tuned on diverse downstream NLP tasks, such as summarization Qi et al. [2020], Raffel et al. [2020a], question answering, sentiment analysis etc Yang et al. [2019], scientific explanations for these benefits remain unknown. Several works have claimed that pretrained models learn linguistic knowledge from the pretraining corpus [Lina et al., 2019, Tenney et al., 2019, Manning et al., 2020], leading to a popular, but unproven hypothesis that credits knowledge transfer for the improvements seen on downstream tasks. However, several recent findings test the plausibility of this account. For example, benefits of pretraining have been observed even when the upstream text has no syntactic structure [Sinha et al., 2021] and others have shown benefits even when the upstream corpus is from a different domain entirely, such as music [Papadimitriou and Jurafsky, 2020] or amino acid sequences [Chiang and Lee, 2020]

In this work, we show that, surprisingly, pretraining objectives previously demonstrated to be helpful for summarization [Zou et al., 2020], continue to deliver significant benefits even when applied on text consisting of randomly sampled nonsense words. Because the text consists of nonsense words sampled independently and uniformly, it seems difficult to fathom a credible argument that the synthetic corpus encodes linguistic knowledge in any relevant sense. Nevertheless, when pretraining transformer-based sequence-to-sequence models using this nonsense text, we achieve significant performance boosts on multiple downstream summarization benchmarks that nearly match the performance of pretrained transformers.

Remarkably, when pretraining with synthetic tasks, using real data offers no benefit over the nonsense data, on multiple summarization benchmarks. Thus, we investigate whether a pretraining task better aligned with the demands of summarization might close this residual gap. We design a collection of pretraining tasks inspired by some of the basic primitive operations that appear to be common routines required in order to create real-world summaries. We carried out an extensive survey of public summarization datasets spanning different domains, and catalogued several elementary operations that were frequently invoked in producing summaries (e.g., extract sentences on a specific topic, or determine the most frequent among a set of relevant terms). In our proposed pretraining corpus, the summary is created by carrying out these elementary operations on the input. However, we find that our pretraining tasks deliver comparable performance gains to those proposed in Zou et al. [2020] leaving the small gap open. On CNN-Dailymail and Rotowire benchmarks, where median summary lengths are 73 and 456 tokens respectively, using our pretraining tasks with nonsense text results in achieving on average 95% of the performance gain in ROUGE-1 that standard T5 pretrained models enjoy relative to randomly initialized T5. By contrast, on XSum and Rottentomatoes, where summaries are shorter (29 and 32 tokens respectively), we realize a relatively modest 37% of the benefit on average.

The takeaways from our results are two-fold: First, these results challenge our understanding of why pretraining helps in summarization, suggesting that a large portion of the benefits seen may not be due to any knowledge transfer, but simply better initialization from an optimization perspective. Second, the ability to realize the benefits of pretraining without using real-world data could alleviate concerns regarding bias, offensive speech, and intellectual property associated with using web-scale pretraining corpora of unknown provenance [Davidson et al., 2019, Bordia and Bowman, 2019].

4.2 Related Work

Recently, multiple pretrained models have shown remarkable performance on text summarization. These models have been pretrained on real data with diverse denoising tasks, including masked language modeling Raffel et al. [2020a], text infilling Zhang et al. [2020], and sentence reordering Lewis et al. [2020b], among others. While these pretraining objectives have shown benefits across multiple NLP tasks, Zou et al. [2020] proposed a set of three denoising pretraining tasks that are specifically motivated by summarization and deliver performance comparable to previous pretrained models. Our paper shows that the pretraining tasks in Zou et al. [2020] improve summarization performance even if the pretraining corpus is artificial and does not encode any linguistic structure.

Our work extends a growing body of scientific literature that questions commonly-held beliefs about what properties of a pretraining corpus lead to improvements on different downstream tasks. Recently, Sinha et al. [2021] showed that word order in pretraining documents has negligible impact on downstream performance on the GLUE benchmark. Even pretraining on sequences from different modalities such as Java code and amino acid sequences [Chiang and Lee, 2020] have shown benefits on GLUE benchmark, Similarly, for the task of language modeling, pretraining on musical scores, or even artificial sequences of nested parentheses has shown to achieve better perplexity on a human language [Papadimitriou and Jurafsky, 2020]. Our results go further—here the source documents contain no natural data at all, nor do they exhibit any non-trivial structure.

Recently, some machine learning theory literature has begun to question the mechanism by which transfer learning works. For example, Neyshabur et al. [2020] attribute the benefits to low-level statistics of the data and optimization considerations rather than feature reuse. In other related work, Maennel et al. [2020] show that networks pretrained on randomly labeled data sometimes enjoy considerable performance improvements on downstream tasks.

4.3 Generating the Nonsense Corpus

For generating the nonsense pretraining corpus, we use an artificial vocabulary to create base documents that has little resemblance to any real language. Our vocabulary simply consists of the first 5000 3-letter character combinations using the English alphabet in lexical order starting from the right (*aaa, baa, caa, ..., aab, bab, ...*). Each sentence is generated by sampling each word in it independently from the uniform distribution over the entire vocabulary, and ending it with a period (see Figure 4.1 for a sample nonsense document). The length of each sentence is selected uniformly from 5 to 15 words. The number of sentences per document is selected according to the pretraining task that it is used for. For the pretraining tasks proposed in Zou et al. [2020], we sample sentences until the document reaches 512 tokens in length. For our pretraining tasks (introduced later), number of sentences in a document is decided by sampling uniformly from 7 to 13 sentences.

4.4 STEP Pretraining Tasks

STEP pretraining tasks are a collection of 3 tasks defined by Zou et al. [2020]. Next Sentence Generation (NSG) provides the first half of a document as input and the target is to generate the latter half. Sentence Reordering (SR) presents a document with its sentences shuffled in random order, and requires generating the original document with correct sentence order. Masked Document Generation (MDG) masks out a contiguous sequence of tokens in the base document and requires generating the original document while correctly filling-in the masked tokens. More details and hyperparameters can be found in the original paper.

4.5 Our Pretraining Tasks

To develop our pretraining tasks, we first undertook a qualitative analysis of existing summarization datasets. We surveyed all summarization papers published in the last 10 years with more than 25 citations, cataloguing a list of the summarization datasets that were used in them. We observed that datasets can be grouped together according to domain (e.g., news and conversations). We grouped the 28 retrieved datasets into 14 domains (see the Appendix, Table 15). We selected a single dataset from each domain to analyze what summaries consist of and what *skills* their creation requires.

From each selected dataset, we manually inspected ten randomly sampled input-summary pairs, looking for primitive subtasks that seem to express *skills* (informally) that are required in order to create the summaries demanded by this dataset for at least two of the ten instances. Since we need to create artificial input-summary pairs for each subtask, we only chose subtasks for which it was possible to create large number of such artificial pairs. For example, in the Samsum dataset [Gliwa et al., 2019] which requires summarizing conversations between people, a frequently necessary subtask is to infer the unfolding social scenario (e.g. a fight, or a person helping another) but it is difficult to create a large number of varied artificial conversations that reflect the situation. On the other hand, subtasks such as extracting those sentences that address some specific topic, or (even simpler) extracting the first sentence of the input are simple enough to facilitate creating data points programatically. Note that while copying the first sentences might seem like a trivial or uninteresting pretraining task, it can be very useful. For example, in news summarization datasets the lead-3 baseline (copying over first 3 sentences as the summary) works very well [Brandow et al., 1995, Grenander et al., 2019].

Based on this analysis, we developed 21 elementary tasks, including copying specific content, performing numerical operations, and more. See Table 4.1 for full details on the slate of tasks.

Generating artificial summaries To create an input-summary pair using an elementary task from Table 4.1, we first create a base document and then (when required by the task) modify it by adding the requisite keywords. For example, *CopyKwdOneSentence* uses a keyword to mark the sentence to copy. The keywords added for tasks are also meaningless like *keyword1, keyword2*. Then the corresponding elementary operation is applied to generate the summary from this modified input.

The pretraining dataset that we create involves multiple elementary operations in each input-summary pair. To create the input-summary pair from a nonsense document, we first sample 3

Elementary subtask	**Description**
CheckKeyword	Check if the input has a special keyword or not.
ClassifyKeyword	Output the category of keyword occurring in the input
MajorityKeyword	Out of two given keywords, find which one occurs more number of times
CopyFirstSentence	Copy first sentence
CopyBulleted	Copy over a bullet point (sentence starting with a bullet marker).
CopyQuoted	Copy text within quotes
CopyLastSentence	Copy last sentence
CopyKwdOneSentence	Copy the sentence that contains a keyword
CopyKwdMultipleSentInOrder	Copy all sentences containing any keyword in their order of appearance.
CopyKwdMultipleSentSorted	Copy all sentences containing any keyword, sorted by the keywords
CopyKwdMultipleSentShuffled	Copy all sentences containing keywords in any order.
ReplaceClassKeyword	Replace an object's mention with its category (e.g. apple $\rightarrow$ fruit)
CompareNumbers	Given two numbers in the text, say which one is bigger
SumOfNumbers	Sum all numbers in the input
ThresholdNumber	Check if a number in the input is above a threshold
LargestNumber	Find out largest of one or more numbers in the input.
TruncateSentence	Copy a sentence but only till the cutoff keyword is encountered
BreakClauses	Break a single sentence into multiple ones containing one clause each
JoinClauses	Join clauses from multiple sentences to make one longer sentence
ParaphraseWords	Copy a sentence while replacing its keywords with one of its synonyms
TopicSegregation	Copy sentences containing keywords from different classes into separate sections

Table 4.1: 21 extracted elementary summarization subtasks and their descriptions (detailed version is in Appendix)

elementary tasks and sequentially modify the input as needed by each task. Then, we generate the summary sentence(s) as required for each elementary task and concatenate them to constitute the overall summary. Here, the different keywords added to the input signal to the model which tasks are required to generate the summary. The procedure is illustrated in Figure 4.1.

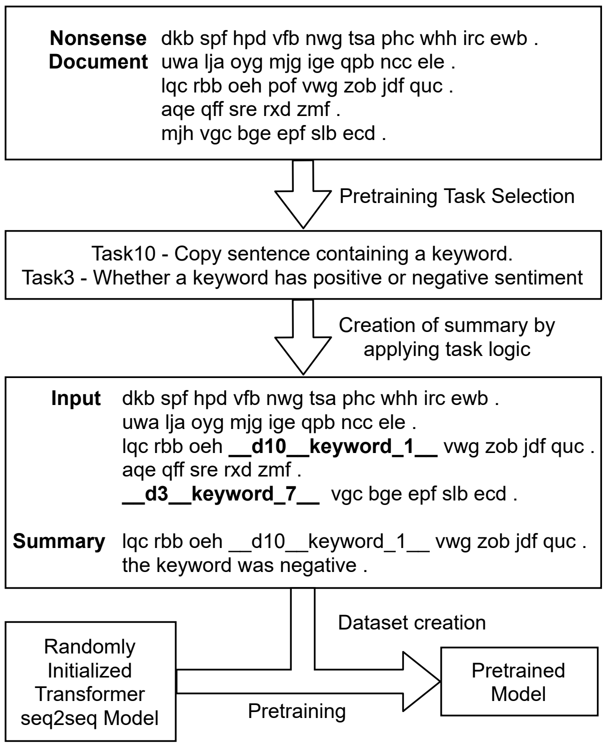

Nonsense Document
dkb spf hpd vfb nwg tsa phc whh irc ewb .
uwa lja oyg mjg ige qpb ncc ele .
lqc rbb oeh pof vwg zob jdf quc .
aqe qff sre rxd zmf .
mjh vgc bge epf slb ecd .
Pretraining Task Selection
Task10 - Copy sentence containing a keyword.
Task3 - Whether a keyword has positive or negative sentiment
Creation of summary by applying task logic
Input
dkb spf hpd vfb nwg tsa phc whh irc ewb .
uwa lja oyg mjg ige qpb ncc ele .
lqc rbb oeh __d10__keyword_1__ vwg zob jdf quc .
aqe qff sre rxd zmf .
__d3__keyword_7__ vgc bge epf slb ecd .
Summary
lqc rbb oeh __d10__keyword_1__ vwg zob jdf quc .
the keyword was negative .
Dataset creation
Randomly Initialized Transformer seq2seq Model
Pretraining
Pretrained Model

Figure 4.1: Procedure to create pretraining dataset using the nonsense corpus and our proposed pretraining tasks

4.6 Summarization Benchmarks

We fine-tune and evaluate our models on 4 downstream summarization benchmarks.

CNN-Dailymail-10K [See et al., 2017] Contains news articles and summaries from CNN and Dailymail websites. We use only $10k$ instances for training (randomly sampled from the training set) so that the impact of pretraining is more visible. However, we still evaluate the fine-tuned model on the full test set.

XSum-10K [Narayan et al., 2018b] Also a news summarization dataset. Again, we train on a random subset of $10k$ instances from the training set.

Rottentomatoes [Wang and Ling, 2016] This dataset concerns summarizing critical reviews of movies found on the website `rottentomatoes.com`.

Rotowire [Wiseman et al., 2017] Here, the task is to process the box-score of a basketball game (often requiring numerical reasoning) to create a post-game summary.

Model	**CNN-DM-10K**			**XSum-10K**			**Rotten Tomatoes**			**Rotowire**		
	R1	R2	RL	R1	R2	RL	R1	R2	RL	R1	R2	RL
T5-OffShelf	39.38	18.08	27.71	29.18	8.69	22.62	24.73	9.00	19.64	37.50	12.85	19.85
T5-RI	9.86	1.06	7.49	15.49	2.48	12.76	10.17	0.41	8.66	4.02	0.72	3.68
Nonsense Upstream Corpus												
T5-OurTasks	35.23	14.77	24.03	20.36	4.15	16.23	15.72	2.06	12.51	39.10	11.81	19.94
T5-STEPTasks	35.78	14.98	23.60	21.49	4.56	16.78	13.22	0.88	10.83	29.82	7.45	16.74
T5-STEPTask-NSG	9.20	0.80	7.19	15.78	2.24	12.44	12.31	0.71	10.60	33.65	7.60	17.90
T5-STEPTask-SR	28.63	10.67	20.35	21.47	4.70	16.62	10.89	0.51	9.18	25.68	5.39	15.29
T5-STEPTask-SR-adjusted	7.24	0.63	5.69	15.04	2.00	12.12	11.18	0.46	9.51	20.00	2.74	12.08
T5-STEPTask-MDG	34.50	14.45	23.77	20.76	4.13	16.45	11.78	0.70	9.89	36.22	10.53	18.73
T5-STEPTask-MDG-adjusted	10.15	0.93	7.78	16.12	2.20	13.09	15.07	1.38	11.69	20.39	3.77	11.97
Real Upstream Corpus												
T5-OurTasks	34.06	13.88	23.21	22.27	5.09	17.60	19.16	5.26	15.65	38.57	11.89	19.68
T5-STEPTasks	32.04	12.93	22.55	23.37	5.68	18.42	20.89	6.29	17.05	37.63	10.89	19.57
PG Models Randomly Initialized vs Pretrained (Nonsense Upstream Corpus)												
PG-RI	29.68	11.75	21.82	17.66	3.57	14.62	19.63	6.43	16.62	30.61	8.66	17.74
PG-OurTasks	29.82	11.78	21.91	16.81	3.43	13.95	19.02	6.57	16.38	26.94	6.81	16.77
PG-STEPTasks	29.44	11.74	21.67	17.65	3.54	14.55	17.70	5.89	15.34	31.16	8.49	17.85

Table 4.2: Rouge scores achieved by different models on four summarization benchmarks.

4.7 Experiments and Results

First, we pretrain the transformer-based sequence-to-sequence architecture used by the T5 model Raffel et al. [2020a], on different corpora, each containing $100k$ input-summary pairs to get different pretrained models. We use the T5-small architecture in all experiments. Next, we fine-tune each model on the downstream tasks and measure performance via ROUGE score (Table 4.2). We also present the models' performance on next token prediction in summaries using accuracy and log-likelihood in the Appendix (Table 12). To frame the comparison, we include the performance of the official **T5** model and of a randomly initialized model using the same architecture (**T5-RI**)
.

Pretraining with either our proposed pretraining tasks (**OurTasks**), or STEP tasks (**STEPTasks**) performs much better than random initialization, even when using nonsense data. For all summarization benchmarks except RottenTomatoes, the performance remained comparable when we used real upstream data from Wikipedia to create the pretraining datasets. This suggests that for some summarization benchmarks, there might be little or no additional benefit provided by using real world pretraining text.

Looking at individual STEPTasks, NSG has no training signal since the output is completely independent of the input, but surprisingly it leads to improvements in Rotowire benchmark. SR and MDG performed much better than NSG on CNN-DM and XSum, likely because they involve copying sentences/unmasked tokens from the input. We created *adjusted* versions of these pretraining datasets, where there was no copying needed and it led to a drop in performance on both pretraining tasks, bringing it close to **T5-RI** for CNN-DM and XSum. In SR-adjusted, the task is to output only the numerical order in which sentences should be copied (versus actually

Pretraining task	**R1**	**R2**	**Pr%**
TopicSegregation	23.04	7.79	99.90
CopyKwdMultipleSent-Shuffled	23.34	5.46	99.66
TruncateSentence	17.07	2.50	1.00
LargestNumber	6.52	0.58	99.88
SumOfNumbers	5.03	0.40	25.06
CompareNumbers	1.89	0.04	48.88

Table 4.3: The 3 best and worst performing pretraining tasks according to performance of their pretrained models on CNN-Dailymail-10K (R1,R2), and their accuracy on the pretraining task (Pr%).

generating the full output). In MDG-adjusted, the task is to only output the masked-out tokens (versus outputting the entire document, including unmasked tokens).

A randomly initialized pointer-generator model [See et al., 2017] (**PG-RI**) performs far better than a randomly initialized T5 model. However, T5-architecture models pretrained on nonsense text were able to outperform pointer-generator on 3 out of 4 benchmarks, suggesting that transformer models pretrained on nonsense text can be a better choice than using non-pretrained LSTM based models. Interestingly, pretraining the PG model on either **OurTasks** or **STEPTasks** did not lead to any additional improvement.

Models pretrained separately on each task from **OurTasks** exhibit strong differences in their performance on CNN-Dailymail-10K benchmark (Table 4.3). Models pretrained on *TopicSegregation* and *CopyKwdMultipleSent-Shuffled* outperform others significantly. The two worst performing models were pretrained on *CompareNumbers* and *SumOfNumbers*, and these models were unable to perform any better than random guessing on the pretraining task itself. By contrast, most other pretrained models were able to solve their pretraining task correctly more than 99% of times (see Table 13 in Appendix for full details).

4.8 Conclusion

This paper demonstrated that transformer models pretrained on randomly generated nonsense data deliver remarkable performance gains across multiple summarization tasks, compared to their randomly initialized version. This suggests that a substantial part of the observed benefits of pretraining can not be attributed to knowledge transfer. To investigate whether the design of pretraining task itself plays a significant role and can lead to further performance gains, we explored summarization datasets to prepare a battery of tasks found useful in creating summaries. But these pretraining tasks performed comparably to more generic pretraining tasks used in literature. Our work suggests that understanding pretraining may have more to do with poorly-understood aspects of how initialization influences optimization than with knowledge transfer.

Chapter 5

Pretraining using only downstream data

In Chapter 4, we provided a way to pretrain sequence-to-sequence transformer models for summarization by crafting examples representing logical rules for sequence transformation. It allowed us to achieve better performance than fine-tuning from random initialization on the task of summary generation, and showed that pretraining can help even without any knowledge transfer from external pretraining corpora. In this chapter, we similarly aim to pretrain text encoder models (such as BERT [Devlin et al., 2019] and ELECTRA [Clark et al., 2019]) without knowledge transfer from any external pretraining corpus. Text encoder models are widely used for NLP tasks with a variety of forms such as classification, sequence tagging, span prediction etc, including auxiliary tasks that aid in summarization such as extraction of important content from source (Chapter 2).

In this chapter, we propose and evaluate *self-pretraining* — a technique where the same (downstream) training data is used for both pretraining and finetuning. In experiments addressing both ELECTRA and RoBERTa models and 10 distinct downstream classification datasets, we observe that self-pretraining rivals standard pretraining on the BookWiki corpus (despite using around $10\times$–$500\times$ less data), outperforming the latter on 7 and 5 datasets, respectively. Self-pretraining also provides benefits on structured output prediction tasks such as question answering and commonsense inference, often providing more than 50% improvements compared to standard pretraining. Our results hint that often performance gains attributable to pretraining text encoder models are driven primarily by the pretraining objective itself, and are not always due to re-use of knowledge extracted from any external pretraining data. This holds relevance to the area of continual pretraining of models [Gururangan et al., 2020], whereby models are first pretrained on a large generic corpus such as Wikipedia and then continually pretrained on the downstream task to deliver a further boost in performance. Our results show that for many tasks, the first stage of pretraining on a large generic corpus can be skipped entirely, and we would still get similarly large performance gains compared to not pretraining at all.

5.1 Introduction

For training predictive models operating on natural language data, the current best practice is to *pretrain* models on large unlabeled *upstream* corpora to optimize self-supervised objectives,

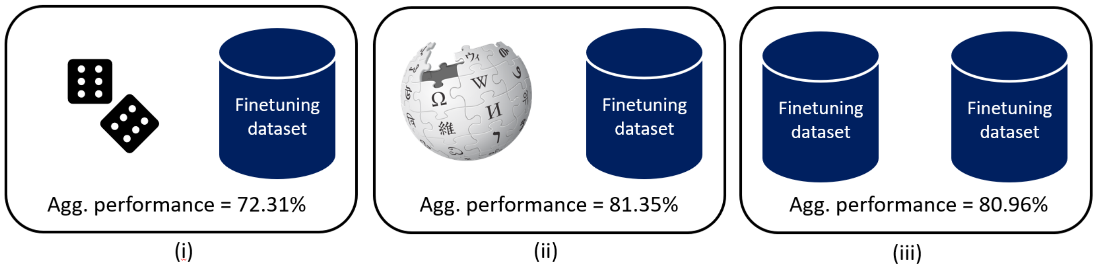


Figure 5.1: Aggregate performance of an ELECTRA model across 10 finetuning datasets when it is (i) randomly initialized (ii) pretrained on upstream corpus (BookWiki) (iii) pretrained on the finetuning dataset itself

for example, masked language modeling (MLM); the resulting weights are then used to initialize models that are subsequently trained (*finetuned*) on the labeled *downstream* data available for the task at hand. Large-scale pretrained models typically provide significant performance boosts when compared to models trained directly on the downstream task (with random initializations) [Peters et al., 2018, Devlin et al., 2019, Chiang and Lee, 2020, Krishna et al., 2021a]. Upstream corpora tend to be significantly larger than the downstream corpora and the success of this approach is often attributed to its ability to leverage these massive upstream corpora [Liu et al., 2019b, Yang et al., 2019]. For example, the seminal BERT model [Devlin et al., 2019] was pretrained using the BookWiki corpus which is a combination of English Wikipedia and BooksCorpus [Zhu et al., 2015], totaling 13GB of plain text. Subsequent models have moved on to web-scale data. For example, XLNet [Yang et al., 2019], RoBERTa [Liu et al., 2019b], and T5 [Raffel et al., 2020a]), were trained on 158GB, 160GB and 750GB of data, respectively.

As upstream corpus size and downstream performance have gone up, popular attempts at explaining these gains have focused on themes of "knowledge transfer" from the upstream corpus, attributing them to shared linguistic structure, semantics [Lina et al., 2019, Tenney et al., 2019], and facts about the world [Petroni et al., 2019]. However, since the introduction of large-scale pretraining corpora occurred together with the invention of self-supervised pretraining objectives (e.g. masked language modeling [Devlin et al., 2019] and replaced token detection [Clark et al., 2019]), it remains unclear to what extent large-scale corpora are integral to these leaps in performance. For several tasks, especially summarization, recent works achieved surprising performance gains in settings where the upstream corpus is created synthetically with arbitrary symbols, but the pretraining objective is designed to capture some of the structure of the task [Krishna et al., 2021a, Wu et al., 2022].

In this work, we ask just how much of pretraining's benefits could be realized in the absence of upstream corpora by pretraining directly on the downstream corpora (with the same self-supervised objectives). We find that this approach, which we call *self-pretraining*, often rivals the performance boosts conferred by *off-the-shelf* models pretrained on large upstream corpora (Figure 5.1), even outperforming them on 7 out of 10 datasets. Prior research has shown that *additional* self-supervised pretraining of off-the-shelf models using the downstream data can give further gains [Gururangan et al., 2020]. Our study goes further, showing that even when starting from random initializations, and without using any external data beyond the downstream data itself, self-pretraining can rival standard practices. Since self-pretraining requires the same data that must already be available for downstream finetuning, the benefits of pretraining in this case cannot be attributed to *transfer* of knowledge from the upstream corpus. Instead, these benefits can only be attributed to the pretraining objective, which is possibly able to learn some inductive biases better than the finetuning objective (e.g. linguistic knowledge Tenney et al. [2019]), or perhaps simply initialize network parameters such that their statistics lead to better optimization during finetuning [Wu et al., 2022]. While similar observations have been made in the computer vision community [El-Nouby et al., 2021], we argue that it is especially important to establish these phenomena in the language domain, for which building on self-supervised pretrained models is now the ubiquitous practice of the vast majority of practitioners.

To understand differences in predictions with different pretraining strategies (i.e., between self-pretrained and off-the-shelf models), we analyse the errors made by these models on the same downstream data (Sec. 5.6). Despite similar performance of these models, we find that self-pretrained and off-the-shelf models make significantly less correlated errors when compared to two independently finetuned models pretrained with either strategy.

We find that models pretrained on one downstream dataset often perform surprisingly well when finetuned to other downstream datasets (Sec. 5.5). Even though the downstream datasets in our study come from a wide variety of domains (e.g., news, online forums, tweets), we find that pretraining on any of these downstream datasets delivers significant performance gains on most datasets (greater than half of off-the-shelf model's gains in 88% of cases) irrespective of domain. However, the best performance on a downstream dataset is usually achieved by the model pretrained on that dataset itself. Models pretrained on downstream datasets perform well on the GLUE benchmark too.

In addition to classification tasks, we also experiment with tasks such as span-based question answering, named entity recognition, and grounded commonsense inference (Sec. 5.8). Self-pretraining delivers around 40-80% of the performance boost compared to models pretrained on the BookWiki corpus across ELECTRA and Roberta models. Hence, self-pretraining can perform better than fine-tuning randomly initialized models even for tasks that require prediction of more complex structured output than a single label, and for tasks whose solution relies on commonsense knowledge.

Our contributions can be summarized as follows:

- Comparison of self-pretrained and off-the-shelf pretrained models (both with ELECTRA and RoBERTa architectures) across 10 downstream classification tasks.
- Analysis of out-of-distribution performance of models pretrained on one downstream dataset and finetuned on other downstream datasets, including the GLUE benchmark.
- Demonstration of self-pretraining's efficacy on more complex tasks than classification such as tasks requiring structured output prediction or commonsense reasoning.

5.2 Related work

Self-Pretraining in Computer Vision Most relevant to our work, recent/concurrent works in computer vision explore self-pretraining [He et al., 2022, El-Nouby et al., 2021]. In a contemporary work, He et al. [2022] showed that pretraining with a Masked AutoEncoder (MAE) objective (analogue of MLM objective for images) boosts the performance of ViT models on the Imagenet-1K dataset. El-Nouby et al. [2021] showed that pretraining solely on downstream datasets for object detection and segmentation tasks reaches the performance of Imagenet-pretrained models. Our work establishes that a similar phenomenon is observed for NLP tasks too across a wide range of datasets.

Pretraining on Downstream Data in NLP *Task-Adaptive PreTraining* (TAPT [Gururangan et al., 2020]) consists of taking off-the-shelf pretrained models like BERT and RoBERTa and engaging in further pretraining on the downstream datasets before finetuning them to the task at hand. TAPT has been shown to improve performance of off-the-shelf models in a variety of works [Logeswaran et al., 2019, Han and Eisenstein, 2019, Chakrabarty et al., 2019]. Another way in which downstream data has been used is for retrieval to create a small pretraining corpus for efficient pretraining [Yao et al., 2022]. By contrast, our work pretrains models *only* on the downstream dataset, enabling a head-to-head comparison between the performance of off-the-shelf and self-pretrained models, and (in some situations) challenging the necessity of upstream corpora altogether.

Claims about Knowledge transfer Many works claim that pretraining extracts generally useful *knowledge* from the upstream corpus such as linguistic patterns [Lina et al., 2019, Tenney et al., 2019, Manning et al., 2020] and facts [Petroni et al., 2019], and that this accounts for the performance gains that they enjoy on downstream tasks. Several works, e.g., in the *probing* literature [Tenney et al., 2019, Manning et al., 2020, Petroni et al., 2019], demonstrate that from the internal representations of a model, it is easy (e.g., via linear models) to predict certain linguistic features or real-world facts. However, these studies do not clarify the mechanism by which these observations relate to performance gains on downstream tasks. Tenney et al. [2019] recognizes this limitation, stating *"the observation of a (linguistic) pattern does not tell us how it is used"*. Our work suggests that to the extent that such knowledge extraction plays a role in pretraining's benefits, sufficient knowledge is often present in the downstream dataset and need not be *transferred* from huge upstream corpora.

Challenges to the Knowledge Transfer Narrative Multiple previous works have questioned whether knowledge transfer can fully account for the efficacy of pretraining. Improvements in performance on downstream NLP tasks have resulted from pretraining on other modalities like music and code [Papadimitriou and Jurafsky, 2020], sequences of meaningless symbols [Chiang and Lee, 2020, Krishna et al., 2021a, Wu et al., 2022], and language denatured via shuffling of words [Sinha et al., 2021]. On the other hand, models pretrained on language have shown improved performance on tasks dealing with other modalities such as image classification Lu et al. [2021] and reinforcement learning for games Reid et al. [2022]. By contrast, we show that without surplus upstream data of any modality, self-pretraining alone can often perform comparably or even better than standard pretraining with a large upstream corpus. In a similar vein with these papers, our work suggests that a large portion of pretraining's success may come from alternative, unexplored mechanisms which have more to do with the pretraining objective than knowledge transfer from upstream corpora.

Dataset	Size (MB)	Classes	Domain	Task
AGNews [Zhang et al., 2015]	27	4	News	topic classification
QQP [Wang et al., 2018]	43	2	Online forum questions	paraphrase detection
Jigsaw Toxicity [Kaggle.com, 2018]	59	6	Wikipedia comments	toxicity detection
MNLI [Williams et al., 2018]	65	3	Diverse	natural language inference
Sentiment140 [Go et al., 2009]	114	5	Tweets	sentiment classification
PAWS [Zhang et al., 2019]	139	2	Wikipedia	paraphrase detection
DBPedia14 [Zhang et al., 2015]	151	14	Wikipedia	topic classification
Discovery [Sileo et al., 2019]	293	174	Web crawl	discourse marker prediction
Yahoo Answertopics [Zhang et al., 2015]	461	10	Online forum answers	topic classification
Amazon Polarity [Zhang et al., 2015]	1427	2	Product reviews	sentiment classification

Table 5.1: The suite of downstream datasets used in this work along with their training set sizes

5.3 Experimental setup

Our experiments center around the ELECTRA model [Clark et al., 2019] and the RoBERTa-base model [Liu et al., 2019b]. On the broadest set of experiments, for which we can only afford to train one model, we employ ELECTRA because it performs better than RoBERTa given comparable compute budgets [Clark et al., 2019]. In particular, we use the small variant of ELECTRA (14 million parameters), which performs similarly to BERT-base on GLUE (difference of $\approx$2 points) while training much faster [Clark et al., 2019]. However, we replicate many of these results on the larger RoBERTa-base model revealing similar results and thus establishing the generality of our findings.

During pretraining, a text sequence is fed into the model with some tokens masked out. While MLM-only models like RoBERTa only have a *generator* network that predicts the content of the masked tokens, ELECTRA has an additional discriminator module that predicts if those predictions were correct. Both the generator and the discriminator networks' parameters are updated simultaneously during pretraining. After pretraining, the generator is discarded and the discriminator is used as an encoder for finetuning on downstream tasks.

We experimented with 10 different downstream datasets (Table 5.1). We chose these datasets in our testbed to span different dataset sizes ranging from 27 megabytes to about 1.4 gigabytes of text in the training split. These datasets are for different tasks such as topic classification, sentiment classification, natural language inference etc., and are created using data sourced from diverse domains. Most of them are multi-class classification tasks except Jigsaw Toxicity which is a multi-label classification task, and Sentiment140 which is modeled as a regression task. For finetuning a pretrained model on any dataset, we passed the input through the model, took the vector representation of the CLS token in the final layer, and passed it through a classification head with one hidden layer to get the output.

5.4 Self-pretraining Performance

In our first set of experiments, we compare self-pretraining's performance with other pretraining techniques. For each dataset, we pretrain an ELECTRA model on text from its training split and then finetune it on the same training data using the associated labels. To create a pretraining corpus from a downstream dataset, we concatenate the input text from each of the examples, assembling them in random order. We evaluate the performance of each finetuned model on the corresponding dataset's test split. For QQP and MNLI we just use the validation split because test set labels are private. For all datasets, we evaluate performance by accuracy, except for Sentiment140 and Jigsaw Toxicity, for which we use Pearson correlation and micro-averaged AUC scores, respectively (these are not multi-class classification problems).

Notably, all self-pretrained models deliver significant performance boosts on their respective datasets (Table 5.2), and over half of them perform even better than the off-the-shelf model. We measured a model's *benefit* as the performance boost that it achieves over a randomly initialized model, divided by the boost achieved by the off-the-shelf ELECTRA model against the same baseline. The average benefit of self-pretraining across all datasets is 103.70%. We do not see a clear correlation between the size of the dataset and the performance of self-pretraining. For example, the highest benefit of 131.33% is achieved for the smallest dataset (AGNews), which is merely 27MB in size, while the minimum benefit is achieved on the Discovery dataset, which is the third largest dataset measuring 293MB. For each downstream dataset, we also pretrain a model on a randomly sampled subset of Wikipedia of the same size as the dataset's training corpus, and finetune it on the downstream task. This approach (called WikiSub) provides a size-adjusted comparision between using separate upstream data vs the downstream data for pretraining. We see that self-pretraining performs better than WikiSub in majority of cases and when it performs worse (MNLI and Discovery datasets), the performance gap is much smaller than the gap between offshelf and self-pretrained models (Table 5.2).

We also evaluated the alternate pretraining technique *TAPT* as described in Gururangan et al. [2020]. In this technique, we take the off-the-shelf ELECTRA model, which has already been pretrained on the upstream BookWiki corpus, and further pretrain it on the downstream dataset for 100 epochs. Self-pretraining outperforms TAPT on 6 datasets, notably including the two datasets where it outperformed the off-the-shelf models by the greatest benefit margin - *AGNews* and *Yahoo Answertopics*. Interestingly, TAPT performs worse than off-the-shelf model on the same 3 datasets where self-pretraining performs worse than off-the-shelf model (except Sentiment140). None of the three pretraining approaches seem to be uniformly better than any other.

Finally, we also evaluate the self-pretrained models on the GLUE benchmark and report results on the dev set [1]. The performance of the models on their pretraining dataset does not correlate strongly with its GLUE score. The GLUE score also does not monotonically go up with increasing dataset size, indicating that the data domain makes some difference. For example, the Amazon Polarity corpus scores just 66.14 on GLUE despite being about 1.4GB in size, while AGNews which is 27MB in size, scores 74.30. The highest GLUE score is achieved by pretraining on Yahoo Answertopics.

[1]Following Clark et al. [2019] we exclude the WNLI task from the results.

Dataset	Size(MB)	RandInit	SelfPretrain	Offshelf	Benefit%	WikiSub	TAPT	GLUE
AGNews	27	91.75	94.34	93.72	131.33	93.51	94.07	74.30
QQP	43	82.93	90.66	90.34	104.34	89.16	90.64	75.43
Jigsaw Toxicity	59	97.83	98.49	98.53	94.99	98.35	98.48	76.65
MNLI	65	65.49	78.39	82.29	76.77	78.64	79.26	78.28
Sentiment140	114	63.75	67.04	66.95	102.91	65.52	65.65	72.67
PAWS	139	50.00	97.53	97.30	100.49	97.42	97.85	74.65
DBPedia14	151	98.59	99.22	99.11	121.17	99.18	99.23	70.38
Discovery	293	17.00	22.38	24.55	71.22	22.47	23.58	77.26
Yahoo Answertopics	461	61.94	65.26	64.55	127.31	64.37	65.05	79.53
Amazon Polarity	1427	93.86	96.27	96.13	106.49	95.82	96.16	66.14

Table 5.2: Performance of ELECTRA-small models pretrained with different techniques on various downstream datasets and on the GLUE benchmark (dev set). For reference, a randomly initialized model scores 53.20 and the off-the-shelf model scores 79.43 on GLUE.

5.5 Cross Dataset Finetuning

In this set of experiments, we investigated if the models pretrained on a dataset are only useful for that specific task, or are they useful across the whole spectrum of tasks that we consider. We took each model pretrained on a dataset in our testbed and finetuned and evaluated it on all other datasets in the testbed. The performance benefits provided in all cases are shown as a heatmap in Figure 5.2.

We found that for almost all downstream datasets, pretraining on any other dataset provides significant advantage (Figure 5.2). In most cases, pretraining on the downstream dataset itself performs the best. Among datasets where self-pretraining performs better than off-the-shelf model (i.e. the diagonal entry is greater than 1), pretraining on datasets of larger size does not help further. However, for the datasets where self-pretraining's benefit is much less than 100% (i.e. MNLI and Discovery), pretraining on a larger dataset (e.g., Yahoo Answertopics) performs better than self-pretraining.

Among all the pretrained models, a few models perform consistently good or bad across different downstream datasets (Figure 5.2). For example, the model pretrained on Yahoo Answertopics gets the highest average score of 0.90 across all datasets, while the PAWS-pretrained model gives the lowest aggregate score of 0.64. Similarly, there are downstream datasets that are benefited consistently by either a large or a small margin by pretraining on different datasets. For example, performance on QQP and PAWS receives huge boosts by pretraining on most datasets. In contrast, performance on sentiment140 is mostly low , even dropping below 20% for 3 pretrained models.

We perform an ablation to investigate that given a fixed dataset to finetune on, is it better to pretrain on the *exact* same data (i.e., using the same set of inputs), or is it better to pretrain on different data with an identical distribution. To test this hypothesis, we divided the training splits of the downstream datasets randomly into two equal subsets (denoted as A and B). We pretrained one model on each subset and then finetuned them on both subsets separately. The validation and test sets used for finetuning are the same as in the original dataset.

We do not see any consistent benefits with pretraining and finetuning on the same dataset

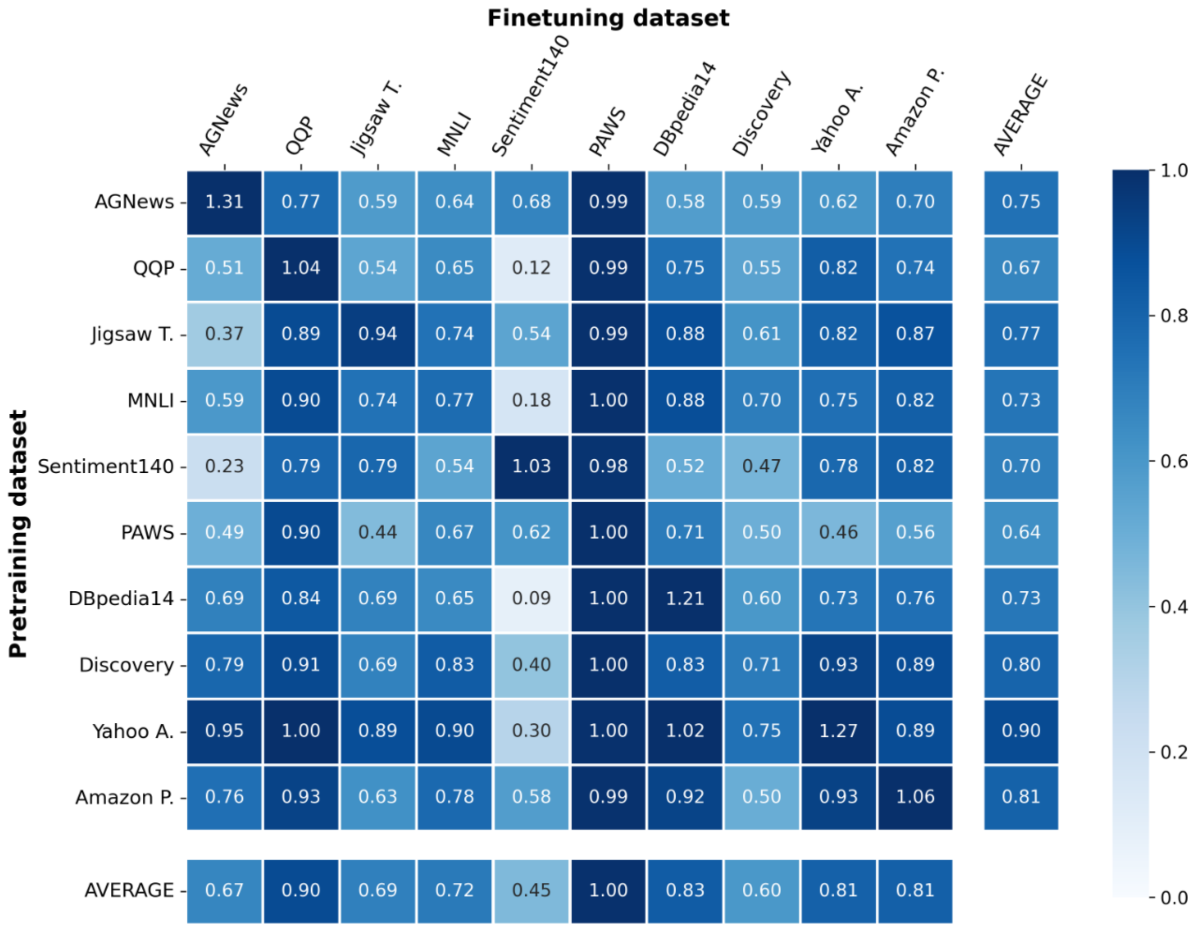


Figure 5.2: Performance benefits of models pretrained on each dataset, upon finetuning on each downstream dataset. Each value is the ratio of performance gains achieved by model pretrained on the row's dataset vs off-the-shelf model, relative to random initialization, upon finetuning on the column's dataset.

(Table 5.3). Instead, we found consistent patterns where models pretrained on one split (either A or B) outperformed models pretrained on the other, irrespective of the split used for finetuning. This suggests that the pretraining data has greater influence on the final performance than the finetuning data. Additionally, we observe that finetuning the superior pretrained model, using the downstream split other than the one used for pretraining, performs the best, suggesting overall exposure to more data helps.

MNLI			**QQP**			**Discovery**			**Yahoo Answertopics**		
	A	B		A	B		A	B		A	B
A	76.00	**76.42**	A	84.28	84.79	A	18.78	18.61	A	64.18	**64.34**
B	75.93	75.05	B	**88.73**	88.41	B	**19.99**	19.98	B	64.09	64.18

Table 5.3: Performance when splitting the dataset into two equal-sized subsets A and B and then pretraining on one (row) and finetuning on another (column)

5.6 Difference in Outputs of Self-pretrained and Off-the-shelf Models

Since self-pretrained models and off-the-shelf models perform similarly in terms of classification accuracy, a natural question to ask is: *do these models make errors on the same set of inputs?* To answer this question, we investigate the difference in predictions made by models pretrained with different strategies across all multi-class classification tasks. In particular, given model f_A and f_B, we compute *error inconsistency*, defined as follows:

$$\frac{\sum_{i=1}^{n}\left(\mathbb{1}\left[f_A(x_i) \neq y_i \wedge f_B(x_i) = y_i\right] + \mathbb{1}\left[f_A(x_i) = y_i \wedge f_B(x_i) \neq y_i\right)\right]}{n}$$

where $\{x_i, y_i\}_{i=1}^{n}$ is the test set. Intuitively, error inconsistency captures the fraction of examples where exactly one model is correct. This definition has been commonly used to estimate diversity in model prediction [Gontijo-Lopes et al., 2022, Geirhos et al., 2020]. Across all the multi-class classification tasks, in addition to computing error inconsistency between self-pretrained and off-the-shelf model, for baseline comparison, we also tabulate error inconsistency between: (i) two independently finetuned versions of a self-pretrained model; and (ii) two independently finetuned versions of the off-the-shelf model.

Compared to error inconsistency between two models with the same pretraining dataset, we observe that models trained with different pretraining datasets have high error inconsistency in predictions (Table 5.4). For models with comparative performance, high error inconsistency highlights the high disagreement in predictions. This demonstrates that while different pretraining datasets produce similarly performing models in terms of overall accuracy, the model predictions are relatively dissimilar. Our observations here align with investigations in vision tasks, where Gontijo-Lopes et al. [2022] observed that models trained with different pretraining datasets produced uncorrelated errors.

Since different pretraining datasets produce models with uncorrelated errors, we ensemble these models to check if uncorrelated mistakes lead to a correct prediction. When the models make different predictions, in particular, when one model is correct and another is incorrect, the ensemble prediction will be dominated by the model with higher confidence in their prediction. As before, we consider ensembles of (i) two independently finetuned self-pretrained models; (ii) two independently finetuned off-the-shelf models; and (iii) a finetuned version, each of the self-pretrained and off-the-shelf models.

We make the following observations: First, as expected we observe that ensembling improves model performance as compared to a single model (Table 5.4). Second, despite hav-

	Ensemble Accuracy			**Error Inconsistency**		
Dataset	2×SelfPretrain	2×Offself	SelfPretrain + Offself	2×SelfPretrain	2×Offself	SelfPretrain + Offself
AGNews	94.66	94.17	94.54	1.76	3.50	4.01
QQP	90.92	90.74	91.63	4.57	5.27	8.91
MNLI	78.51	82.37	82.31	6.94	6.42	14.82
PAWS	97.70	97.45	97.75	0.96	1.30	2.07
DBPedia14	99.28	99.19	99.24	0.38	0.48	0.51
Discovery	22.98	25.25	25.02	7.85	9.18	12.66
Yahoo	65.32	64.69	65.64	5.27	5.49	9.55
Amazon	96.40	96.24	96.51	1.26	1.58	2.48

Table 5.4: Performance of ensemble models of self-pretrained and off-the-shelf models. For ensembling, we aggregate predictions of models after calibration with Temperature Scaling [Guo et al., 2017]. We observe that in most of the datasets, SelfPretrain + Off-the-shelf ensembling does not improve over ensembles of two models with the same pre-training strategy, despite relatively higher error inconsistency of SelfPretrain + Off-the-shelf models.

ing larger error inconsistency, we do not observe any significant improvements in ensembles of self-pretrained and off-the-shelf model as compared to ensembles of two models with the same pretraining strategy (Table 5.4). This is in contrast with findings on vision tasks where Gontijo-Lopes et al. [2022] observed that larger error inconsistency led to larger improvement in ensemble performance.

5.7 Ablations with Other Pretraining Architectures

We conducted our experiments so far with ELECTRA-small architecture because it is faster to pretrain than other popular models, yet delivers good downstream performance [Clark et al., 2019] (e.g. comparable to BERT-base on GLUE benchmark). Here, we conduct experiments with a larger model and a different pretraining objective to test the efficacy of self-pretraining more broadly.

We experiment with the RoBERTa model which uses the masked language modeling objective, rather than ELECTRA's objective. We use the RoBERTa-base architecture, which has a much larger parameter count of 110 million, compared to ELECTRA-small's 14 million. Due to resource constraints, we pretrained the RoBERTa models for fewer iterations as in Warstadt et al. [2020]. We pretrain a RoBERTa-base model on the BookWiki corpus for the same number of iterations. Our results show that self-pretraining performs comparably to pretraining on BookWiki corpus, delivering over 85% of pretraining benefit on 9 out of 10 datasets, and outperforming the model pretrained on BookWiki corpus (Table 5.5) on 5 datasets.

Dataset	**RandInit**	**SelfPretrain**	**BookWiki**	**Benefit %**	**TAPT**
AGNews	91.91	94.28	94.22	102.27	94.07
QQP	76.50	88.68	90.18	89.05	90.64
Jigsaw Toxicity	97.32	97.72	98.03	56.02	98.48
MNLI	31.82	75.12	80.90	88.23	79.26
Sentiment140	56.68	68.55	60.19	338.26	65.65
PAWS	50.00	97.34	97.08	100.55	97.85
DBPedia14	98.57	99.21	99.24	95.98	99.23
Discovery	17.36	25.85	26.30	94.91	23.58
Yahoo Answertopics	61.11	65.96	64.58	139.80	65.05
Amazon Polarity	89.02	96.68	96.11	108.13	96.16

Table 5.5: Performance of RoBERTa-base models pretrained with different techniques on downstream datasets.

5.8 Performance on structured prediction and commonsense NLI

While the bulk of our experiments were on a variety of classification tasks, we also experiment with some tasks beyond simple classification. We experiment with three types of tasks: (i) span based question answering, (ii) named entity recognition (NER), and (iii) grounded commonsense inference. For question answering we use the SQuAD dataset [Rajpurkar et al., 2016] (v1.1) and report the F1-score. For NER, we use the CONLL-2012 NER task which uses annotations from Ontonotes v5.0 [Weischedel et al., 2013] involving 18 kinds of named entities. To measure performance, we use the overall F1 score. We use the seqeval library for evaluation (`https://github.com/chakki-works/seqeval`). We include SWAG [Zellers et al., 2018] and HellaSwag [Zellers et al., 2019] for multiple-choice sentence completion.

For Electra-small models, we see that for each of these datasets self-pretraining achieves more than 70% pretraining benefit, and for Roberta-base model the benefit is 40-80% (Table 6). Even for the SWAG and HellaSwag datasets, which are designed to use rely on *commonsense inference* of pretrained models, we see performance boosts by pretraining using only the task's training set.

Datasets	**Size(MB)**	**ELECTRA-small**				**Roberta-base**			
		RI	SP	OS	Benefit%	RI	SP	BW	Benefit%
SQuAD	19	15.82	63.01	75.96	78.47	14.93	67.23	81.89	78.11
SWAG	22	27.55	60.56	73.76	71.43	27.95	45.18	70.37	40.62
HellaSwag	30	29.27	39.14	42.91	72.36	24.53	31.03	34.28	66.67
CONLL-2012	6.4	54.49	75.66	82.65	75.18	63.65	72.64	86.25	39.78

Table 5.6: Performance of ELECTRA and Roberta models pretrained with different techniques. RI: random initialization, SP: self-pretraining, OS: off-the-shelf; BW: pretrained on BookWiki by us.

5.9 Conclusion and Future Work

In this work, we showed that pretraining models only on text from the downstream dataset performs comparably to pretraining on a huge upstream corpus for a wide variety of datasets. The errors made by such *self-pretrained* models on the downstream tasks are significantly different from the ones made by the *off-the-shelf* models pretrained on upstream corpora. Our results suggest that the importance of learning from surplus upstream data for improving downstream task performance may have been overestimated. Crucially, our experiments also do not show that upstream data does not help at all or knowledge transfer does not occur, but simply questions to what extent it is responsible for downstream gains. For example, the impressive zero-shot performance very large language models such as GPT-3 [Brown et al., 2020] clearly suggests knowledge transfer is involved. One direction of future work would be to investigate how the performance of self-pretraining compares of pretraining on upstream corpora as the model sizes go up by orders of magnitude.

We found that the quantity and quality of data required for pretraining to provide significant benefit (over a randomly initialized model trained only with a supervised loss) is quite low. Downstream datasets which are tiny in comparison to typical upstream corpora, still function as useful pretraining corpora for getting performance gains across a wide range of datasets.

Since self-pretraining does not involve any upstream corpus, it prevents exposure of the model to potentially undesirable contents in the large upstream corpus, while still delivering large performance benefits. Research has demonstrated the negative influence of web-sourced pretraining corpora on models, such as generating toxic language [Gehman et al., 2020] or reflecting racial biases in predictions [Ahn and Oh, 2021]. For use cases that require avoding such issues, self-pretraning can provide a viable alternative to standard pretraining. In future work, we hope to compare how self-pretrained models and off-the-shelf models perform on these negative measures such as toxicity and social biases.

Chapter 6

USB: A Unified Summarization Benchmark across tasks and domains

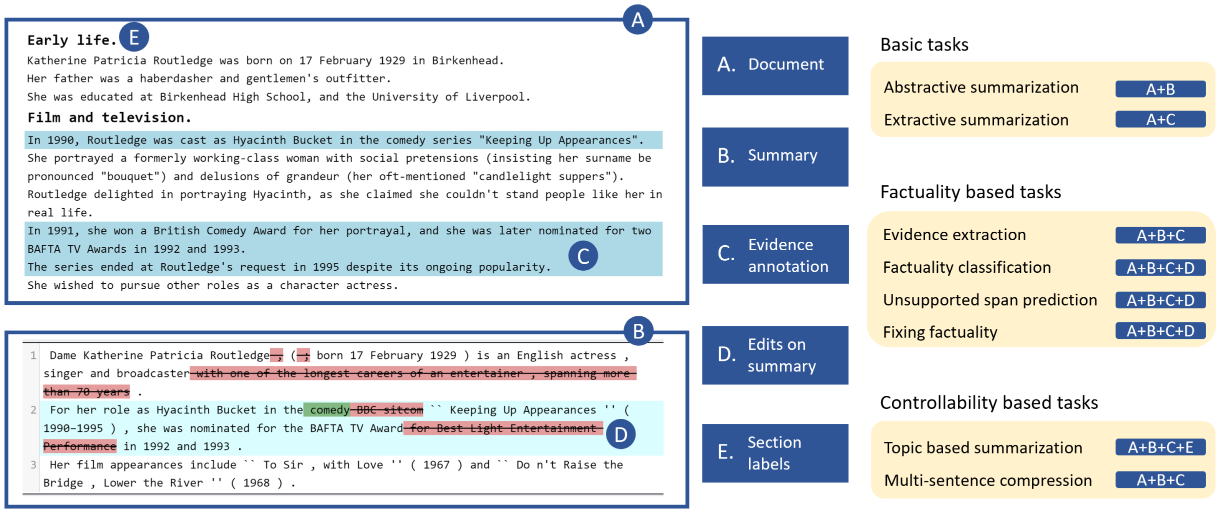
Factual errors in model-generated summaries are a common occurrence and have been widely acknowledged [See et al., 2017, Maynez et al., 2020, Fabbri et al., 2022]. In Chapter 2, we also observed that when generating summaries of doctor-patient visits, even the best performing method produced false statements in about 10% of generated sentences. Compared to the task of summary generation, tasks related to ensuring factual accuracy such as such as editing summaries to fix factual errors, and extracting evidence for its claims, have received lesser attention in literature. Although a small number of labeled datasets exist to train models for these tasks, they use artificially labeled data. These approaches include generating synthetic factual errors using rules [Cao et al., 2020], or hallucinations using text infilling [Balachandran et al., 2022]. Synthetically generated factual errors may not be diverse enough to capture the breadth of errors made by real world models [Goyal and Durrett, 2021]. Similarly, for the task of extracting evidence for summaries, heuristics have been used to create artificial gold labels [Chen and Bansal, 2018, Ernst et al., 2021]. There is a need for a comprehensive summarization benchmark which comprises of human-labeled datasets for tasks related to fact-checking summaries.

In this chapter, we introduce a Wikipedia-derived benchmark, complemented by a rich set of crowd-sourced annotations, that supports 8 interrelated tasks: (i) extractive summarization; (ii) abstractive summarization; (iii) topic-based summarization; (iv) compressing selected sentences into a one-line summary; (v) surfacing evidence for a summary sentence; (vi) predicting the factual accuracy of a summary sentence; (vii) identifying unsubstantiated spans in a summary sentence; (viii) correcting factual errors in summaries. We compare various methods on this benchmark and discover that on multiple tasks, moderately-sized fine-tuned models consistently outperform much larger few-shot prompted language models. For factuality-related tasks, we also evaluate existing heuristics to create training data and find that training on them results in worse performance than training on $20\times$ less human-labeled data. Our articles draw from 6 domains, facilitating cross-domain analysis. We found that for tasks related to fact-checking, trained models perform better in out-of-domain settings, whereas for tasks focusing on summary generation, training specifically on data from the target domain even if limited, is better. [1]

[1]The dataset can be downloaded from `https://github.com/kukrishna/usb`

6.1 Introduction

Numerous summarization benchmarks have been proposed to facilitate the development of summarization methods [Nallapati et al., 2016, Narayan et al., 2018b, Wang and Ling, 2016, Gliwa et al., 2019]. However, the majority of previous work has primarily focused on evaluating the models' ability to generate summaries similar to reference summaries, neglecting key auxiliary properties of text summarization systems. Recent research has highlighted the importance of addressing additional aspects in text summarization. These aspects include the ability to steer summaries by controlling its focus on a topic or on specific parts of the source text [Gehrmann et al., 2019]. Furthermore, there is an increasing emphasis on ensuring *factual correctness* and implementing mechanisms to eliminate factual errors from model outputs Scialom et al. [2021], Balachandran et al. [2022]. Similarly, to foster trust in the outputs, it is desirable for summarization systems to present evidence from sources that corroborate the generated summaries. As models have improved in generating coherent and readable summaries [Goyal et al., 2022], these auxiliary considerations have gained importance. Aligning summaries with user requirements and ensuring sufficient factual support are critical frontiers in summarization research. The current summarization benchmarks fail to provide a comprehensive evaluation of model capabilities across various summarization tasks, encompassing properties such as factuality and controllability.



Early life.
Katherine Patricia Routledge was born on 17 February 1929 in Birkenhead.
Her father was a haberdasher and gentlemen's outfitter.
She was educated at Birkenhead High School, and the University of Liverpool.
Film and television.
In 1990, Routledge was cast as Hyacinth Bucket in the comedy series "Keeping Up Appearances".
She portrayed a formerly working-class woman with social pretensions (insisting her surname be pronounced "bouquet") and delusions of grandeur (her oft-mentioned "candlelight suppers").
Routledge delighted in portraying Hyacinth, as she claimed she couldn't stand people like her in real life.
In 1991, she won a British Comedy Award for her portrayal, and she was later nominated for two BAFTA TV Awards in 1992 and 1993.
The series ended at Routledge's request in 1995 despite its ongoing popularity.
She wished to pursue other roles as a character actress.

1 Dame Katherine Patricia Routledge~~ ,~~ (~~;~~ born 17 February 1929) is an English actress , singer and broadcaster ~~with one of the longest careers of an entertainer , spanning more than 70 years~~ .
2 For her role as Hyacinth Bucket in the comedy ~~BBC sitcom~~ `` Keeping Up Appearances '' (1990–1995) , she was nominated for the BAFTA TV Award ~~for Best Light Entertainment Performance~~ in 1992 and 1993 .
3 Her film appearances include `` To Sir , with Love '' ( 1967 ) and `` Do n't Raise the Bridge , Lower the River '' (1968) .

A
B
C
D
E
A. Document
B. Summary
C. Evidence annotation
D. Edits on summary
E. Section labels
Basic tasks
Abstractive summarization A+B
Extractive summarization A+C
Factuality based tasks
Evidence extraction A+B+C
Factuality classification A+B+C+D
Unsupported span prediction A+B+C+D
Fixing factuality A+B+C+D
Controllability based tasks
Topic based summarization A+B+C+E
Multi-sentence compression A+B+C

Figure 6.1: A schematic of our dataset, annotations, and the supported tasks. The example shown (abridged) displays the edits made by a human annotator on the initial candidate summary (deletions in red with strike-through; additions in green). Every summary sentence is supported by one or more evidence sentences highlighted in blue.

In this work, we introduce USB, a comprehensive benchmark for text summarization that supports eight auxiliary tasks. The benchmark includes labeled datasets with high-quality human annotations collected from diverse documents across six domains. To create the benchmark, we sampled Wikipedia articles from various categories, such as people, organizations, and events. We utilized the introductory section of the articles as a reference summary and the remaining content as the source text, resulting in imperfect source-"summary" pairs. Human annotators then searched for evidence to support each summary sentence. If evidence was lacking, corresponding spans or entire sentences were removed. Whenever conflicting evidence was encountered, the summary was revised with minimal edits to align with the available evidence. The resulting annotations can be repurposed to create labeled datasets for 8 useful tasks (Figure 6.1).

We offer the first human-labeled training datasets for various summarization tasks, including evidence extraction and identifying spans in summaries without supporting evidence. These datasets enable the training and evaluation of models specifically for these crucial aspects. We benchmark the performance of several models such as instruction-tuned encoder-decoder models and LLMs on our tasks, including both fine-tuning and well as few-shot prompting based approaches. Notably, we found that fine-tuning even small models (fewer than a billion parameters) substantially outperforms few-shot prompting of much larger open-source and private large language models.

Prior efforts have relied on heuristics to generate synthetic training data for certain tasks included in our benchmark. For instance, a common heuristic employed is lexical overlap to identify and extract supporting evidence [Chen and Bansal, 2018]. Similarly, artificial factual errors have been introduced into summaries to train models for factuality classification or correction [Kryściński et al., 2020, Balachandran et al., 2022]. Although such automatic approaches enable easy creation of large training datasets, heuristically derived annotations are typically noisy compared to human annotations. Our findings demonstrate that models trained on minimal amount of human-labeled data outperform those trained on heuristically generated labeled datasets, even when the latter are 20$\times$ larger.

A common challenge to real-world adoption of models is their use in resource-poor domains where one does not have access to abundant labeled training data. We compare how the size of available training data matters vis-a-vis its domain for different summarization tasks. We found that for tasks related to factual correctness of summaries, the amount of training data matters more than its domain; but for other tasks having domain-specific training data matters more. Our benchmark is explicitly segmented into 6 domains based on Wikipedia categories, and hence provides a natural test-bed for such domain transfer studies.

Summary of contributions:

- Multi-domain benchmark for training and evaluating models on 8 different tasks dealing with some critical but understudied aspects of text summarization.
- Comprehensive evaluation of models and training strategies, including fine-tuning, few-shot prompting, and multi-task training.
- Comparison of relative value of training data labels generated by humans versus heuristics, showing that for multiple tasks, human annotations yield better models even with 20$\times$ less training data.
- Practical insights about out-of-domain generalization for different tasks, identifying the tasks for which the size of the training data matters more than it being from a specific target domain.

6.2 Dataset Curation

To create the USB benchmark, we first collected a set of manual annotations on Wikipedia articles. We then used the collected annotations to create labeled data for the benchmark tasks. In this section we describe the process of collecting these manual annotations. We consider the text in a Wikipedia article overview (leading) section as the target summary S, and the rest of the article as D. In well-written articles, the overview section (S) provides a broad summary of the article, and the rest of the article (D) provides specifics. Hence, the content in S which highlights parts of D can be effectively considered its summary. However, for S to be a valid summary of D, we need to remove contents within it that mention *new* information that is not present in D and cannot be inferred from it.

We recruited annotators and asked them to execute the following tasks: (1) Find and annotate evidence in D for each summary sentence of S, and; (2) Delete parts of S that are not supported by D. This yields a document-summary pair where the summary is fully supported by the document, and the supporting evidence is explicitly marked. We provide a detailed description of our data creation process below.

Retrieval of Wikipedia articles We downloaded the Wikipedia English articles dump from 1 July 2022. We extracted the articles from this corpus using the Wikiextractor tool. [2] We dropped tables and lists during extraction, but retained section headers. We used a set of category filters to retrieve pages about specific types of entities which helps us in creating a dataset with diverse domains. We manually filtered domains to select those in which articles generally had a substantial part of S supported by evidence present in D. We retrieved articles for the following 6 domains: biographies, companies, schools, newspapers, landmarks, and disasters.

Selecting documents for annotation Our heuristic is to assume that the overview section of a Wikipedia article will feature a significant amount of overlap with the remaining part which would be retained after the annotators remove non-overlapping parts. To derive a good document-summary pair from an article, there should ideally be a large amount of overlap between the overview part and remaining article. Otherwise, after human annotation (to remove parts of the summary unsupported by the corresponding document) one would be left with little text in the summary.

Given an article, with the overview section represented by S and the remaining part represented by D, we broke the summary into sentences $s_1 s_2 s_3 ... s_n$ using Spacy[3]. We calculated how many of the summary sentences have at least one entity which is also present in D. For this step, we automatically marked entities in S and D by considering all the words with internal hyperlinks to other Wikipedia pages as entities. If two hyperlinked words pointed to the same page, they were considered the same entity. For annotation, we only retained articles that have more than 75% of sentences in S with at least one entity overlapping with D. We also controlled for summary length by discarding any article where S has fewer than 4 or more than 12 sentences.

Flagging entity overlaps to help annotators find evidence To help annotators find evidence supporting any given summary sentence, our interface highlights entities present in that sentence

[2] https://github.com/attardi/wikiextractor
[3] https://spacy.io

and also in the source document, with a different color for each entity. To maximize the number of entities detected, we took a union of entities extracted using Wikipedia's hyperlinks, Spacy and DBpedia. [4]

Selection and monitoring of Mechanical Turk Workers We ran a qualification task on Mechanical Turk, tasking workers with annotating one document-summary pair according to the provided instructions. To take this qualifier, we required workers have a HIT approval rate $> 95\%$, and have more than 1000 approved HITS. Each worker was allowed to take the qualification task only once. All workers were given the same document-summary pair for annotation. A total of 174 workers took the qualification task. Out of these, 28 workers were approved by checking their annotation quality manually. The approved workers were then permitted to work on the *main* task where they were asked to annotate different document-summary pairs. Each pair was annotated by exactly one worker. After 300 annotations for the main task, we analyzed the annotation quality of the responses again. For many approved workers, the annotation quality on the main task was significantly worse than the qualification task, and hence we restricted the worker set to only 3 workers whose annotation quality was much better than the rest (hereafter referred to as *primary* workers). The remaining annotations were done by these workers, and a total of 1988 document-summary pairs were annotated.

Verifying annotations Due to the complexity of the annotation task, evidence has not been annotated in some parts in the summaries after the first round. To address this, we trained a model to predict unsupported spans in summaries. Specifically, we trained models that accept an initial summary sentence s and the evidence annotated by the workers as the input, and then predict which spans in s were deleted by the annotator to in their submitted version s'. We applied this model to the summary sentences submitted by annotators to predict unsupported spans in them. We fine-tuned Flan-T5 XL [Chung et al., 2022] for this task. We divided the set of document-summary pairs annotated by our primary workers into two halves, trained a model on each half, and used it to predict the unsupported spans in the other half. We used one of these models for prediction on the remaining document-summary pairs submitted by other workers. Using these model predictions, we selected around 20% of the total summary sentences most likely to contain unsupported spans, and flagged them for verification. This included about 15% of the sentences annotated by primary workers and 45% of sentences annotated by other workers, which aligns with our manual inspection of quality of the workers' annotations. We then designed a slightly modified interface for the verification task, where summary sentences have highlights showing potentially unsupported content, and the workers can select additional evidence or edit the summary as before. After incorporating the changes made in this verification round, we arrived at the final version of the annotated corpus.

6.3 Task Definitions

We derived labeled datasets for tasks using the collected annotations. The resulting benchmark consists of the following 8 tasks:

[4] https://www.dbpedia-spotlight.org

Extractive Summarization (EXT): Given the full document as input, extract all important sentences that it contains. We define the ideal "reference" extractive summary as the set of all source sentences marked as evidence for the summary.

Abstractive Summarization (ABS): Generate a multi-sentence summary of the source document by not just simply selecting important content, but also rephrasing it for succinctness and coherence. The ground truth is the full-length summary created after the annotators' edits.

Factuality Classification (FAC): Predict if a summary sentence is factually correct and sufficiently supported by the information present in the source. We create labeled data by assigning *non-factual* and *factual* labels to the before and after versions of each edited summary sentence, with the marked evidence as source context fed in the input.

Fixing Factuality (FIX): Given a factually incorrect summary sentence, edit it to make it factually correct, with reference to the source text. We create annotations using pre-edited summary sentence and the marked evidence as the input, and the post-edited sentence as the target.

Topic-based Summarization (TOPIC): Given the source article and a topic, the task is to generate a summary for a given topic from a source article. We use Wikipedia section headers as topics and select summary sentences from our labeled dataset that have evidence from a single section only. These sentences act as target summaries, while the full document and section header serve as input.

Multi-sentence Compression (COMP): Given a cluster of sentences from the source document, generate a single sentence summary that incorporates information from all of them. We create labeled data for this by using each summary sentence as a target and its marked evidence as the input.

Evidence Extraction (EVEXT): Given a source document and a summary sentence, identify a minimal set of source sentences which collectively provide supporting evidence for all facts present in that summary sentence. The labeled data consists of each summary sentence and the full source document as input, and the evidence links marked by annotators as the ground truth.

Unsupported Span Prediction (UNSUP): Given a summary sentence and a set of sentences from the source providing evidence, predict spans in the summary which are not supported by the evidence. To create labeled data, we select those summary sentences where annotators only made deletions (no additions or replacements). The input is the pre-edit summary sentence and the marked evidence, and the gold target is the set of spans that were deleted from the summary by annotators.

6.4 Dataset Overview and Statistics

The USB is a benchmark comprising 6 different domains with a varying number of instances in each (Table 6.1). We use a 40:20:40 split for train, validation and test set size for each domain, except the *landmarks* and *newspapers* domains due to small size. Articles from these two domains are kept as challenging test sets to measure the out-of-domain generalization. Length distributions of source documents and their summaries are shown in Figure 6 in the Appendix.

Domain	**Count**	**Domain**	**Count**
Biographies	1514	Companies	97
Schools	150	Landmarks	50
Disasters	145	Newspapers	32

Table 6.1: Number of annotated documents in various domain splits of our benchmark. Total 1988 documents across all domains.

Both exhibit long-tail distributions with lengthy sequences — about 32% of source documents have more than 2000 words and 10% of summaries have more than 200 words. We also find that 27% of summary sentences correspond to 4 or more marked evidence sentences (Figure 6 in the Appendix). This suggests a high degree of abstractiveness, because information needs to be combined from many source sentences and expressed in a single sentence. Annotators deleted about 22% of the words on average from the initial summary presented to them, while adding about 2% new words.

6.5 Benchmarking Different Models

Model	**COMP**	**EVEXT**	**EXT**	**FAC**	**FIX**	**ABS**	**TOPIC**	**UNSUP**
Metric$\rightarrow$	Rouge	F1	AUC	AUC	ExactMatch	Rouge	Rouge	F1
Fine-tuned models								
RoBERTa-Large	-	71.01	84.06	92.69	-	-	-	49.21
T5-Large	41.97	77.22	87.00	94.89	31.26	33.44	23.81	51.71
Flan-T5-Large	43.23	77.71	**87.99**	95.15	32.94	32.05	23.62	58.57
Flan-T5-XL	**44.87**	**79.23**	87.81	95.30	35.10	**32.69**	**24.26**	**64.94**
Flan-T5-XL (multitask)	44.32	76.64	86.44	**95.38**	**36.71**	31.83	23.46	58.51
Few-shot prompted LLMs								
Llama-13B	28.12	5.56	52.90	49.34	8.20	5.51	2.47	0.63
Vicuna-13B	31.35	6.65	52.76	55.28	4.28	5.56	2.84	1.47
GPT-3.5-turbo	33.21	26.78	61.63	60.81	3.29	29.77	14.59	7.80

Table 6.2: Performance of models on different tasks evaluated on the full test dataset. Tasks: **COMP**: Multi-sentence Compression **EVEXT**: Evidence Extraction **FAC**: Factuality Classification **FIX**: Fixing Factuality **ABS**: Abstractive Summarization (of full document) **EXT**: Extractive Summarization **TOPIC**: Topic-based Summarization **UNSUP**: Unsupported Span Prediction

We run a suite of models on all tasks in our benchmark and present the results in Table 6.2. For this set of experiments, we use the consolidated train, validation and test splits, which are a union of the corresponding splits from all domains. For tasks that involve generation of summaries, we use Rouge score Lin [2004a] as the metric. We show geometric mean of the 1,2, and L variants for succinctness (Table 6.2). One exception is the Fixing Factuality task for which we use exact match as the metric. For Unsupported Span Prediction, we measure the F1 score based on BIO tagging format [Sang and Buchholz, 2000]. For the remaining tasks we use standard binary classification metrics.

For the classification/span prediction tasks in our benchmark, we fine-tune Roberta-Large (Liu et al. 2019b; Table 6.2). We recast these as seq2seq tasks and fine-tune variants of T5 models on each of the 8 tasks. We include the original [Raffel et al., 2020a] and the instruction-tuned Flan version [Chung et al., 2022]. T5 Large outperforms Roberta-Large on all the classification/span prediction tasks. Flan-T5 Large performs similarly to T5 Large, though achieves notable gains on Unsupported Span Prediction. Flan-T5 XL consistently improves performance over larger models on almost all tasks, suggesting model size helps (Table 6.2). We also train a multi-task variant of Flan-T5-XL (on all tasks jointly). This mostly retains similar performance as a dedicated XL model trained only on that task, except for Evidence Extraction and Unsupported Span Prediction (Table 6.2).

We run large language models including publicly released models (for research purposes) such as Llama [Touvron et al., 2023] and Vicuna [Chiang et al., 2023], and closed models such as OpenAI's gpt-3.5-turbo[5], i.e., *ChatGPT*. For tasks where the full document is fed as input, we use 4 examples for few-shot prompting owing to limitations in the maximum feasible sequence length for these models, while for the rest we use 16 examples (for details, see the Appendix). ChatGPT consistently outperformed Vicuna-13B and Llama-13B on all tasks except Fixing Factuality. This is because for the Fixing Factuality task, ChatGPT almost always adds new unnecessary information to the summary, even after prompting it to not do that. Compared to ChatGPT, finetuned models perform better on every task. The performance difference is largest for factuality-based tasks such as Unsupported Span Prediction, Evidence Extraction, and Fixing Factuality. ChatGPT does comparatively well on tasks that involve generating summaries.

Since automatic metrics for measuring summary quality like ROUGE [Lin, 2004a] do not necessarily mirror human preference [Cohan and Goharian, 2016], we conducted *human evaluation* of the generated summaries in the COMP, ABS and TOPIC tasks. We collect ratings for summaries generated by Flan-T5 and ChatGPT for 50 randomly selected documents from the test set, using a questionnaire (see the Appendix for more details). We found that on average, ChatGPT's summaries are mostly preferred over Flan-T5-XL model's summaries for all 3 summary generation tasks in terms of relevance and factuality (Table 7.3). This suggests that while fine-tuned models produce summaries closer to the ground truth in the dataset (thus achieving high ROUGE), humans may find the summaries of few-shot prompted LLMs better. For example, in the Topic-based summarization task, while Flan-T5-XL produces summaries with an average length of 46.3 words, ChatGPT generates summaries with an average length of 110.2 words. The ground truth summaries for that task are 36.9 words long on average, which is more closely matched by Flan-T5-XL, but the much longer summaries of ChatGPT are considered better by human annotators as reflected in the human ratings (Table 6.3).

For the Fixing Factuality (FIX) task, we compare the *fixed* summaries generated by Flan-T5-XL and ChatGPT, asking which of them (i) remove more factual errors; (ii) mistakenly remove more correct information; (iii) add more new information; to the initially provided incorrect summary. We found that while ChatGPT removes more factual errors from summaries than Flan-T5, it often does so by removing lots of (even factual) information altogether, and replacing

[5]We used the frozen version codenamed gpt-3.5-turbo-0301

Abstractive Summarization (ABS)		
Question	**Flan-T5**	**GPT-3.5-turbo**
Which of the following summaries is better in terms of effectively summarizing the given full content?	36.4%	39.7%
Which of the following summaries is more factual, accurately representing the information presented in the given full content?	33.8%	33.1%
Multi-sentence Compression(COMP)		
Question	**Flan-T5**	**GPT-3.5-turbo**
Which of the two summaries covers more information touching upon all the highlighted sentences?	27.6%	50.0%
Which of the following summaries is more factual, accurately representing the information presented in the document?	21.1%	38.8%
Topic-based Summarization(TOPIC)		
Question	**Flan-T5**	**GPT-3.5-turbo**
Which of the two summaries is better in terms of effectively summarizing the given topic?	10.0%	85.3%
Which of the two summaries is more related to and exclusive to the given topic?	11.3%	77.3%
Fixing Factuality(FIX)		
Question	**Flan-T5**	**GPT-3.5-turbo**
Which of the two summaries removes more contradictory/unsupported information from the incorrect summary, in reference to the context?	18.0%	38.0%
Which of the two summaries removes more correct information (which is actually well-supported by the context) from the incorrect summary?	3.0%	24.0%
Which of the two summaries adds more new facts compared to the incorrect summary?	2.0%	67.0%

Table 6.3: Win rate for model outputs along different aspects as indicated in human evaluation for different tasks

it with new content to effectively make a new different summary (Table 6.3).

6.6 Out-Of-Domain Performance on Tasks

We next evaluate the performance of fine-tuned models when tested on a domain different from what they were trained on. Our benchmark has training data from 4 domains (i.e. excluding *landmarks* and *newspapers*), with different amounts of labeled data for each. To control for training set size, we randomly subsample annotated documents for each domain to isolate 40, 19, and 38 documents for train, validation and test splits. These sizes of the splits were chosen to match the smallest of the 4 domains i.e. *companies* (Table 6.1).

We train and evaluate Flan-T5 Large models on different domains and plot average scores for all tasks training and test domain pair in Figure 6.2. Models trained on the same domain as the test domain perform best or negligibly close to it. But across test domains, the best out-

Training Domain	**COMP**	**EVEXT**	**EXT**	**FAC**	**FIX**	**ABS**	**TOPIC**	**UNSUP**
Metric→	Rouge	F1	AUC	AUC	ExactMatch	Rouge	Rouge	F1
Companies								
Companies	30.02	61.61	66.36	90.10	11.76	19.30	18.51	7.41
Biographies	-1.83	+2.07	+2.22	-2.80	-5.88	-3.19	-3.62	-7.41
Biographies (full)	+0.67	+6.42	+16.57	+3.84	+20.59	+0.40	-2.92	+46.85
Disasters								
Disasters	31.69	52.89	77.89	77.67	3.03	21.68	16.95	5.80
Biographies	-2.75	+7.15	-9.84	+6.91	-1.01	-5.31	-1.54	-5.80
Biographies (full)	-2.09	+12.52	+6.36	+12.55	+15.15	+0.45	+0.78	+40.22
Schools								
Schools	38.63	62.72	73.92	88.89	3.19	28.89	25.04	2.70
Biographies	-2.09	-0.24	-0.72	-1.84	+2.13	-8.45	-5.67	+0.12
Biographies (full)	+0.69	+5.20	+10.60	+3.44	+26.60	-4.88	-4.51	+37.98

Table 6.4: Out-Of-Domain evaluation of fine-tuned Flan-T5-Large models. In each section of the table, we evaluate 3 variants - A) Model trained on the test domain (Companies, Disasters & Schools), B) Model trained on the Biographies domain (training sets of A and B are subsampled to have equal number of datapoints: train-40, validation-19, test-38), and C) Model variant trained on the full biographies dataset with 607 datapoints for training. Factuality related tasks benefit greatly from an abundance of training data, even if it's not from the target domain.

of-domain trained model has $< 15\%$ performance drop compared to this, showing respectable average out-of-domain performance. Going by the in-domain performance of models trained on equal amounts of data, the *biographies* domain is the easiest and the *disasters* domain is the most difficult. One distinction between the *disasters* domain and others which might explain its difficulty is that it deals with summarizing an event rather than an entity.

For each task in our benchmark, we investigate whether having access to a large training dataset (irrespective of domain) is more important than having training data from the test domain. We use the test splits of 3 domains (*companies*, *disasters*, and *schools*), and on each of them we evaluate 3 different models trained on: (1) The training split of the same domain; (2) The training split of the *biographies* domain, and; (3) The *full* training split of the *biographies* domain (before subsampling) which contains 607 annotated documents. Training on equivalent amounts of data from the test domain and biographies domain leads to comparable or worse performance of the latter (Table 6.4).

However, training on the full train set of the biographies domain achieves much higher performance on many tasks, despite the domain shift (Table 6.4). Gains are most visible on the Unsupported Span Prediction and Fixing Factuality tasks. By contrast, for tasks requiring summary generation, using the large biographies training set often does worse than using the $15\times$ smaller in-domain train set. This might happen because domain-specific knowledge is required to learn the style of summaries to generate for a given domain. On the other hand, factuality related tasks tend to be more objective (e.g., judging factual correctness), and so model skills are transferrable across domains.

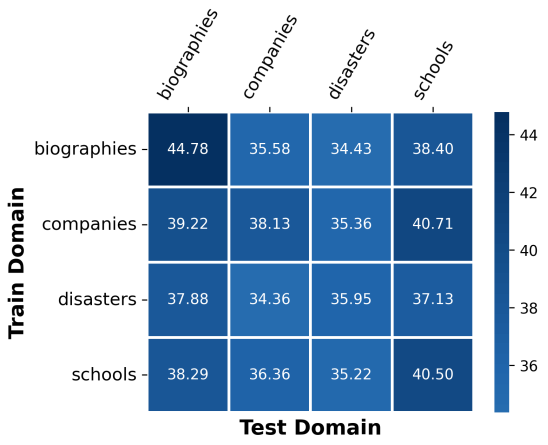


Figure 6.2: Average cross-domain model performance (using Flan-T5-Large) on benchmark tasks. All domains are subsampled to use equal number of annotated documents (train–40, validation–19, test–38).

6.7 Comparison with Heuristic Annotations

For some tasks in our benchmark, past works have used heuristics to create large labeled training data sets as an alternative to collecting manual annotations [Chen and Bansal, 2018, Kryściński et al., 2020, Balachandran et al., 2022]. We use such proposed heuristics to train models and compare them with models trained on high-quality, human annotated data. We conduct experiments on the Evidence Extraction, Factuality Classification and Fixing Factuality tasks. Because the primary advantage of heuristic-generated training sets is their size, we also assess how smaller human-labeled training sets fare in comparison.

For the Evidence Extraction task, we use lexical overlap as a proxy to derive “reference” evidence alignments. For example, we select the source sentence with the highest ROUGE-L score with a summary sentence as its evidence, as outlined in Chen and Bansal [2018]. We also create a training set variant where entity overlap is used instead of ROUGE-L to derive references. Finally, we use *SuperPAL* [Ernst et al., 2021] as an out-of-the-box solution to predict evidence labels for our dataset's summaries, and then use them for model training.

To train models to detect or fix factual errors, we artificially introduce errors into summaries to be used as exemplars. We do this via transformations such as swapping entities, numbers, pronouns, introducing negation, and so on, inspired by prior work [Kryściński et al., 2020]. To generate diverse errors and hallucinations, we follow Balachandran et al. [2022]; we mask parts of the summary out and then use a language model to infill these with (mostly unsupported) information.

We train models for 3 tasks on both heuristically-generated and manually annotated training datasets, and evaluate them on clean human-labeled test sets. Training on human-annotated data performs better than all heuristic-based alternatives across all tasks (Table 6.5). Next, we train models on subsets of the manually annotated datasets with varying sizes and compare their performance on the test sets; this shows how even a little human-annotated data can outperform large amounts of heuristic-generated data for different tasks. For each of the three tasks, the performance achieved using only 5% of the human annotated training set, still outperforms the

Evidence Extraction	
	F1
SuperPAL [Ernst et al., 2021]	53.8
ROUGE [Chen and Bansal, 2018]	40.9
Entity overlap	47.0
Human annotations 100% (N=765)	77.7
Human annotations 5%	70.9
Factuality Classification	
	AUC
FactEdit [Balachandran et al., 2022]	74.6
FactCC [Kryściński et al., 2020]	68.9
Human annotations 100%	95.1
Human annotations 5%	90.4
Fix factuality	
	Exact Match
FactEdit [Balachandran et al., 2022]	1.0
FactCC [Kryściński et al., 2020]	0.8
Human annotations 100%	32.9
Human annotations 5%	11.2

Table 6.5: Comparing use of human annotations vs heuristic annotations for finetuning Flan-T5 Large models. We also report performance when finetuning on 5% of the training set with human annotations.

heuristically labeled full training set (Table 6.5). This highlights the value in collecting manual annotations, even if in small quantities, over using heuristics to generate training data labels.

6.8 Related Work

The tasks in our benchmark have been studied in prior work to varying degrees. The greatest amount of attention has gone to the tasks of extractive summarization [Wong et al., 2008, Kågebäck et al., 2014, Zhang et al., 2016, Nallapati et al., 2017], and abstractive summarization [Liu and Lapata, 2019, Zhang et al., 2020, Raffel et al., 2020a, Lewis et al., 2020b, Goyal et al., 2022]. There exist plenty of datasets for abstractive summarization [Narayan et al., 2018b, See et al., 2017, Kim et al., 2019b, Wang and Ling, 2016]. However, many of them were created heuristically, with "targets" being automatically extracted via rules from documents pulled from the web. This can lead to poor quality reference summaries Bommasani and Cardie [2020], Krishna et al. [2023c], and training on them can yield models prone to generating hallucinations [Nan et al., 2021, Ji et al., 2022]. By contrast, we use manual annotation to ensure that summaries are fully supported by sources, resulting in a high quality abstractive summarization dataset.

Past works for predicting factual correctness of summaries incorporate question-answering models and natural language inference methods [Scialom et al., 2021, Fabbri et al., 2022, Goyal and Durrett, 2021], or use synthetically introduced factual errors [Kryściński et al., 2020] to train models. In contrast, the USB benchmark introduces a high-quality manually annotated dataset for predicting factual correctness. For the task of *editing* summaries to fix factual errors, datasets with both synthetic and model-generated errors have been created [Balachandran et al., 2022, Liu et al., 2022]. The task of unsupported span prediction is akin to detecting hallucinated content in generated summaries, and to the best of our knowledge, no labeled dataset exists for this task.

For extracting evidence for a summary, past works have used lexical overlap based heuristics [Chen and Bansal, 2018, Lebanoff et al., 2019]. A manually annotated dataset for the task was introduced by Ernst et al. [2021], albeit our work provides a substantially larger manually annotated dataset. Similarly, for multi-sentence compression we introduce a much larger manually labeled dataset than prior works [Slobodkin et al., 2022]. Prior research has mostly approached topic based summarization by adopting a predefined set of topics [Krishna and Srinivasan, 2018, Akhtar et al., 2017, Hayashi et al., 2021]. However, we did not restrict the set of topics in our dataset, resulting in a long tail of (potentially challenging) rare topics.

6.9 Conclusion

We introduced the USB benchmark comprising tasks to measure model performance across different text summarization sub-tasks. We showed that fine-tuned smaller models outperform few-shot prompting of much larger LLMs by a large margin on tasks related to appraising the factuality of summaries. We studied how fine-tuned summarization models perform on out-of-domain data, and identified several tasks where the training dataset size is more important than its domain.

Finally, we showed that rather than training models on large volumes of heuristically labeled data, one can get better performance by creating a much smaller ($\approx$ 20$\times$ smaller) manually labeled training set instead. The resultant USB benchmark permits the training of models for useful tasks such as extracting evidence for a summary, correcting factual errors in it, and generating summaries focused on specific topics. Our hope is that this benchmark spurs further research on these tasks and will serve as a barometer for progress in them.

6.10 Limitations

Despite efforts to collect a diverse dataset, the benchmark used in this study may still exhibit certain biases. The sampling process and the selection of Wikipedia articles as the primary data source could introduce inherent biases, potentially affecting the generalizability of the results. These biases may stem from the specific domains or topics covered in the dataset, as well as the way in which Wikipedia articles are written and formatted. The dataset's reliance on Wikipedia articles as the primary source of data might not adequately represent the nuances and challenges encountered in different domains or sources. One prominent example is conversations which are frequently used in summarization research but are not represented in the benchmark. Similarly, a model's ability to detect errors/hallucinations in summaries in the benchmark may not necessarily reflect its ability to detect errors more broadly in summaries generated by a variety of models.

While the benchmark dataset was annotated by human annotators, it is important to acknowledge the possibility of annotation errors or inconsistencies. Despite efforts to ensure high-quality annotations, the presence of errors should be taken into account when interpreting the results. Human annotation is subjective by nature, and different annotators may have varying interpretations in some situations, e.g., deciding whether a fact in the summary requires explicit evidence or should be presumed as common knowledge.

6.11 Ethics Statement

Potential biases: When selecting the pool of annotators on Amazon Mechanical Turk (AMT) for creating the dataset, we required their location to be the United States. This was done since the US has a very large population of native English speakers, which can help in getting high quality annotations. However, this geographical restriction can also lead to biases in the annotation process. For example, it would affect what's considered common knowledge when assessing evidence for summaries. An annotator from the United States would likely consider a person's birth in Los Angeles as evidence of them being from California, because they know Los Angeles is in that state. However, if it were some other city and state in a country unfamiliar to them, they may not make a similar inference.

Compensation for annotators: For the initial qualification task, workers were paid 2 USD. After selecting the qualified workers, for the main annotation task workers were paid 2 to 3 USD per document-summary pair, depending on the number of sentences in the summary and the domain where it came from (we observed that some domains were more difficult). For the second round for verification, we paid annotators between 0.3 to 1.0 USD depending on the number of sentences in the summary which were flagged for verification, which can be as low as 1 sentence. The creation of the entire dataset costed about 6000 USD including platform fees paid to AMT and server hosting costs.

Use of proprietary LLMs: We included the GPT-3.5-turbo large-language-model from OpenAI in our experiments since it has demonstrated excellent performance on diverse NLP tasks in zero-shot and few-shot settings. Unfortunately, OpenAI could discontinue hosting the model in future at which point it may not be possible to reproduce its results on the tasks proposed in this work. For this reason we have also included results with public open-source LLMs like Llama and Vicuna, as these models are publicly available and hence their results can always be reproduced.

Chapter 7

GenAudit: A tool to fix factual errors in model-generated summmries

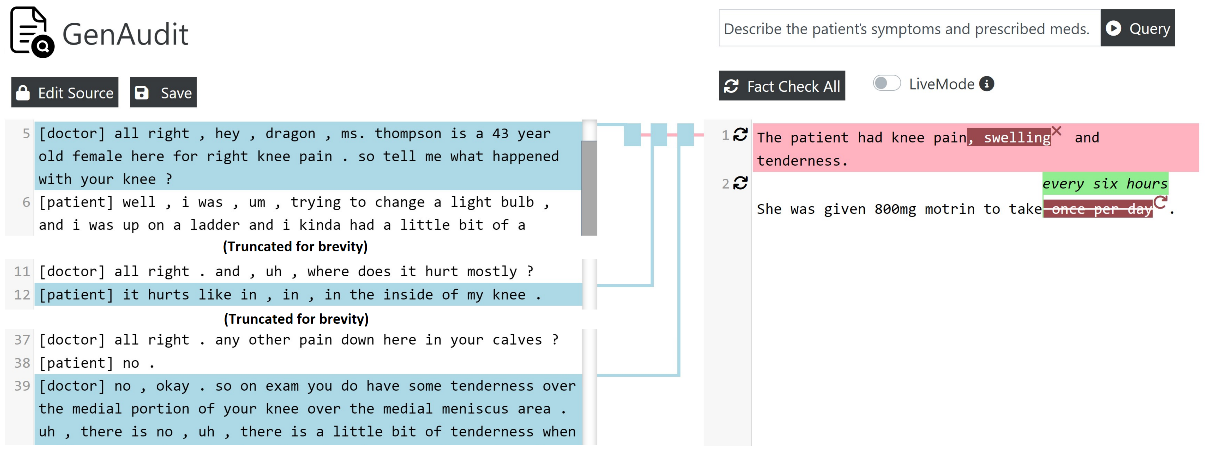
Despite the availability of methods to detect and fix factual errors in summaries (e.g. models trained in Chapter 6), it is unlikely that the resulting summary would be completely error free since the error detectors aren't perfect. Hence, in many high-stakes applications (e.g., healthcare or finance) it becomes necessary for a human to manually verify the summaries. However, this process can be time-consuming, especially if the corresponding source to verify against is long. A tool utilizing trained models for factuality verification can provide valuable assistance to the humans in the fact-checking process. Additionally, the tool should be broadly usable, i.e. it should be able to detect errors in summaries irrespective of what language model is used for generating it, and should work with diverse domains of data such clinical records, news articles, conversation transcripts etc.

In this chapter, we present GENAUDIT — a tool to assist fact-checking LLM responses for document-grounded tasks. GENAUDIT suggests edits to the LLM response by revising or removing claims that are not supported by the reference document, and also presents evidence from the reference for facts that do appear to have support. We train models to execute these tasks, and design an interactive interface to present suggested edits and evidence to users. Comprehensive evaluation by human raters shows that GENAUDIT can detect errors in outputs from 8 different LLMs when summarizing documents from diverse domains. To ensure that most errors are flagged by the system, we propose a method that can increase the error recall while minimizing impact on precision. We release our tool (GENAUDIT) and fact-checking model for public use.[1]

7.1 Introduction

LLMs can produce factually incorrect or unsubstantiated statements [Li et al., 2023a, Min et al., 2023], even when they are explicitly provided relevant context such as documents [Adams et al., 2023, Sadat et al., 2023]. Incidentally, such *document-grounded* generation is often involved in high-stakes usage scenarios where factual correctness is (especially) paramount. For example, a doctor using an LLM to summarize a patient's medical history [Adams et al., 2023, Kanwal

[1]https://genaudit.org



Describe the patient's symptoms and prescribed meds.

5 [doctor] all right , hey , dragon , ms. thompson is a 43 year old female here for right knee pain . so tell me what happened with your knee ?
6 [patient] well , i was , um , trying to change a light bulb , and i was up on a ladder and i kinda had a little bit of a

11 [doctor] all right . and , uh , where does it hurt mostly ?
12 [patient] it hurts like in , in , in the inside of my knee .

37 [doctor] all right . any other pain down here in your calves ?
38 [patient] no .
39 [doctor] no , okay . so on exam you do have some tenderness over the medial portion of your knee over the medial meniscus area . uh , there is no , uh , there is a little bit of tenderness when

1 The patient had knee pain, swelling and tenderness.
2 She was given 800mg motrin to take ~~once per day~~ every six hours.

Figure 7.1: An illustration of GenAudit's user interface and sample predictions. Reference document (a clinical transcript) is on the left and the generated text to be fact-checked is on the right (generated by querying any LLM, but manually entered here for ease of illustration). Spans in the text which are not supported or are contradicted by the reference are highlighted in red, with suggested replacements in green. As the user moves to any line in the generated text, evidence found for all facts in it are highlighted using blue links. Evidence and error predictions shown here are made by a fine-tuned Flan-UL2 model backend.

and Rizzo, 2022] might make an incorrect decision if the generated summary contains errors. Manually verifying LLM outputs in such settings is therefore prudent, but also time-consuming and so undercuts the motivation for using language technologies in the first place. This motivates the need for a system that can *assist* users in efficiently verifying LLM output.

To this end, we introduce GenAudit, a tool for fact-checking LLM responses in document-grounded tasks such as summarization and question answering. Given a document and an LLM-generated output conditioned on the same, GenAudit (i) locates factual errors in the output text and proposes edits to fix them, and (ii) displays evidence to support facts in the (potentially edited) text. The system consists of two components: an interactive interface which presents evidence and edit suggestions for the user to act upon, and a bespoke backend model (fine-tuned LLM) capable of producing edits and identifying evidence. The interface allows the user to make edits to the LLM-generated text, and then observe updated predictions from the fact-checking model. Notably, in addition to supporting the task of fact-checking itself, the interface can also be used as a tool to evaluate and compare different backend fact-checking models, collecting data on human edits to fine-tune better models, and carry out counterfactual testing of fact-checking models by editing source documents.

We designed and evaluated different models to generate the fact-checking predictions for the tool, including fine-tuned and few-shot prompted LLMs. We treat this as a sequence-to-sequence task: Given an input document and a claim sentence, the model is required to simultaneously generate the sentence ids in the document which provide evidence, and a *revised* version of the claim which fixes any factual errors. We used data from the USB benchmark [Krishna et al., 2023a] to train and evaluate models on the fact-checking tasks. We found that fine-tuned open-source LLMs perform better than few-shot prompted ChatGPT and GPT-4 models, at least when evaluated on an in-domain held-out test set.

Ideally, a fact-checking tool would support verifying text produced by any LLM, based on reference documents from any domain. We evaluated GenAudit using 8 different models to summarize documents from 3 different domains. Human annotators were asked to accept or reject edits suggested by the tool, fix errors that were not caught by it, and also to provide feedback on the usefulness of suggested evidence. On average, GenAudit highlighted $\sim 40\%$ of erroneous words in summaries with a precision of $\sim 95\%$.[2] In terms of extracting useful evidence, GenAudit achieved $\sim 91\%$ recall and $\sim 95\%$ precision.

Human evaluations also show that GenAudit can be used to verify summarization outputs in different domains, including clinical conversations, news articles and social media posts. This is despite the fact-checking model being trained only on Wikipedia data. We also evaluated its performance at factuality classification on the SummEdits benchmark [Laban et al., 2023] which consists of data from 10 different niche domains such as legal documents, scientific papers, and emails. GenAudit outperforms alternative fact-checking methods like DAE [Goyal and Durrett, 2021] and QAFactEval [Fabbri et al., 2022] and many LLMs (both open-source and proprietary), with the exceptions of GPT-4 and Gemini-pro.

GenAudit successfully identified errors in outputs from 8 different LLMs including Mistral-7B [Jiang et al., 2023], LLama2-70B [Touvron et al., 2023], Gemini-pro [Team et al., 2023] and GPT-4 [Achiam et al., 2023] in human evaluation. Observed precision ranged between $79-100\%$ while, recall ranged from $23 - 57\%$ for different generation models. Our human evaluation yielded a collection of 702 summaries generated by state-of-the-art models carefully annotated with factual errors; this may be useful for future research on fact-checking LLMs.

The relative trade-off between identifying errors (recall) and making efficient use of expert time (precision) will depend on the particular use-case. We therefore introduce a decoding algorithm for fact-checking models which generate revised/fixed versions of claims, which can increase the recall of error detection with minimal cost in precision. This approach entails intervening at time-steps where the output probabilities fall below a threshold τ to select alternate decoding paths. Varying τ allows us to make more or fewer edits, effectively trading recall against precision. This approach produces a better precision-recall frontier than a baseline of randomly selecting additional words to edit to boost recall.

Our contributions are as follows:

- We present GenAudit, a tool to assist fact-checking LLM outputs in document-grounded tasks. The tool identifies and fixes errors, and highlights evidence for claims.
- We evaluate and release fine-tuned LLMs which serve as backend models for fact-checking; these perform comparably to SOTA proprietary LLMs in few-shot settings.
- We evaluate GenAudit for fact-checking errors in summaries generated by 8 LLMs for documents from 3 domains.
- We present and evaluate a decoding-time method that allows one to improve error detection recall (with some cost in precision).

[2]For reference, $\sim 4\%$ of words in summaries are erroneous, on average.

Input:
You are provided a document and its summary. The summary may potentially contain factual errors. The last sentence of the summary is marked as a claim. Find all sentences in the document providing evidence for the claim, and then revise the claim to remove or replace unsupported facts.
DOCUMENT: SENT0 Micheal Ward SENT1 Early life. SENT2 Micheal Ward was born in Spanish Town, Jamaica on 18 November 1997. SENT3 His mother was 18 years old when he was born. SENT4 He has three sisters. ... SENT17 Ward's breakout year came in 2019, when he starred as Jamie in Netflix's revival and third series of "Top Boy". SENT18 He also appeared in a leading role in the film "Blue Story" in the same year. SENT19 The film received critical acclaim, and Ward won the BAFTA Rising Star Award for his performance. ...
SUMMARY: Micheal Ward (born 18 November 1997) is a Jamaican-British actor and former model.
CLAIM: His films include "Blue Story" (2018) and "The Old Guard" (2020).

Output:
EVIDENCE: SENT18
REVISION: His films include "Blue Story".

Table 7.1: Sample datapoint with input-target formatting from the USB dataset

7.2 Background

We fine-tune LLMs to perform evidence extraction and claim editing, and use them as backend for GENAUDIT. In this section we provide an overview of the training dataset and models we use to power the underlying evidence extraction and factual error correction tasks.

7.2.1 The USB dataset

The USB dataset [Krishna et al., 2023a] is composed of Wikipedia articles, their summaries and (human) annotations on them. The summaries have two versions: (i) An initial version which may have content that is unsupported by the article or contradicted by it, and (ii) An edited version which annotators have created by making minimal edits to the initial version to remove errors. Additionally, each sentence in the edited summary is linked to a minimal set of article sentences that provide sufficient evidence for all facts that it contains.

We format the dataset in a sequence-to-sequence format to use it for fine-tuning LLMs (Table 7.1). The input to the model starts with the task instruction. It is followed by the reference document where each sentence is prefixed by a sentence ID (e.g. SENT1, SENT2...). It is then followed by the summary sentences upto the sentence to be fact-checked (called the *claim*). The sentences preceding the claim are included so that relevant context from it (e.g. coreferences) can be used for better understanding of the claim. Finally, the claim is appended to the input. The target output consists of the two parts. The first part contains a list of sentence ids from the document which provide evidence for the claim, and the second part consists of a revised version of the claim which removes its unsupported information and replaces incorrect facts.

We use a custom split of the USB dataset for training and evaluating our model. We shuffle and divide the entire dataset into train, validation and test splits of size 94%, 3% and 3% of the full dataset. This differs from the original USB splits in two ways. First, the training split is much larger at 94% instead of the original 40%. Second, the training split consists of articles from all

Model	Error Identification			Evidence		
	Recall	Precision	F1	Recall	Precision	F1
Finetuned decoder-only LLMs						
Falcon-7B	69.03	61.54	65.07	59.85	54.23	56.90
Llama2-7B	74.85	39.19	51.44	68.03	68.47	68.25
Mistral-7B	80.53	73.34	76.77	72.25	86.66	78.80
Fine-tuned encoder-decoder LLMs						
Flan-T5-XL	73.01	87.07	79.42	78.90	85.69	82.16
Flan-T5-XXL	**80.38**	84.50	**82.39**	**81.46**	85.96	**83.65**
Flan-UL2	76.47	**87.44**	81.59	80.56	**86.42**	83.39
Few-shot prompted proprietary LLMs						
GPT-3.5-turbo (8shot)	38.79	48.57	43.13	51.79	45.15	48.24
GPT-4 (4shot)	37.98	63.89	47.64	74.42	38.52	50.76

Table 7.2: Performance of different models on the test split of the USB dataset for the tasks of (i) identifying erroneous words, and (ii) highlighting relevant evidence

6 domains in the benchmark, whereas originally 2 of the domains where reserved as challenging OOD examples to occur in only the test set. The motivation for both of these changes is that we want to create a tool which generalizes to other diverse data beyond simply Wikipedia articles, and hence we train on as much and as diverse data as possible.

7.2.2 Reducing memory requirement for training

We aim to fine-tune large models for the fact-checking task, since they are more likely to perform better and generalize to unseen domains due to internal knowledge. However, it is challenging to do so without access to large compute clusters due to memory constraints. The requirement to feed in the entire reference document to the model, which can be thousands of tokens long, further increases memory requirements.

To address the memory constraints, we use low-rank adapters with 4-bit quantization [Dettmers et al., 2023]. Low rank adapters [Hu et al., 2021] reduce the memory requirement during training by reducing the number of trainable parameters, thus reducing gradient computation. To reduce the sequence length in cases where the reference document (Wikipedia article) is too long, we iteratively drop sections in it which are not relevant to any sentence in the summary (i.e. do not provide any evidence for it). We follow this process until the input sequence length is reduced to within a maximum limit, which is kept the same regardless of what model we are fine-tuning. We also use gradient accumulation and gradient checkpointing [Chen et al., 2016] to reduce memory footprint for training. For more details, please see the Appendix.

7.3 Experiments

We use the USB dataset for training/prompting and evaluating 8 different models for two factchecking tasks. For the evidence extraction task, we report precision, recall and F1 score as in a binary

	Error Identification				Replacements	Evidence Extraction			
	BaseRate	Recall	Precision	F1	Accepted%	Recall	Precision	F1	Sufficient%
Aggregate	3.97	40.37	95.04	56.66	78.18	90.83	95.22	92.97	85.98
Summary generation models									
Llama2-7B	4.29	30.21	89.29	45.15	66.67	90.71	96.12	93.33	86.65
Mistral-7B	1.99	23.83	92.00	37.86	40.00	91.43	95.80	93.57	87.59
Falcon-7B	21.84	47.95	97.46	64.28	86.36	84.65	87.08	85.85	76.77
Llama2-70B	3.29	43.38	95.93	59.75	100.00	91.98	92.09	92.04	86.10
Flan-UL2	9.68	34.04	96.04	50.26	80.00	90.18	93.96	92.03	84.16
Gemini-pro	1.80	27.75	78.69	41.03	66.67	91.27	96.98	94.04	86.68
GPT-3.5-turbo	1.10	29.06	89.47	43.87	75.00	92.32	97.23	94.71	88.09
GPT4	2.53	56.77	100.00	72.42	87.50	90.39	97.07	93.61	85.49
Datasets									
XSum	4.54	55.00	98.69	70.64	77.78	94.81	97.92	96.34	92.83
ACIBench	2.73	44.04	90.65	59.28	88.89	87.57	93.30	90.34	80.84
Reddit	4.88	22.52	92.41	36.22	60.00	91.93	95.50	93.68	86.55

Table 7.3: Results from human evaluation of GENAUDIT predictions (using fine-tuned Flan-UL2 backend) on LLM-generated summaries of documents from different datasets.

classification task where given a claim and reference document, each sentence in the reference is classified as relevant evidence or not. To evaluate the model's ability to remove errors, we compare the words removed/replaced by the model vs those removed in the ground truth revision. Given the original claim and a revision, we tokenize each into words and compute the diff between them (using Python's showdiff library). The words in the claim that are removed/replaced in the revision are tagged as incorrect and the remaining words are tagged as correct. We use the ground truth revision in the dataset to compute the ground truth tags, the model-generated revision to compute the predicted tags, and compute the corresponding precision, recall and F1 scores. Notably, it is difficult to compare the replacement text proposed by the model with ground truth replacements automatically, since the underlying text span being replaced must match exactly to make an aligned comparison. This requires human evaluation which we discuss in the next section.

We fine-tune and evaluate 6 different models. These include decoder-only models from the Falcon [Almazrouei et al., 2023], Llama2 [Touvron et al., 2023] and Mistral [Jiang et al., 2023] series, and encoder-decoder Flan-T5 models [Chung et al., 2022, Tay et al., 2022]. Finally, we also use OpenAI's GPT-3.5-turbo and GPT-4 models for the task via few-shot prompting, using 8 and 4 exemplars respectively.

We find that there is large variation in performance of decoder-only LLMs (Table 7.2). Llama2 outperforms Falcon in evidence extraction, but underperforms it in the claim editing task, due to its low precision. Mistral outperforms both Falcon and Llama2 models by a large margin. There is relatively less variation in performance of the three encoder-decoder models. Flan-T5-XXL and Flan-UL2 models perform the best, with the former providing better recall and the latter providing better precision on both tasks. Few-shot prompted GPT models perform worse than all fine-tuned models on both error identification and evidence extraction.

7.4 Human Evaluation

In the previous section we saw that models fine-tuned on the USB dataset perform well when evaluated on its test split. However, this does not imply that they would also perform well when deployed in diverse out-of-domain scenarios. Two types of domain shift can occur here. The first is a change in the domain of reference documents used for fact-checking. USB consisted of Wikipedia articles only, but we would ideally want a finetuned model to work with other document types such as news articles or meeting transcripts. Another sort of domain shift is the specific model generating the content to be fact-checked. USB consists only of claims written by humans, but we would want models to detect and fix errors in content generated by a arbitrary LLMs.

We run experiments to evaluate the performance of GENAUDIT when fact-checking summaries generated by different models for documents sampled from different domains. We include a diverse set of open-source models, including three decoder-only 7B parameter LLMs (Mistral [Jiang et al., 2023], Llama2 [Touvron et al., 2023], Falcon [Almazrouei et al., 2023]), one large 70B parameter model (Llama2), and one encoder-decoder model (Flan-UL2 [Tay et al., 2022]). As proprietary API-based models, we use GPT-3.5-turbo, GPT-4, and Gemini-pro models for summary generation. We then use the Flan-UL2 model finetuned on USB to fact-check the generated summaries.

We select documents for summary generation from the following three datasets.

XSum [Narayan et al., 2018a] A summarization dataset consisting of BBC news articles covering diverse topics and events.

ACI-Bench [Yim et al., 2023] A dataset for summarization of patient visits to the doctor comprising transcripts of doctor-patient encounters.

Reddit-TIFU [Kim et al., 2019b] A dataset consisting of posts from the online discussion forum Reddit in which users narrate personal day-to-day experiences.

We randomly select 30 documents from each of the three datasets for which to generate summaries. For the Reddit-TIFU dataset, we manually filtered out examples containing profanity or sexually explicit content. While generating summaries with open-source models, we decode using top-p nucleus sampling [Holtzman et al., 2019] from the output token distribution with a top-p value of 0.9 and a temperature of 1.0.

We hired annotators via Upwork,[3] and instructed them to evaluate all edits suggested by the fact-checker, accept those that fix legitimate factual errors, and mark incorrect suggestions. Annotators were also instructed to find any missing errors in summaries which were not highlighted by the system, and to fix them by making minimal edits.

To provide feedback on the highlighted evidence, annotators provided binary (relevant/not relevant) feedback for each suggested evidence sentence in the summary. They were also instructed to consider if the highlighted evidence for each summary sentence is sufficient or not; of not, then they should mark additional source sentences which contain the missing evidence. Additionally, we asked annotators to flag incomprehensible summaries, which were then excluded from the analysis. For example, instead of a summary, sometimes Falcon-7B model outputs a continuation of instructions, such as *"when creating a summary, use the information given and*

[3] https://www.upwork.com/

avoid superfluous details." The Appendix includes additional evaluation details.

Results are shown in Table 7.3. We use the metrics described in Section 7.3 for rating suggested errors and evidence generated by GenAudit. On an aggregate level—across all domains and summary generation models—GenAudit identifies erroneous words with high precision (95.04%) and moderate recall (40.37%). Note that achieving high recall is challenging here given the low prevalence of erroneous words (3.97%). With respect to evidence extraction, we observe high precision and recall (95.22% and 90.83%, respectively), suggesting that most evidence sentences highlighted by the model are useful for fact-checking the given claim, and only few evidence sentences are missed (not highlighted) by the model.

The rate of errors in outputs varies considerably across models. Summaries from GPT-3.5-turbo have the lower error rate at 1.10%, while Falcon-7B has the highest error rate of 21.9%. The highest recall and precision for error detection is observed for the latter model. The precision of error detection remains around or above 90% for all models except Gemini-pro. Recall varies widely ($\sim 23 - 57\%$) across different models. The lowest recall is for Mistral-7B (23.83%); for context, its error rate is 1.99%. For evidence extraction, the performance with most models is quite similar with both precision and recall, falling between $\sim 85 - 97\%$. The lowest F1 score is 85.85% for Falcon-7B, and highest is 94.71% for GPT-3.5-turbo.

Among the datasets considered, GenAudit's performance at error identification was best for XSum (news articles), followed by ACIBench (clinical conversations), and finally Reddit/TiFU (social media posts). While the precision stays above 90% on all datasets, the recall ranges from 22.52% on the Reddit dataset to 55.00% on XSum. On the evidence extraction task, GenAudit achieves F1 scores of 90%+ on all three datasets.

While the previously discussed metrics measure success at identifying parts of the text which are incorrect, we also measure the quality of model-generated replacements when they are suggested. The percent of model-suggested replacements accepted was $\sim$78%, on average (Table 7.3), suggesting the quality of generated replacement strings. The percent of generated summary sentences for which the highlighted evidence was sufficient for verification was $\sim$86%, indicating that generated evidence highlights may make fact-checking more efficient.

Using the annotations collected above, we evaluate additional models on the error detection task. Few-shot prompted GPT-4 achieves better recall than Flan-UL2, while other fine-tuned models achieve lower recall (Table 7.4). All models achieve lower precision than Flan-UL2. Edits suggested by GPT-4 add about 8.8% more words to them on average, while other fine-tuned models add a negligible amount, reflecting the tendency of GPT-4 to make substantial changes to text. Finally, we also evaluate the FAVA model trained by Mishra et al. [2024] for factual error correction. We see that the model achieves the lowest recall compared to all the models evaluated, with a slightly higher precision than GPT-4.

Model	Recall	Precision	%Del	%Add
Flan-UL2	40.37	**95.04**	1.69	0.18
Flan-T5-XL	25.75	74.23	1.38	0.12
Mistral-7B	35.08	45.24	3.08	0.16
GPT-4 (4-shot)	**40.68**	28.50	5.67	8.80
FAVA [Mishra et al., 2024]	14.18	31.34	1.80	0.43

Table 7.4: Performance of models fine-tuned by us on the USB dataset, few-shot prompted GPT-4, and the FAVA model [Mishra et al., 2024] at identifying erroneous words in model-generated summaries, along with the percentage of summary words deleted and added by their edits.

7.5 Improving Recall of Error Detection

Users fact-checking LLM outputs using GENAUDIT may give more importance to a higher recall than precision to be confident that most errors are highlighted for review, even at the cost of false positives. While it is always possible to increase recall by indiscriminately flagging additional text spans as errors, a naive strategy would lead to a large drop in precision. We propose a decoding algorithm for the fact-checking model which uses the output token probabilities to achieve a better precision-recall trade-off.

Our proposed approach (Algorithm 1) for increasing error detection recall relies on observing the probabilities of tokens generated as the revision by the fact-checker, and intervening at timesteps with low model confidence. Given a document D, claim C, and an initially generated revision $R = r_1 r_2 .. r_m$, we find the first position t where the probability assigned to token r_t falls below a threshold τ. At that timestep we then generate the token with the highest probability *excluding* r_t. We generate the remaining tokens (from $t+1$) as usual via greedy decoding to compute an alternate revision R'. Given R and R', assume the span $r_k r_{k+1} .. r_{k+w}$ was replaced by $x_1 .. x_q$. We make the replacement in R yielding $r_1 ... r_{k-1} x_1 .. x_q r_{k+w+1} ... r_m$. We then repeat the process of finding low probability tokens and making edits for the remaining tokens. After each iteration of the while loop, the value of $(|R| - t)$ decreases by at least 1, which guarantees termination of the program.

Increasing the value of τ in Algorithm 1 would lead to more edits being made to the claim, and vice-versa. We run the Flan-UL2 fact-checking model with different values of τ ranging from 0.0 to 0.99 and plot the resulting recall and precision at detecting errors annotated in the human evaluation experiment in Section E.4. We find that using Algorithm 1, we are able to increase the recall from about 40% to 60%, with a drop in precision from 95% to 40%. Although there is a drop in precision, the drop is much lower than what one would get by using a simple randomized baseline. The baseline strategy we compare against is to boost recall by randomly selecting additional words (beyond the ones already predicted as erroneous by the model) and mark them as erroneous too. We compute the number of words that need to be selected to boost expected recall to a certain level, and the resulting drop in expected precision that it entails (see Appendix for derivation). The thresholding approach maintains a much higher precision with increasing recall compared to the baseline strategy, where the precision already falls to around 28% when the recall increases to merely 43% (Figure 7.2).

Algorithm 1 Thresholded Edit

```
Input: document $D$, claim $C$, predicted evidence $E$, predicted revision $R$, model $\mathcal{M}$, threshold $\tau$
$Q = (D, C, E)$
$t = 1$
while $t \leq |R|$ do
    $p_1 p_2 ... p_{|V|}$ = NextTokProb$_{\mathcal{M}}(r_1..r_{t-1} \mid Q)$
    if $p_{r_t} \leq \tau$ then
        $r' = \arg\max_k (p_k \mid k \neq r_t)$
        prefix $= r_1 r_2 .. r_{t-1} r'$
        compl = Generate$_{\mathcal{M}}$(prefix $\mid Q$)
        $R'$ = prefix + compl
        $N_{del}, N_{add}$, repl = DiffAtPos$_t(R, R')$
        $R$ = prefix + repl + $r_{(t+N_{del})} ... r_{|R|}$
        $t = t + N_{add}$
    end if
    $t = t + 1$
end while
Output: updated revision $R$
```

We also compare using the custom decoding strategy in Algorithm 1 with simply flagging additional tokens as non-factual if their probability of generation (by the fact-checking model) falls below a variable threshold. We see that this strategy performs worse than using Algorithm 1 (Figure 7.2). This suggests that post-hoc usage of token probabilities from the fact-checking model does not isolate the non-factual spans as well as active intervention during the decoding process as done in Algorithm 1.

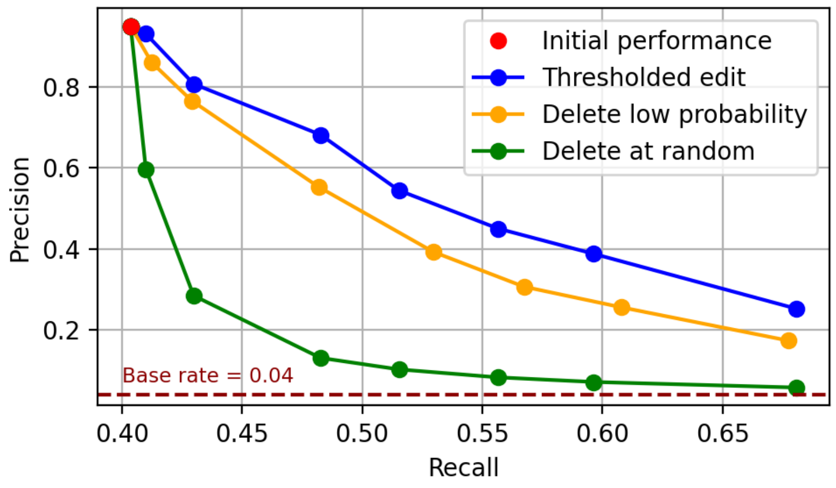


Figure 7.2: Variation in precision and recall of error identification by a fine-tuned Flan-UL2 model when using thresholded editing (Algorithm 1) versus editing out additional tokens either at random or by selecting the ones with low probability.

7.6 Binary Classification of Factuality

In the previous sections, we evaluated the performance of GENAUDIT at localizing factual errors within text and suggesting edits. However, it can also be repurposed as a binary classifier which simply predicts whether a long-form generated text is factually consistent or not with respect to given reference. To do that, we simply declare a given passage of text as factually inconsistent with respect to reference document if GENAUDIT suggests any edit to any sentence in it.

We evaluate the performance of GENAUDIT on the SummEdits benchmark [Laban et al., 2023] which consists of document-summary pairs where the summaries potentially contain factual errors. The source documents are taken from 10 different datasets representing a diverse group including legal documents, scientific papers, emails etc. We tokenize the source document into individual sentences before passing it through the fact-checking model. GENAUDIT achieves a balanced accuracy score of 74.7, outperforming many LLMs and traditional fact-checking methods, with the exception of Gemini-pro and GPT-4 (Table 7.5)[4].

[4]Values taken from the official Github repository https://github.com/salesforce/factualNLG

Model	**Balanced Accuracy**
Human Performance	90.92
GPT4	82.06
Gemini-pro	75.49
GENAUDIT	**74.75**
Claudev21	74.36
Claudev2	73.58
ChatGPT	71.18
PaLM-bison	69.04
QAFactEval [Fabbri et al., 2022]	65.46
Llama2-13b	58.35
Mistral-7b	57.78
SummaCConv [Laban et al., 2022]	57.14
DAE [Goyal and Durrett, 2021]	55.17
Llama2-7b	50.36

Table 7.5: Performance of models on the SummEdits benchmark for binary classification of factuality. Here GENAUDIT uses the fine-tuned Flan-UL2 backend, whereas other LLMs are zero-shot prompted.

7.7 Related Work

The tendency of language models to have factual errors in long-form generation was first noted in early research on machine translation [Arthur et al., 2016] and abstractive text summarization [See et al., 2017, Cao et al., 2018]. Subsequent works focused on the task of classifying whether a generated summary contains any factual error or not, using trained models [Kryściński et al., 2020, Goyal and Durrett, 2021], question-answering based approaches [Fabbri et al., 2022], or prompting LLMs [Laban et al., 2023]. While these works predicted factual correctness with respect to a given source document, recent works have implemented fact-checking against a large corpus (Wikipedia) by combining evidence retrieval and factuality prediction [Kamoi et al., 2023, Min et al., 2023]. Our effort goes beyond binary prediction of factual correctness, by also localizing the errors in the claims and fixing them via minimal editing. A concurrent work from Mishra et al. [2024] also attempts to fix factual errors via editing, and we compared the performance of their released model with ours in Section 7.4

Liu et al. [2023] and Krishna et al. [2023a] introduced the DeFacto and USB datasets respectively with human annotations to train and evaluate models for revising incorrect claims, and extracting evidence from a reference document. While both datasets can potentially be used to train GENAUDIT backend models, we used the USB dataset because of two reasons. First, USB contains comprehensive evidence labels for all facts in the claim, while DeFacto contains only evidence which justify the edits made to the claim. Second, DeFacto dataset contains mostly single-sentence summaries, while USB contains multi-sentence summaries, which are more common in practice. We extend these lines of work by contributing an interactive tool for fact-checking, and a comprehensive evaluation of the models trained on such data at fixing errors in modern LLM outputs with evidence.

Recent research has also proposed some approaches to prevent the generation of factually incorrect text from LLMs, instead of fixing the errors post-hoc. DoLa[Chuang et al., 2023] contrasts the output token logits computed at different layer outputs to promote appropriate tokens leading to more factual outputs. ITI [Li et al., 2023b] introduces biases in certain attention heads in the transformer model to promote factual outputs. While these two methods do not update model parameters, Tian et al. [2024] improve the factuality of LLM outputs by fine-tuning them using Direct Preference Optimization [Rafailov et al., 2023].

7.8 Conclusion and Future Work

We introduced GenAudit, a tool to assist users in fact-checking LLM generated outputs against inputs by presenting supporting evidence and highlighting (and fixing) errors. We trained models for fact-checking tasks which rival few-shot prompting of SOTA LLMs, and designed a web-interface for users to interact with. We evaluated GenAudit for fact-checking summaries generated by 8 LLMs for documents in 3 domains. Finally, we proposed a decoding algorithm for our fact-checking model to improve the recall of error identification while minimizing the cost in precision.

An important direction for future work is to enable the use of a large corpus of reference documents instead of single reference, which could extend GenAudit to work in open-ended generation scenarios too. Although GenAudit supports real-time prediction of evidence and factuality labels as the text is edited by user, the latency is high due to the large models used. Designing smaller models for fact-checking with comparable performance but lower latency would be useful here.

Acknowledgements

This work was supported in part by the National Science Foundation (NSF), grant RI: 2211954 and 2211955.

Limitations

Deciding whether a span of generated text is a hallucination involves subjective judgement. While some cases clearly fall in one category (e.g. if the model invents the name of a person), others can be quite debatable. For example, while summarizing a Reddit post about a person's bad experience, the summary mentioned them feeling *left out*, which was changed to *disappointed* by one annotator but was left as-is by the other annotator. Such decisions are motivated by the annotators' subjective judgement of whether the inferences made in the summary are consistent with the source, leading to difference in the edits made by them. This leads to the lack of a single gold ground-truth, which makes automatic evaluation challenging.

We used the edits made to LLM-generated summaries by human annotators as gold reference while evaluating different models' performance at error identification (Table 7.4). During the process, the annotators were shown suggested evidence and edits from the best-performing fine-tuned model (Flan-UL2) to evaluate its quality and to assist in the annotation process. Using the predictions from a model to assist collecting ground truth labels may lead to a benchmark that favors it and similar models. Ideally, we should conduct separate human evaluation for each model's predictions where annotators look at the suggestions from that model before making edits. However, that would be prohibitively expensive and time-consuming, and so we conduct automatic evaluation instead by re-using the collected annotations as ground truth.

Ethical Considerations

To evaluate GENAUDIT, we recruited proficient proofreaders who were selected after a qualifying round, focusing on their ability to identify inaccuracies in summaries. Annotators received compensation at an average rate of $25 USD per hour for their contributions. Annotators were provided the opportunity to discuss their concerns and questions with the authors throughout the annotation process.

The models presented in this work for fixing factual errors in LLM outputs are not perfect and some errors may not be detected by it. Hence, in critical application areas (such as clinical settings), it should be used in conjunction with a human verifier who uses GENAUDIT as a tool rather than as a perfect error detector. We have made this clear by transparently providing the performance of GENAUDIT via human evaluation in Section 7.4.

We verified that the different models and datasets in this work have licenses that permit research use.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In summary, this thesis presents methods, resources and insights to address challenges that are encountered by summarization models in real-world deployment.

In the first two chapters, we introduced methods to improve quality of model outputs when confronted with inputs having challenging characteristics. In Chapter 2, we presented a modular approach to summary generation to generate structured summaries of long inputs such as meeting transcripts. This leads to summaries with better overall quality and factuality as rated by automatic metrics as well as human evaluation. Additionally, since this approach uses shorter sequences in the summary generation step, it reduces the peak memory consumption and enables the use of larger and more potent models for it. In Chapter 3, we quantified the impact of different kinds of input noise on the quality of summaries generated by finetuned Pegasus models [Zhang et al., 2020]. We designed a lightweight method to detect and remove noisy tokens using encoder representations, and showed that it leads to recovery in output quality in experiments across different datasets and noise types.

In the next two chapters, we introduced pretraining approaches that do not use any surplus real-world corpus, yet are much better alternatives than finetuning from random initialization. In Chapter 4, we designed synthetic pretraining tasks using sequence transformations inspired by basic operations useful for summarization. We showed that even when using a gibberish vocabulary to create the pretraining examples, the models pretrained on them perform much better than randomly initialized models when finetuned on 4 different summarization datasets. In Chapter 5, we introduced *self-pretraining* — a technique for pretraining text encoder models such as RoBERTa [Liu et al., 2019b], which simply uses unlabeled text from the downstream fine-tuning dataset for pretraining. While prior works have looked at continual pretraining of models [Gururangan et al., 2020], we showed that even without the first phase of pretraining on a giant corpus such as BookWiki [Devlin et al., 2019], we are often able to perform nearly as well. Self-pretraining delivers strong performance compared to standard pretraining on a suite of 14 diverse NLP tasks, sometimes even outperforming standard pretraining. Together, these two chapters show that knowledge transfer from upstream pretraining corpora is not the only reason why pretraining works, and large gains actually come from some mechanism that is yet to be understood properly.

In the final two chapters, we introduced resources and tools to help tackle the problem of factual errors in model-generated summaries. In Chapter 6, we introduced the USB benchmark, a collection of labeled datasets for 8 different summarization-related tasks. The primary highlight of the benchmark is that it introduces high-quality human-labeled datasets for 4 fact-checking tasks: extracting evidence for summaries, classifying whether each summary sentence is factual or not, finding unsupported spans (hallucinations) in it, and editing summaries to fix any factual errors. We showed that for these tasks, training on human-labeled data greatly outperforms training using synthetically generated labels, which was the predominant practice in prior literature. In Chapter 7, we introduced GenAudit — a tool to assist humans in fact-checking LLM-generated summaries against the corresponding source documents. GenAudit highlights relevant evidence supporting each sentence in the generated summary, and suggests edits to the text to fix potential inconsistencies or hallucinations. We designed and released the interactive interface for the tool as well as open-sourced models to carry out the fact-checking tasks. Human evaluation demonstrates that GenAudit can fix factual errors in summaries generated from a variety of LLMs for documents taken from different domains.

8.2 Directions for Future Work

8.2.1 Processing long sequences

With the growing popularity of LLMs served via inference APIs, it is now easier to process long context sequences for tasks such as creating summaries. For example, OpenAI's GPT-4-Turbo [Achiam et al., 2023] can process upto 128k tokens, Anthropic's Claude 2.1 can process 200k tokens, and Gemini-1.5 [Reid et al., 2024] can process a whopping 10 million tokens at a time. However, the cost of using these LLM APIs grows with the number of tokens input. For example, feeding 100k tokens through GPT-4 would cost 1 USD. While it may seem like a small amount, it can quickly compound if multiple calls are made, for example to ask different questions about the same corpus. This simply reflects the high cost of passing tokens through a huge transformer model, which typically grows quadratically or at best linearly with simplifications in the attention mechanism [Zaheer et al., 2020, Xiao et al., 2023].

Hence, the idea of doing a multi-step inference by first extracting relevant information out of the long sequence using a computationally cheap model and then solving the main task using the expensive LLM (e.g. as outline in Chapter 2) still holds value. In the LLM era, this idea is epitomized in the form of Retrieval-Augmented Generation (RAG) systems [Gao et al., 2023]. In this setting, to answer queries, we first query a large corpus for relevant documents/snippets and then feed the retrieved information with the query to an LLM to generate the response. The primary research direction here is to design better retrieval modules by using vector search with text embedding models [Karpukhin et al., 2020, Ni et al., 2022, Khattab and Zaharia, 2020]. Another interesting direction is learning when to invoke the retrieval model to query for information while an LLM generates a long response [Asai et al., 2023].

Besides causing wastage of compute power and money, feeding irrelevant information into an LLM's prompt that's unrelated to the given task also increases the occurrence of incorrect responses [Weston and Sukhbaatar, 2023]. In such cases, filtering out of unnecessary information from the prompt and re-prompting the model with reduced information has been shown to be an effective strategy [Weston and Sukhbaatar, 2023].

8.2.2 Data-efficient pretraining

As generative AI models produce increasing amounts of revenue for the companies who own them, there are rising calls for compensation from the creators/platforms whose data was used to train them [Vincent, 2023, Grynbaum and Mac, 2023, Stempel, 2024]. Platforms such as Reddit and StackOverflow have changed their terms of usage to charge users who want to use their data for pretraining [Shakir, 2023, Dave, 2023]. Some companies are also reportedly paying creators directly to license the data used for pretraining [Lardinois, 2023]. In the wake of all this, it is clear that high-quality pretraining data would come with a price, making it is worthwhile to explore pretraining approaches which use lesser amount of real-world data.

One approach to data-efficient pretraining which broadly follows the spirit of Chapter 4 is to use synthetically generated data for pretraining. Li et al. [2023c] showed that creating synthetic data by prompting LLMs based on an initial smaller corpus of *seed* real-world data can deliver performance comparable to $10\times$ larger models pretrained on $10\times$ real-world data, on reasoning tasks. Maini et al. [2024] introduced a recipe for augmenting pretraining data with synthetically generated paraphrases, which can improve performance on a variety of NLP tasks. Another important direction of research is to decide the value of including certain documents in the pretraining corpora. There have been some recent works in this space which perform ablation studies on the pretraining corpus [Longpre et al., 2023] or propose ways to increase diversity of pretraining corpus [Tirumala et al., 2024].

8.2.3 Ensuring truthfulness of LLMs

In Chapter 7, we introduced models to post-edit LLM-generated summaries to fix hallucinations by checking against the given source document. However, often the LLM's response may contain information that is not present in the source but is factually correct [Cao et al., 2022]. Such hallucinations come from the internalized information in the model's parameters, and should not necessarily be penalized. Designing methods to differentiate between such harmless hallucinations and harmful ones which contain false/unverifiable claims is an interesting direction for future work.

Besides creating methods and models to fix factual errors in LLM-generated responses, it is also worthwhile to update the LLM itself so that it doesn't commit factual errors in its outputs. This is desirable because it can help avoid the hassle and computational cost of running a separate fact-checker model to fix errors. Tian et al. [2024] used Direct Preference Optimization [Rafailov et al., 2024] to tune Llama models to produce more factual responses. However, while the process increases the likelihood of generating factual responses, unfortunately it also promotes other output characteristics confounded with it, such as a simpler sentence structure. More research should look into improving factuality of model outputs without changing their other aspects. Another research direction to improving factuality of LLMs without running a separate fact-checking model is to use light-weight probes on its internal representations to detect when it's hallucinating [CH-Wang et al., 2023].

Chapter 9

Appendices

A Appendix for Chapter 2

A.1 Decoder Results with Oracle extracts

We present additional quantitative results (Table 3), including (i) The ROUGE scores on the test set when using oracle noteworthy utterances with both oracle and predicted clusters (for CLUSTER2SENT models). (ii) Two ablations on EXT2SEC: ALLEXT2SEC uses binary classification to extract all noteworthy utterances (not per-section), and an abstractive decoder that conditions on the section; while EXT2SECNOCOND uses a multilabel classification based extractor but *does not use section-conditioning* in the abstractive module. Both methods mostly perform worse than EXT2SEC demonstrating the benefit of using both section-specific extraction and section-conditioning in abstractive decoder.

Copy mechanism	**R-1**	**R-2**	**R-L**
Present	63.63	35.62	48.85
Absent	61.92	34.37	47.86

Table 1: Impact of copy mechanism in peformance of a pointer-generator model on medical dataset in CLUSTER2SENT using oracle noteworthy utterance clusters

Model	**R-1**	**R-2**	**R-L**
Pretrained T5	66.45	39.01	52.46
Randomly initialized T5	40.07	20.95	32.42

Table 2: Impact of pretraining on performance of T5-Base model on medical dataset with CLUSTER2SENT using oracle noteworthy utterance clusters

	Medical dataset			AMI corpus		
Method	**R-1**	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
EXT2NOTE (PG)	52.95	27.6	32.87	21.23	6.71	14.95
EXT2NOTE (T5-Small)	-	-	-	41.10	14.12	25.03
ALLEXT2SEC (PG)	50.74	24.33	32.18	40.20	13.71	22.52
ALLEXT2SEC (T5-Small)	-	-	-	41.68	15.43	24.72
EXT2SECNOCOND (PG)	56.10	32.05	43.23	42.51	15.71	23.79
EXT2SECNOCOND (T5-Small)	58.69	34.92	47.24	48.14	18.49	28.23
EXT2SEC (PG)	61.00	33.64	45.2	43.30	16.56	24.83
EXT2SEC (T5-Small)	62.37	36.39	49.11	46.85	18.19	28.74
CLUSTER2SENT (PG)	63.63	35.62	48.85	51.86	21.86	31.84
CLUSTER2SENT (T5-Small)	66.50	38.41	51.73	54.23	22.90	34.54
CLUSTER2SENT (T5-Base)	66.45	39.01	52.46	57.42	24.45	35.70
CLUSTER2SENT (PG+clustering heuristic)	63.12	35.08	47.96	47.17	18.99	27.31
CLUSTER2SENT (T5-Small+clustering heuristic)	66.08	37.73	50.66	47.53	19.70	28.95
CLUSTER2SENT (T5-Base+clustering heuristic)	65.94	38.26	51.31	51.24	21.47	29.81

Table 3: ROUGE scores achieved by different abstractive decoders using oracle noteworthy utterances

A.2 Impact of copy mechanism

When we do not use copy mechanism in the pointer-generator model, we observed a drop in its performance in the CLUSTER2SENT setting with oracle noteworthy noteworthy utterances and clusters(Table 1). Hence, we have used copy mechanism in all the pointer-generator models we train in this work.

A.3 Impact of pretraining

When training a randomly initialized T5-Base model on the medical dataset, even in CLUSTER2SENT setting with oracle clusters, it only got a ROUGE-1 around 40 (Table 2). This is over 16 points lower than what we get by starting with off-the-shelf pretrained T5 parameters, and is even worse than CONV2NOTE, highlighting the importance of pretraining.

A.4 Sample generated SOAP notes

Due to privacy concerns, we can not publish conversations from our dataset. Here, we present an obfuscated conversation from our test dataset, modified by changing sensitive content such as medicines, diseases, dosages (Figure 2). We also present the SOAP note generated by our best method, as well as the ground truth.

A.5 Model implementation details

For the hierarchical LSTM classifier, we have a word embedding size of 128 and both bidirectional LSTMs have a hidden size of 256. For BERT-LSTM, the BERT embeddings are initialized from bert-base-uncased (768 dimensions). LSTMs in either direction have a hidden-layer of size 512 and the entire model is optimized end-to-end with a learning-rate of 0.001. For BERT-LSTM, an input conversation is divided into chunks of 128 utterances. Due to GPU constraints, these chunks are processed one at a time. The pointer-generator models have a word embedding size of 128, and a hidden size of 256 for both the encoder and the decoder. The section embeddings used in section-conditioned pointer-generator network have 32 dimensions. During training of all pointer-generator models, the model is first trained without coverage loss [Tu et al., 2016] to convergence, and then trained further with coverage loss added. We tried coverage loss coefficients varying from 0.5 to 8.

The pointer-generator models were trained using Adam optimizer before coverage and using SGD after adding coverage. We tried learning rates between 10^{-4} and 10^{-3} with Adam. The next word prediction accuracy was used as the validation criterion for early stopping while training abstractive modules, with the exception of coverage-augmented models that used a combination of crossentropy and coverage loss. Micro-averaged AUC was used as the validation criterion for training of extractive modules.

We employ beam search with beam size 4 to decode outputs from both models. For the vanilla pointer-generator model used in CONV2NOTE and EXT2NOTE, we modified the beam search procedure to make sure that all the SOAP note sections are generated in proper order. We start the beam search procedure by feeding the header of the first section (chief complaint). Whenever the model predicts a section header as the next word and it shows up in a beam, we check if it is the next section to be generated. If not, we replace it with the correct next section's header. Any `end-of-summary` tokens generated before all the sections have been produced are also replaced similarly. Note that producing all sections simply means that the headers for each section have to be generated, and a section can be left empty by starting the next section immediately after generating the previous header. The decoding length for beam search is constrained to be between 5^{th} and 95^{th} percentile of the target sequence length distribution,

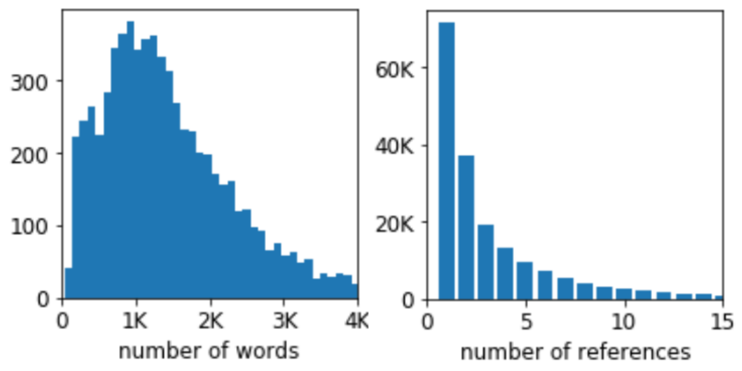


Figure 1: Histogram of number of words in a conversation and the number of evidence utterances per summary sentence for the medical dataset

	Medical conversations			**AMI corpus**		
Count	C2N	C2S-P	C2S-T	C2N	C2S-P	C2S-T
Total sentences	956	1268	1277	414	358	381
Repetitive	96	127	147	213	14	14
Incoherent	162	158	58	9	134	27
True statements	587	848	931	89	103	227
False statements	100	116	125	71	75	68
Truthfulness undecided	11	19	16	32	32	45
Irrelevant	25	34	24	14	15	2
Under incorrect section	56	42	39	4	2	18

Table 4: Number of sentences produced by different methods that were judged to have different listed characteristics by human raters. C2N:CONV2NOTE, C2S-P:CLUSTER2SENT with pointer-generator, C2S-T:CLUSTER2SENT with T5-base. BERT-LSTM used for medical dataset, hierarchical-LSTM used for AMI corpus.

calculated on the training set.

A.6 Simulating ASR Errors

We simulate ASR errors at any given percentage rate by randomly selecting the percentage of the words in the conversation and replacing them with phonetically similar words. To reduce the search space of possible candidates for each word, we use the `suggest()` function taken from the Pyenchant[1] library that provides auto-correct suggestions for the input word. Each suggestion is then passed through the Refined SoundEx algorithm to find the phonetic distance between the original and the suggested word. We use the pyphonetics[2] package for a python implementation of this algorithm. For our final candidate list, we choose words that are at phonetic distance of 1 from the original word. Finally, a candidate is chosen at random from this list to replace the original.

A.7 More Experimental Details

We trained models on multiple Nvidia `Quadro RTX 8000`, `RTX 2080Ti` and `V100` GPUs. The extractive modules were evaluated using standard classification metrics from `scikit-learn` [3] and quality of summaries were evaluated using ROUGE scores calculated with the `pyrouge` Python package [4] which is a wrapper around the `ROUGE-1.5.5` Perl script.

[1]https://pypi.org/project/pyenchant/
[2]https://pypi.org/project/pyphonetics/
[3]`https://scikit-learn.org`
[4]`https://pypi.org/project/pyrouge`

Subsection	ROUGE-1	ROUGE-2	ROUGE-L	N	L
chief complaint	44.34	28.12	43.59	592	11.46
review of systems	46.88	28.35	43.28	514	29.24
past medical history	53.48	37.70	51.80	547	17.81
past surgical history	58.44	43.08	57.04	230	10.36
family medical history	51.94	36.49	50.13	72	16.14
social history	57.72	37.82	56.30	97	10.33
medications	49.56	23.53	47.64	549	15.28
allergies	39.29	6.63	38.32	21	8.57
miscellaneous	28.87	11.61	24.90	415	34.44
immunizations	55.95	27.49	54.81	25	7.32
laboratory and imaging results	58.36	41.18	55.11	448	19.37
assessment	39.01	15.31	25.35	570	132.41
diagnostics and appointments	52.85	35.70	50.43	488	17.67
prescriptions and therapeutics	50.53	33.51	48.10	446	18.73
healthcare complaints	30.11	15.79	29.57	43	16.74

Table 5: Average ROUGE scores (from CLUSTER2SENT T5Base+BLSTM) for each section of SOAP note (N-number of test datapoints with the section populated, L-average number of words in ground truth)

Section	Base rate(%)	Precision	Recall	F1	Accuracy	AUC
chief complaint	3.12	34.71	33.93	34.31	95.95	86.81
review of systems	5.10	51.35	51.82	51.58	95.04	93.12
past medical history	3.41	36.00	36.52	36.26	95.63	88.00
past surgical history	0.99	33.80	34.50	34.14	98.68	93.74
family medical history	0.31	52.31	45.25	48.53	99.70	99.23
social history	0.53	59.81	54.87	57.23	99.56	95.41
medications	4.45	51.66	49.13	50.36	95.69	92.02
allergies	0.16	30.86	12.44	17.73	99.82	89.46
miscellaneous	3.71	24.06	16.17	19.34	95.00	80.05
immunizations	0.05	63.64	64.62	64.12	99.96	97.63
laboratory and imaging results	2.46	50.00	55.15	52.45	97.54	93.84
assessment	14.19	38.09	42.01	39.96	82.08	76.89
diagnostics and appointments	2.10	55.60	40.16	46.63	98.07	94.22
prescriptions and therapeutics	3.11	41.28	38.43	39.81	96.39	92.40
healthcare complaints	0.25	20.47	21.90	21.17	99.60	85.93

Table 6: Performance of BERT-LSTM on extracting noteworthy utterances for various SOAP sections

Predicted relevant subsections	Conversation utterances
(PT) (A)	**DR** Okay, so, um, we are going to talk a little bit about being a Metformin candidate .
(CC) (PMH) (A)	**DR** Um , we have talked about your hemoglobin and the things , what are , so what are the things that , that keep you from , um , from managing your anemia well ?
	DR, I know there's a lot of stuff that troubles you.
(M)	**PT** Snacking and stress eating.
	PT Eating late in the evenings instead of, um, at a reasonable time -
	DR Right.
	PT At night, late.
(M) (A)	**PT** Poor meal planning.
(PMH) (LIR) (A)	**DR** Right, and I think that's in the, we can all take a little note for but one of things that really got me worried because your last Hemoglobin was really low -
	PT Uh-huh.
(LIR) (A)	**DR** It was below , it was below 10 , and we 've had this consistent pattern and you 've really , I mean , you really have given it an effort and I have to give it up to you that you 've been trying and , um , so we 're down to like just a couple of options and so I want to just kind of put them before you .
(A) (PT) (Med)	**DR** I 've got, I 'm, I 'm considering once a day Metformin with you at some point .
(A)	**DR** Um, I do n't want to use that as a threat.
(A)	**DR** I do n't want to use it as like a, oh , you 've been a bad patient you deserve to be on Metformin .
(A) (PT)	**DR** Um , I do have one other option , um , but I want to counsel you that , that Metformin , even if , if we did , we do go to it , it is not a punishment .
(A)	**DR** It is something to kind of get your baseline down to a regular, regular situation and you only have to do it once a day.
(A)	**DR** Um, and I know that one of the things that we have for anemics is their eating habits .
(A) (PT)	**DR** And, so , I am proposing as instead of using Metformin this time , um , that we use something called Lipitor for the , for the eating at nighttime .
(A)	**DR** Um, it's supposed to reduce the incidence of having those nighttime cravings so that you can work , you can do your things , you can plan a little bit better .
(A)	**DR** It 's , it's originally for ADHD so some people actually feel a little bit more focused , um , and controlled but it also affects appetite centers and so it's supposed to do it for the longer term as opposed to using like a fen phen , um , so , which is short term .
	DR So, um , I 'm really hoping with your interest in it and with the coverage hopefully , I know , with your particular plan it should be covered and we can get a discount .
(PT)	**DR** Um, we do it once a day with your other medications , which are actually pretty minor .
(DA)	**DR** Um, and then we check you again in eight weeks .
(DA)	**PT** Okay.
	DR All right?
(A) (DA)	**DR** And, so what we do is we say , you know , it should be , we usually will do three months but then eight weeks we should see some difference from today .
	DR We should see some kind of improvement and then we can sort of celebrate that in and of itself, if that's okay with you.
	PT That sounds great.
(DA)	**DR** Cool, all right well we will plan to meet again in eight weeks .
	PT Okay.
	DR And, uh , and we 'll go from there .
	PT Okay.
	DR Cool, all right , cool .

Cluster2Sent+T5Base

Chief Complaint: anemia .
Past Medical History: anemia .
Medications: metformin .
Miscellaneous: patient has snacking and stress eating . poor meal planning .
Laboratory and Imaging Results: last hemoglobin was low at 10 .
Assessment: discussed about being a metformin candidate . discussed about hemoglobin and the things that keep patient from managing anemia well. discussed that patient 's last hemoglobin was really low , it was really low , it was really low , it was really low , it was really low , and we have had this consistent pattern and you really have given it effort and we have had this. followup in 8 weeks .
Diagnostics and Appointments: followup in 8 weeks .
Prescriptions and Therapeutics: the patient will be a metformin candidate. metformin once a day. coumadin twice a day with other medications, which are actually pretty minor .

Ground truth

Chief Complaint: follow-up. anemia.
Past Medical History: anemia.
Medications: metformin
Miscellaneous: patient is not following a correct diet plan (snacking and stress eating).
Laboratory and Imaging Results: hemoglobin was really low below 10.
Assessment: anemia. night time eating. discussed with the patient the importance of bringing up the hemoglobin to a considerable level and also discussed couple of other options. discussed the new medication called lipitor which will help the patient bringing up the hemoglobin and can take it once a day with other medications. discussed that the lipitor will reduce the nighttime cravings so that the patient can plan better (originally for ADHD to better focus). discussed with the patient that with the current insurance coverage , the patient may get a discount with lipitor.
Diagnostics and Appointments: advised to follow up in 8 weeks.
Prescriptions and Therapeutics: metformin. lipitor.

Figure 2: Sample conversation (obfuscated) with SOAP note generated by the best method and the ground truth

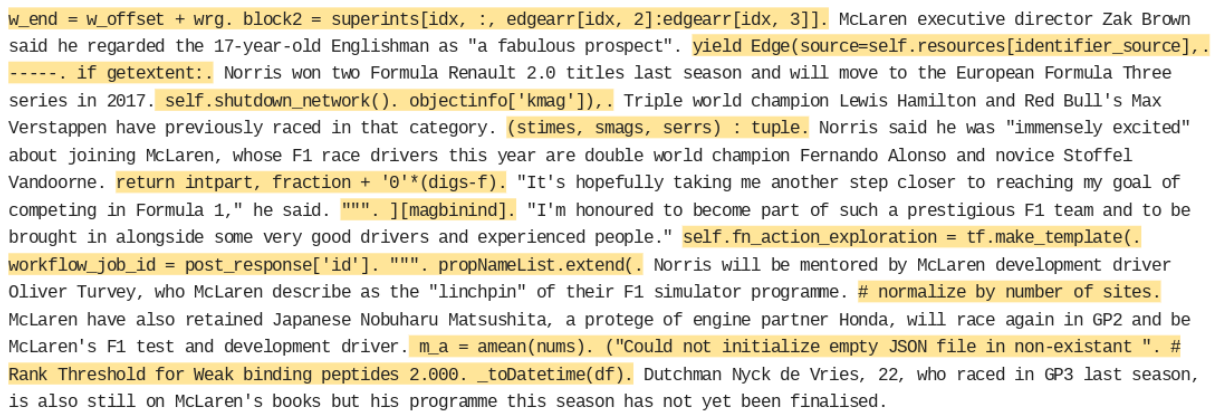
B Appendix for Chapter 3

B.1 Selection of shorter inputs to avoid truncation

In our experiments, we exclude those datapoints from the datasets which are longer than a certain threshold. This is done to avoid any truncation of the input (including inputs with added noise) when feeding them into the model. Since adding noise to the input increases its length, it may happen that some clean tokens might be pushed beyond the maximum allowed input length and hence removed when the input is truncated. In such a scenario, removing noisy tokens before feeding the sequence into the model would also cause such clean tokens to be fed into the model again because they can now be accommodated within the input length limit. When measuring the benefit of noise filtering, the benefit from removal of noisy tokens would then be confounded with the benefit from such "resurrection" of clean tokens. To avoid this we only retain those inputs in our datasets where the input length would be within limit even after addition of noise. Since the maximum noise amount we use in our experiments is 0.5, we only retain datapoints which have no more than half of the maximum allowed tokens to input into the model. (Table 7).

Table 7: Number of datapoints retained in the test set of datasets after removing inputs longer than maximum length (Maxlen)

Dataset	Count	Maxlen	Retention
XSUM	7516	512	66.5%
CNN/DailyMail	2948	512	25.6%
RedditTIFU-long	2790	512	66.2%
SAMSum	686	256	83.8%



```
w_end = w_offset + wrg. block2 = superints[idx, :, edgearr[idx, 2]:edgearr[idx, 3]]. McLaren executive director Zak Brown
said he regarded the 17-year-old Englishman as "a fabulous prospect". yield Edge(source=self.resources[identifier_source],.
-----. if getextent:. Norris won two Formula Renault 2.0 titles last season and will move to the European Formula Three
series in 2017. self.shutdown_network(). objectinfo['kmag']),. Triple world champion Lewis Hamilton and Red Bull's Max
Verstappen have previously raced in that category. (stimes, smags, serrs) : tuple. Norris said he was "immensely excited"
about joining McLaren, whose F1 race drivers this year are double world champion Fernando Alonso and novice Stoffel
Vandoorne. return intpart, fraction + '0'*(digs-f). "It's hopefully taking me another step closer to reaching my goal of
competing in Formula 1," he said. """. ][magbinind]. "I'm honoured to become part of such a prestigious F1 team and to be
brought in alongside some very good drivers and experienced people." self.fn_action_exploration = tf.make_template(.
workflow_job_id = post_response['id']. """. propNameList.extend(. Norris will be mentored by McLaren development driver
Oliver Turvey, who McLaren describe as the "linchpin" of their F1 simulator programme. # normalize by number of sites.
McLaren have also retained Japanese Nobuharu Matsushita, a protege of engine partner Honda, will race again in GP2 and be
McLaren's F1 test and development driver. m_a = amean(nums). ("Could not initialize empty JSON file in non-existant ". #
Rank Threshold for Weak binding peptides 2.000. _toDatetime(df). Dutchman Nyck de Vries, 22, who raced in GP3 last season,
is also still on McLaren's books but his programme this season has not yet been finalised.
```

Figure 3: Sample excerpt from an article from XSUM dataset corrupted with code noise.

Table 8: ROUGE scores on clean input and changes when adding different kinds of noise, and after the noise is filtered out using the Sent method based OOD scores (Noise amount: 0.5)

Variant	**Noise type**	**ROUGE-1 / 2 / L**		
		XSum		
		Small	Base	Large
Clean	-	43.35 / 20.49 / 35.73	47.03 / 23.72 / 39.05	48.92 / 25.65 / 40.95
Noisy	Code	31.54 / 12.44 / 25.07	38.74 / 17.34 / 31.43	47.53 / 24.48 / 39.63
	Emoji	31.79 / 15.10 / 26.29	40.08 / 20.32 / 33.24	47.86 / 25.02 / 40.16
	Randomsent	32.38 / 13.10 / 26.09	36.54 / 16.63 / 29.87	42.67 / 21.06 / 35.38
	URL	36.47 / 15.45 / 29.42	37.56 / 16.91 / 30.55	47.37 / 24.45 / 39.64
Filtered	Code	38.72 / 17.00 / 31.61	43.50 / 20.98 / 35.94	48.55 / 25.45 / 40.64
	Emoji	42.94 / 20.24 / 35.38	46.04 / 23.29 / 38.27	48.41 / 25.41 / 40.61
	Randomsent	39.84 / 17.81 / 32.42	43.88 / 21.30 / 36.11	46.65 / 23.84 / 38.85
	URL	41.86 / 19.30 / 34.43	45.69 / 22.69 / 37.80	48.41 / 25.34 / 40.54
		CNN-Dailymail		
		Small	Base	Large
Clean	-	44.50 / 22.74 / 32.27	45.70 / 23.72 / 33.43	46.20 / 24.08 / 33.61
Noisy	Code	36.74 / 16.58 / 26.50	38.54 / 17.22 / 27.32	42.23 / 20.32 / 30.23
	Emoji	43.97 / 22.11 / 31.35	45.25 / 23.21 / 32.77	45.95 / 23.74 / 33.27
	Randomsent	42.63 / 21.02 / 30.16	44.09 / 22.17 / 31.40	44.81 / 22.75 / 32.07
	URL	42.19 / 20.57 / 30.17	43.60 / 21.45 / 31.05	44.89 / 22.69 / 32.27
Filtered	Code	33.64 / 15.60 / 24.33	36.66 / 16.97 / 26.31	43.45 / 21.78 / 31.46
	Emoji	44.14 / 22.27 / 31.68	45.30 / 23.40 / 33.00	45.84 / 23.68 / 33.17
	Randomsent	42.99 / 21.34 / 30.58	44.26 / 22.35 / 31.70	45.16 / 23.08 / 32.51
	URL	42.77 / 21.27 / 30.87	43.89 / 21.97 / 31.65	45.34 / 23.43 / 32.83
		Samsum		
		Small	Base	Large
Clean	-	50.56 / 25.66 / 42.16	51.73 / 27.80 / 43.64	53.50 / 29.53 / 45.68
Noisy	Code	44.81 / 21.32 / 37.62	48.32 / 25.29 / 41.30	50.24 / 26.85 / 43.29
	Emoji	49.27 / 24.41 / 41.54	50.75 / 27.37 / 43.30	53.31 / 29.25 / 45.70
	Randomsent	39.81 / 17.27 / 32.31	42.79 / 21.30 / 35.83	42.22 / 21.35 / 35.85
	URL	46.46 / 22.25 / 38.60	48.31 / 25.22 / 41.21	50.51 / 27.57 / 43.24
Filtered	Code	49.22 / 24.56 / 41.24	50.70 / 26.94 / 43.07	52.43 / 28.87 / 45.23
	Emoji	49.00 / 24.41 / 41.42	50.49 / 27.25 / 43.03	53.32 / 29.21 / 45.64
	Randomsent	47.36 / 23.31 / 39.43	49.47 / 25.64 / 41.60	50.40 / 26.56 / 42.89
	URL	49.65 / 25.16 / 41.58	51.29 / 27.63 / 43.39	52.56 / 28.70 / 45.07
		Reddit-TIFU		
		Small	Base	Large
Clean	-	24.06 / 7.81 / 19.86	26.74 / 9.20 / 21.95	27.45 / 9.65 / 22.56
Noisy	Code	17.95 / 5.74 / 14.87	18.72 / 6.21 / 15.46	20.35 / 6.91 / 16.85
	Emoji	20.25 / 6.47 / 16.65	20.09 / 7.14 / 16.51	22.51 / 7.90 / 18.55
	Randomsent	21.15 / 6.62 / 17.18	22.09 / 7.14 / 18.08	21.47 / 7.09 / 17.73
	URL	21.02 / 6.66 / 17.23	24.25 / 8.09 / 19.76	24.55 / 8.17 / 20.26
Filtered	Code	20.98 / 6.83 / 17.37	22.24 / 7.58 / 18.27	24.31 / 8.46 / 20.15
	Emoji	23.49 / 7.71 / 19.42	21.95 / 7.59 / 17.99	23.79 / 8.40 / 19.59
	Randomsent	23.05 / 7.32 / 18.81	25.57 / 8.64 / 20.78	26.37 / 9.12 / 21.59
	URL	21.96 / 7.10 / 17.94	24.88 / 8.44 / 20.37	25.74 / 8.84 / 21.14

C Appendix for Chapter 4

C.1 Human evaluation of summary quality

We conducted a manual evaluation by human raters to compare the quality of summaries generated by a T5 model trained from random initialization, versus a model fine-tuned after pretraining with our designed tasks with a nonsense corpus. We call these two models T5-RI and T5-OurTasks following previous convention (Table 4.2). We generate summaries for 25 randomly selected articles from the test set of CNN-Dailymail dataset and 25 articles from the XSum dataset. Annotators are asked to choose the better summary out of the ones generated from finetuned T5-RI and T5-OurTasks models. Each pair of summaries is annotated by two annotators. The annotation interface is shown in Figure 4. The annotators were recruited from Amazon Mechanical Turk, and were required to satisfy the given requirements: they should have completed at least 1000 HITs with an approval rate greater than or equal to 95%, they should be located in the US, and should have the Masters status provided by AWS.

We found that summaries generated from T5-OurTasks had a win-rate of 96% for CNN-Dailymail articles and 70% for XSum articles, showing that it produces better summaries on average than T5-RI. The p-values for a one-tailed Student's t-test are $p < 0.0001$ on CNN-Dailymail dataset and $p < 0.003$ on XSum dataset.

C.2 Variation in performance with multiple seeds

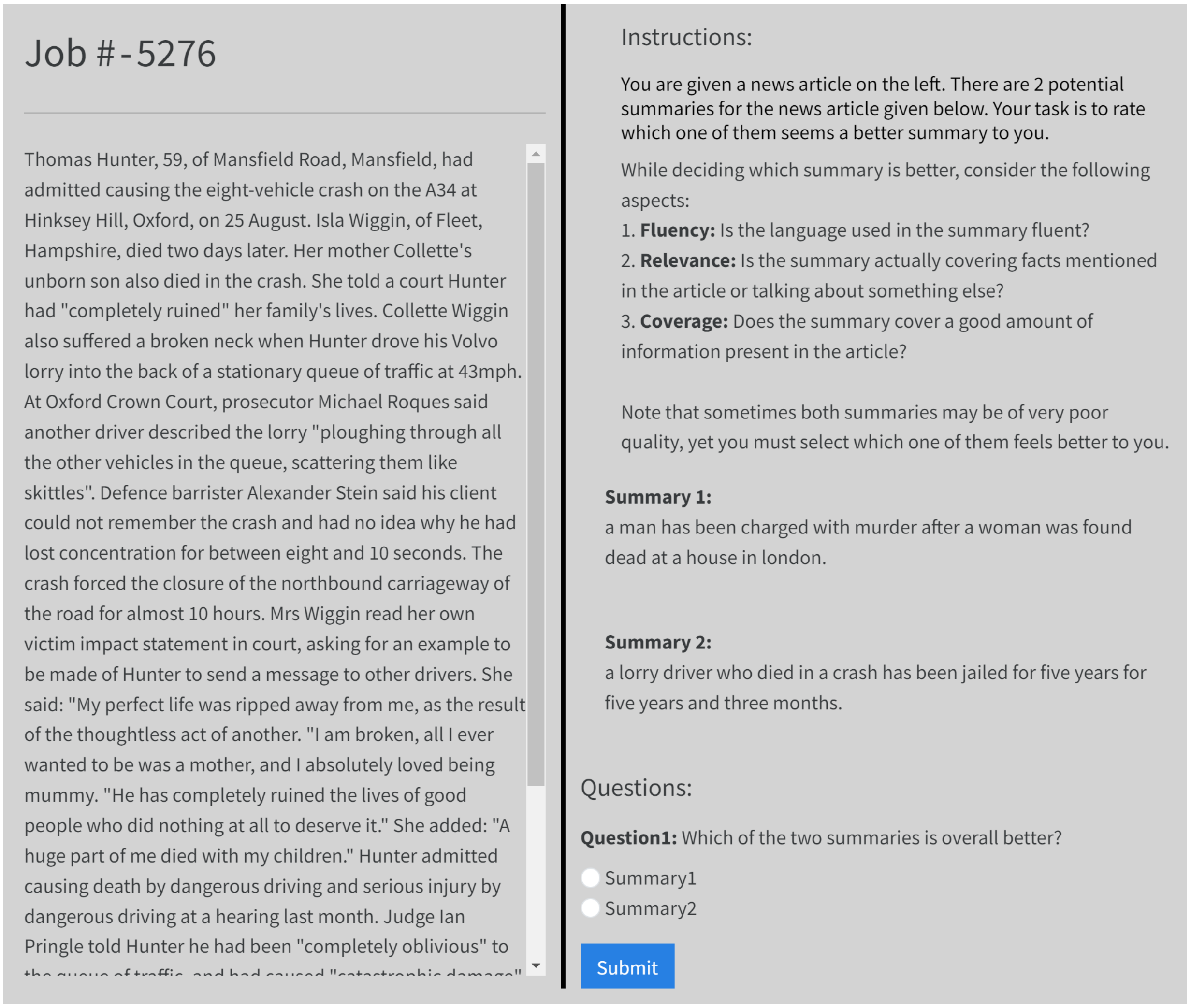
We fine-tuned each pretrained model on each summarization dataset 3 times with different seeds and plotted the mean and standard deviation of the resulting performance in Table 9. The variation in performance is mostly small, suggesting the robustness of the comparative trend between the 3 initialization schemes.

C.3 Exclusions from ensemble of our tasks

When creating artificial summaries requires using multiple of our proposed elementary tasks, the different keywords added to the input signal to the model which tasks are required for it. Three of our proposed tasks do not always involve keyword addition— *CopyFirstSentence, CopyLastSentence, CheckKeyword*. Hence we exclude them when creating the pretraining corpus with our ensemble of tasks. We also exclude the *SumOfNumbers* and *CompareNumbers* tasks because they could not be learnt even in isolation by a randomly initialized T5 model training on 100k datapoints.

C.4 Evaluation metrics

We measure the quality of generated summaries using ROUGE scores Lin and Hovy [2002] which measure n-gram overlap between a generated and reference summary to assess its quality. We use the ROUGE-1,2 and L variants of this metric which measure overlap in unigrams, bigrams and longest common subsequence respectively. We also present the average performance


Job # - 5276

Thomas Hunter, 59, of Mansfield Road, Mansfield, had admitted causing the eight-vehicle crash on the A34 at Hinksey Hill, Oxford, on 25 August. Isla Wiggin, of Fleet, Hampshire, died two days later. Her mother Collette's unborn son also died in the crash. She told a court Hunter had "completely ruined" her family's lives. Collette Wiggin also suffered a broken neck when Hunter drove his Volvo lorry into the back of a stationary queue of traffic at 43mph. At Oxford Crown Court, prosecutor Michael Roques said another driver described the lorry "ploughing through all the other vehicles in the queue, scattering them like skittles". Defence barrister Alexander Stein said his client could not remember the crash and had no idea why he had lost concentration for between eight and 10 seconds. The crash forced the closure of the northbound carriageway of the road for almost 10 hours. Mrs Wiggin read her own victim impact statement in court, asking for an example to be made of Hunter to send a message to other drivers. She said: "My perfect life was ripped away from me, as the result of the thoughtless act of another. "I am broken, all I ever wanted to be was a mother, and I absolutely loved being mummy. "He has completely ruined the lives of good people who did nothing at all to deserve it." She added: "A huge part of me died with my children." Hunter admitted causing death by dangerous driving and serious injury by dangerous driving at a hearing last month. Judge Ian Pringle told Hunter he had been "completely oblivious" to

Instructions:

You are given a news article on the left. There are 2 potential summaries for the news article given below. Your task is to rate which one of them seems a better summary to you.

While deciding which summary is better, consider the following aspects:
1. **Fluency:** Is the language used in the summary fluent?
2. **Relevance:** Is the summary actually covering facts mentioned in the article or talking about something else?
3. **Coverage:** Does the summary cover a good amount of information present in the article?

Note that sometimes both summaries may be of very poor quality, yet you must select which one of them feels better to you.

Summary 1:
a man has been charged with murder after a woman was found dead at a house in london.

Summary 2:
a lorry driver who died in a crash has been jailed for five years for five years and three months.

Questions:

Question1: Which of the two summaries is overall better?

Summary1
Summary2

Figure 4: Annotation interface used for collecting human preferences for summaries generated by finetuned T5-RI and T5-OurTasks models on the XSum dataset

Model	Metric	T5-RI	T5-OurTasks	T5-Offshelf
CNN-DM-10K	Rouge-1	9.56±0.27	35.4±0.15	39.6±0.15
	Rouge-2	1.02±0.06	14.83±0.12	18.27±0.14
	Rouge-L	7.31±0.15	24.05±0.16	27.86±0.11
XSum-10K	Rouge-1	15.67±0.14	20.42±0.13	28.82±0.27
	Rouge-2	2.44±0.03	4.11±0.08	8.48±0.15
	Rouge-L	12.87±0.1	16.27±0.1	22.36±0.2
RottenTomatoes	Rouge-1	12.21±1.68	15.12±0.55	24.65±0.68
	Rouge-2	0.32±0.06	2.2±0.1	9.1±0.48
	Rouge-L	10.31±1.55	12.15±0.3	19.66±0.61
Rotowire	Rouge-1	3.05±1.61	38.41±0.49	38.67±1.0
	Rouge-2	0.41±0.3	11.36±0.32	13.27±0.45
	Rouge-L	2.71±1.37	19.55±0.28	20.32±0.41

Table 9: Mean and standard deviation of performance of models finetuned from different initialization schemes on 4 datasets, across 3 random seeds. T5-RI: random initialization, T5-OurTasks: weights pretrained with a nonsense corpus using tasks proposed in Section 4.5, T5-Offshelf: standard pretrained checkpoint

of models at predicting the next token of a summary given all the ground truth past tokens (Table 12). To measure this, we use the accuracy and the negative-log-likelihood metrics which are standard for multi-class classification. We average these metrics across different decoding timesteps of summary generation, and then average it again across all the summaries in the test set.

C.5 Experimental details

Hyperparameters We use the T5-Small architecure with 60.5 million parameters as our transformer-based model. The models are all trained using the BertAdam optimizer with a learning rate of 10^{-4}. For the pointer-generator model, the token embedding size is 128, its encoder is a bidirectional LSTM with hidden size 256 the decoder is a unidirectional LSTM of the same size. The entire model had 4.4 million parameters. For a fair comparision, we use wordpiece tokenization with all models with the same tokenizer and vocabulary as used by the standard T5 model. The validation metric used in all experiments was accuracy on the next-token prediction on the summaries. A patience value of 5 epochs was used for early stopping.

For CNN-Daiymail dataset, we truncated the input and output lengths according to Zou et al. [2020] (Table 11). We use the same lengths for the XSum dataset as well . For the Rotowire and Rottentomatoes dataset, the input and output lengths were much longer and even with a batch size of 1, we had to truncate them to values that allowed us to accommodate training with the available GPU memory (32GB). While decoding, we used beam search with beam size 4, and set the minimum and maximum decoding lengths to the 5 and 95 percentile of their observed distribution.

Computing infrastructure Most experiments were carried out on 8 Nvidia V100 GPUs with 32 GB of memory. Some experiments with CNN-Dailymail and XSum datasets were carried out on 4 Nvidia RTX2080Ti GPUs with 11GB of memory.

Details of dataset splits For the Rotowire and RottenTomatoes datasets, we use the standard training, validation and test splits with sizes shown in Table 10. For the CNN-Dailymail and XSum datasets, we use the standard test splits, but reduce the training and validation set sizes to 10k and 1k respectively by uniformly subsampling from the standard full dataset splits.

	CNN-DM-10K	**XSum-10K**	**RottenTomatoes**	**Rotowire**
Train	10000	10000	2458	3398
Validation	1000	1000	536	727
Test	11490	11333	737	728

Table 10: Sizes for Train, validation and test splits for all datasets

	CNN-DM-10K	**XSum-10K**	**RottenTomatoes**	**Rotowire**
max source length	512	512	6000	5160
max target length	256	256	∞	815
batch size	16	16	1	1
max decode length	148	42	52	815
min decode length	44	18	16	223

Table 11: Hyperparameters used for fine-tuning models on the 4 datasets

Experiment	**CNN-DM-10K**		**XSum-10K**		**Rottentomatoes**		**Rotowire**	
	Acc	NLL	Acc	NLL	Acc	NLL	Acc	NLL
T5-OffShelf	65.15	1.71	53.68	2.34	51.78	2.77	68.04	1.50
T5-RandomInit	29.78	4.92	32.60	4.75	24.75	5.36	48.30	2.61
Nonsense Upstream Corpus								
T5-OurTasks	54.74	3.18	38.98	4.27	33.42	5.08	63.59	1.78
T5-STEPTasks	54.71	3.18	39.47	4.21	28.65	5.13	58.89	1.99
Real Upstream Corpus								
T5-OurTasks	54.87	2.93	41.21	3.76	39.64	4.12	64.02	1.78
T5-STEPTasks	57.91	2.46	46.83	3.08	45.34	3.43	64.08	1.63
PG Models Randomly Initialized vs Pretrained (Nonsense Upstream Corpus)								
PG-RandomInit	51.14	2.91	33.05	4.14	33.35	4.37	59.12	1.92
PG-OurTasks	51.70	2.89	33.80	4.14	34.40	4.29	59.30	1.92
PG-STEPTasks	51.79	2.88	34.13	4.14	35.06	4.21	59.00	1.94

Table 12: Accuracy (Acc) and negative log likelihood (NLL) for next token prediction on summaries

Pretraining task	R1	R2	RL	Pr%
CopyKwdMultipleSent-Shuffled	**23.34**	5.46	15.41	99.66
TopicSegregation	23.04	**7.79**	**16.52**	99.88
TruncateSentence	17.07	2.50	11.81	100.00
CopyQuoted	11.03	1.32	8.32	99.82
BreakClauses	10.46	1.18	7.95	99.80
CopyKwdMultipleSent-InOrder	10.14	1.14	7.70	99.84
ReplaceClassKeyword	9.70	0.95	7.36	99.98
ParaphraseWords	9.70	0.99	7.42	99.98
CopyKwdOneSentence	9.45	1.06	7.23	99.90
CopyFirstSentence	9.28	1.08	7.22	99.88
CopyBulleted	9.01	1.00	6.88	99.58
CopyKwdMultipleSent-Sorted	8.48	0.83	6.59	99.68
MajorityKeyword	8.45	0.85	6.49	100.00
ThresholdNumber	7.83	0.77	6.05	100.00
CheckKeyword	7.79	0.77	5.94	100.00
CopyLastSentence	7.78	0.72	6.12	98.40
JoinClauses	7.72	0.81	6.09	98.82
ClassifyKeyword	6.80	0.62	5.34	100.00
LargestNumber	6.52	0.58	5.14	99.88
SumOfNumbers	5.03	0.40	4.14	25.06
CompareNumbers	1.89	0.04	1.75	48.88

Table 13: For different models pretrained on one individual task each, their performance on CNN-Dailymail-10K in terms of ROUGE (R1,R2,RL), and their accuracy in percentage on the pretraining task (Pr%)

Elementary subtask	**Description**
CheckKeyword	Check if the input has a special keyword or not.
ClassifyKeyword	Input contains 1 of 10 special keywords - 5 or them are positive and 5 of them are negative adjectives. Task is to tell whether mentioned adjective was positive or negative
MajorityKeyword	Out of two given keywords, find which one occurs more number of times
CopyFirstSentence	Copy first sentence
CopyBulleted	Exactly one sentence is a bullet point and starts with the bullet marker. You have to copy over that sentence without copying the marker.
CopyQuoted	Copy text within quotes
CopyLastSentence	Copy last sentence
CopyKwdOneSent	Copy single sentence containing one of many special defined keywords
CopyKwdMultipleSentInOrder	Copy all sentences containing any special keyword in the same order as they appear in text.
CopyKwdMultipleSentSorted	Copy all sentences containing keywords but sort them according to the canonical ordering of keywords
CopyKwdMultipleSentShuffled	Copy all sentences containing keywords in any order. The sentences in ground truth may be any possible order.
ReplaceClassKeyword	There exist many keywords, each belonging to one of 3 classes. You have to mention the class of the mentioned keyword
CompareNumbers	Given two numbers in the text, say which one is bigger
SumOfNumbers	Sum numbers
ThresholdNumber	The input contains a number between 0 and 100. You have to say if the number was above or equal to the threshold of 50 of lower than it
LargestNumber	Find out largest of one or more numbers in the input.
TruncateSentence	Copy a sentence but only till the cutoff keyword is encountered
BreakClauses	Break a single sentence into multiple ones containing one clause each
JoinClauses	Join clauses from multiple sentences to make one longer sentence
ParaphraseWords	Copy the sentence containing one of pre-specified special keywords. But replace the keyword with any of its multiple synonyms. The j^{th} synonym of i^{th} keyword src_i is given by $target_{ij}$
TopicSegregation	Copy all sentences containing keywords belonging to different classes but put them in corresponding sections (each class gets a separate section, which can be empty too, sections always occur in sorted order)

Table 14: 21 extracted elementary summarization subtasks and their descriptions

Domain	**Dataset name**	**Paper using the dataset**
News	CNN-Dailymail	See et al. [2017]
	NYT	Paulus et al. [2018]
	Gigaword	Paulus et al. [2018]
	XSUM	Liu and Lapata [2019]
	Newsroom	Zhang et al. [2020]
Code	Code to Documentation dataset	Iyer et al. [2016]
	Git diff to commit-message dataset	Allamanis et al. [2016]
Scientific Paper	Arxiv	Cohan et al. [2018]
	Pubmed	Cohan et al. [2018]
	ScisummNet	Yasunaga et al. [2019]
Patent	BigPatent	Sharma et al. [2019]
Instructional guides	Wikihow	Zhang et al. [2020]
Social media post	Reddit-TIFU	Zhang et al. [2020]
Email	AESLC	Zhang et al. [2020]
Bills	BillSum	Zhang et al. [2020]
Reviews	Amazon reviews	Gerani et al. [2019]
	Yelp reviews	Chu and Liu [2019]
	CNET reviews	Gerani et al. [2019]
KeyValue Attributes	Wikibio	Lebret et al. [2016]
	E2E dataset	Novikova et al. [2017]
Knowledge Graphs	DBPedia triples to Wikipedia	Vougiouklis et al. [2018]
	AMR to sentence dataset	Song et al. [2018]
	Agenda	Koncel-Kedziorski et al. [2019]
	WebNLG	Moryossef et al. [2019]
Numerical Table	Rotowire box-score	Puduppully et al. [2019]
Miscellaneous webpages	Wikisum	Liu et al. [2018]
Conversations	SamSum	Gliwa et al. [2019]
	AMI	Wang and Cardie [2013]

Table 15: Existing summarization datasets in various domains, along with corresponding papers that use them and came up during the search procedure to characterize elementary tasks in summarization

D Appendix for Chapter 5

D.1 The Role of Sentence Order in Pretraining Corpora

For virtually all pretrained models like BERT, ELECTRA, XLNet, the sentences in the pretraining corpora are ordered as they naturally occur in some document such as Wikipedia article. Devlin et al. [2019] mention in their work : *"It is critical to use a document-level corpus rather than a shuffled sentence-level corpus (...) in order to extract long contiguous sequences."* However, for many of our pretraining corpora made from downstream datasets, the sentence taken in order do not form a coherent document or narrative text. For example, in the MNLI or QQP corpora, neighboring sentences will simply be premise-hypothesis pairs or potential paraphrase candidates.

Despite the sentence order not forming a coherent document, many pretraining corpora achieve high performance boosts on the GLUE language understanding benchmark (Table 16). For example, MNLI achieves around 96% of the performance boost of the off-the-shelf model (Table 16). Interestingly, shuffling the sentences in these corpora leads to a large drop in performance (Table 16). This suggests that there is some value to keeping the sentence order in a way that puts sentences from the same example in datasets like MNLI and QQP next to each other. A likely explanation of this is in Levine et al. [2021] where authors showed that including similar sentences in the same input sequence when pretraining should lead to improved performance via theoretical analysis and empirical experiments.

We test if GLUE performance can be improved by artificially re-ordering a set of sentences to promote the occurrence of similar sentences together. We rearrange the sentences in the sentence-shuffled versions of pretraining corpora to encourage content overlap among neighboring sentences, and see if this can recover some of the drops in performance that occurred due to shuffling. Our algorithm creates the corpus by iteratively appending sentences to it, such that at each step the new sentence is the one with maximum TF-IDF similarity with the previous sentence. Such a way of constructing a corpus by similarity based retrieval has been used in past works [Levine et al., 2021, Yao et al., 2022], with the main difference that they retrieved sentences from external corpora similar to the ones present in the downstream dataset, whereas we simply use it to reorder sentences already present in the downstream dataset for pretraining We also make sure that the algorithm does not accidentally recover the original order of sentences (e.g. by matching the premise-hypothesis pairs originally in the MNLI dataset).

We experiment with 5 different datasets and find that the sentence-reordering scheme improves performance compared to random sentence order for all of them except QQP. For Discovery and DBPedia14 datasets, it scores even higher than our *standard* sentence ordering scheme which preserves the adjacency and order of sentences within each datapoint. This shows that re-ordering sentences to promote content similarity between neighboring sentences, can potentially improve GLUE score, without introducing any new information or narrative structure.

D.2 Experiments with smaller ELECTRA models

In addition to experimenting with a *base*-sized architecture (110M parameters), we also experiment with architectures which are even smaller than ELECTRA-small. We train ELECTRA models of smaller size by either reducing the number of layers in the generator and discriminator, or reducing the hidden dimension of the discriminator[5]. As the models get smaller, self-pretraining continues to significantly outperform random initialization and often outperforms pretraining on BookWiki corpus (Figure 5). Interestingly, the relative performance of self-pretrained and BookWiki-pretrained models tends to stay the same across model size. For example, for QQP self-pretraining is always best and for MNLI BookWiki-pretraining is always best irrespective of number of layers or hidden size.

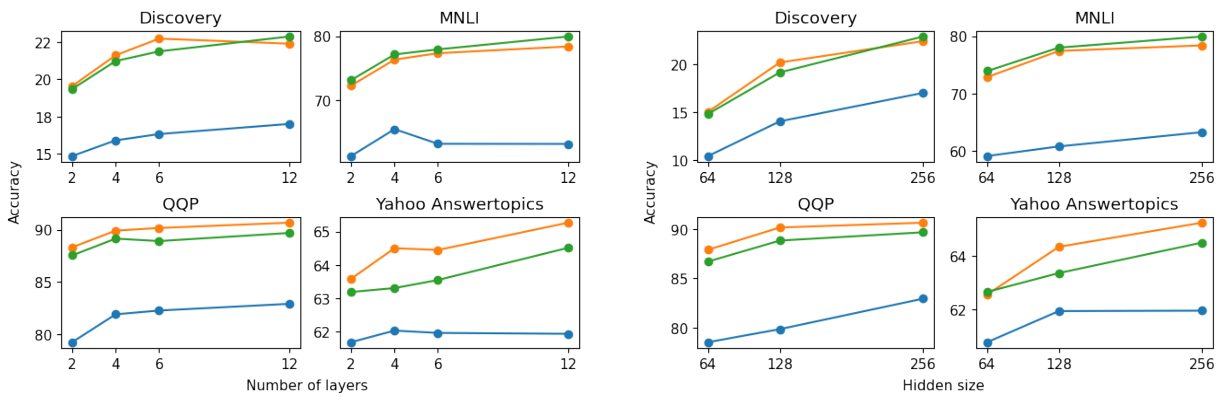


Figure 5: Variation in performance of ELECTRA models with change in number of layers and hidden size (— randomly initialized, — self-pretrained, — BookWiki-pretrained)

D.3 Implementation details for pretraining and finetuning

Hyperparameters for pretraining For pretraining ELECTRA-small models, we use the standard hyperparameters (Table 17) as described in Clark et al. [2019]. For the Roberta-base models, training with the standard hyperparameters with our computing resources would be prohibitively slow, and so we used hyperparameters from Warstadt et al. [2020] which require lesser time to train (Table 17). For task-adaptive pretraining(TAPT), we follow Gururangan et al. [2020] and further pretrain off-the-shelf models for 100 epochs on the downstream task's training set, with the first 6% of the resulting total updates used for learning rate warmup.

Hyperparameters for finetuning For finetuning the models on the 10 downstream datasets, we use hyperparameters as shown in Table 18. We use the AdamW optimizer [Loshchilov and Hutter, 2018] for finetuning. We use early stopping based on validation set performance. The validation metric used is mean squared error for the sentiment140 dataset (regression), average binary crossentropy for the jigsaw dataset (multi-label classification), and accuracy for all other datasets (multi-class classification). The patience parameter for early stopping is set to 3 epochs. For finetuning ELECTRA-small models on the GLUE datasets, we use the standard learning rate of 1e-4 following Clark et al. [2019].

[5]In ELECTRA, the generator's hidden size is already much smaller than that of the discriminator by design. So we do not reduce it further, in order to have a reasonably well-performing generator.

Details about use of downstream datasets All downstream datasets used in this paper were sourced from the Huggingface library[6]. For the Yahoo Answertopics dataset, we use only the text from the answer (not the question) as input to the models (both for pretraining and finetuning). For the PAWS dataset, we use the version called "Unlabeled PAWS$_{\text{wiki}}$" in Zhang et al. [2019], which is actually *not* unlabeled but has silver labels. We preferred that version over others because of its larger size. For datasets which had a train and test split but no validation split (e.g. Yahoo Answertopics), we extracted 5000 random datapoints from the the train split to make the validation split. If a dataset had a train and validation split but no test split (e.g. Unlabeled PAWS$_{\text{wiki}}$), we designated the validation split to be the test split, and created a new validation set by extracting 5000 random datapoints from the train set.

D.4 Software packages and hardware used

For pretraining ELECTRA models, we used Nvidia's implementation of the ELECTRA codebase[7], run using Nvidia's Tensorflow cotainer image 21.07 [8]. For pretraining Roberta models, we used the official implementation in the Fairseq library[9]. For finetuning experiments, we used the AllenNLP library for training and evaluation routines, coupled with the Huggingface library for the model architectures.

We used a collection of Nvidia V100 (32GB) and A6000(48GB) GPUs for our experiments. Pretraining an ELECTRA-small model takes around 1.5 days on 2 GPUs while pretraining a Roberta-base model takes around 1.5 days on 4 GPUs.

[6]https://huggingface.co/docs/datasets/index

[7]https://github.com/NVIDIA/DeepLearningExamples/tree/master/TensorFlow2/LanguageModeling/ELECTRA

[8]https://docs.nvidia.com/deeplearning/frameworks/tensorflow-release-notes/rel_21-07.html

[9]https://github.com/facebookresearch/fairseq

Pretraining Dataset	**Random**	**Standard**	**TF-IDF(Ours)**
None (RandomInit)	-	53.20	-
Sentiment140	-	72.67	75.29
DBpedia14	72.82	70.38	75.44
Discovery	71.79	77.26	78.94
MNLI	62.80	78.28	76.33
QQP	71.09	75.43	69.57
BookWiki (Off-the-shelf)	-	79.43	-

Table 16: GLUE scores achieved by different strategies for ordering sentences from the downstream dataset used for pretraining. Random: randomly ordered sentences; Standard: sentences within a datapoint occur contiguously in original order; TF-IDF: sentences reordered using content similarity.

Hyperparameter	**ELECTRA**	**Roberta**
Size (Parameter count)	Small (14M)	Base (110M)
Training steps	1M	100K
Warmup steps	10K	6K
Batch size	128	512
Peak learning rate	5e-4	5e-4
Sequence length	128	512

Table 17: Hyperparameters used for pretraining models

Hyperparameter	**ELECTRA**	**Roberta**
Training epochs	20	20
Batch size	32	32
Learning rate	{1e-4,1e-5}	2e-5
Max sequence length	512	512

Table 18: Hyperparameters used for finetuning models on 10 downstream tasks

E Appendix for Chapter 3

E.1 Sample datapoints for different tasks

We show a sample labeled datapoint for each task from the validation set of USB in Figure 8 and Figure 9.

E.2 Instructions used in model inputs

We list the instructions used in the inputs to Flan-T5 models in Table 19, Llama-13B in Table 20, Vicuna-13B in Table 21, and GPT-3.5-turbo in Table 22.

E.3 Implementation details for models

In this section we outline the architectures, and input/output formatting used for different models used in our experiments. Additionally, we report the hyperparameters used during training and inference for each model in Table 23.

Roberta For the Factuality Classification task, we feed in the evidence and summary separated by the SEP token into a standard classifier setup, which applies a linear layer with sigmoid activation on top of the CLS embedding. For Evidence Extraction, we use the same architecture and input individual pairs of a summary sentence with each source sentence to make a prediction for each of them. For the Extractive Summarization task, we use a hierarchical architecture identical to the one described as BERT-LSTM in Krishna et al. [2021b], except that we use a Roberta encoder instead of BERT. For Unsupported Span Prediction, we frame it as a sequence tagging problem where the given summary sentence and evidence are passed through Roberta and a linear layer with sigmoid predicts whether each token is supported or not. The consecutive positive predictions are concatenated to turn them into spans.

T5/Flan-T5 We preface each input with an instruction for the task to be done, followed by the text from the source/summary to be input. We frame the Evidence Extraction and Extractive Summarization tasks as a sequence of Yes/No predictions for each sentence in the source. Each source sentence in the input is prefixed by an enumerated sentence id (e.g. SENT34), and the ground truth target is the sequence of all sentence ids, with a Yes/No following each according to it's positive/negative label (e.g. "SENT0 Yes SENT1 No SENT2 No..."). Similarly, for Factuality Classification, the target is a single Yes/No based on the label. During inference, we measure the probabilities of generated Yes/No tokens which allows us to measure AUC scores too. For Unsupported Span Prediction, we generated the ground truth target by surrounding the unsupported spans in the summary with begin-span and end-span tags.

Llama/Vicuna For Llama and Vicuna we use the exact same input formatting. Compared to the Flan-T5 data formatting, we use a different set of instructions for these models, after trying out plausible variants for each task on the validation set. We provide 4 different instances as few-shot examples following the instruction in each datapoint for each task. The few-shot examples are chosen by sampling from the training set without replacement. Due to limitations in sequence length, we only use a maximum of 2048 tokens for the few-shot examples. For the tasks which require the full document in the input (i.e. ABS, EXT, EVEXT, TOPIC), we use 4 examples with each having a maximum of 512 tokens. For the remaining tasks, we use 16 examples each with a maximum length of 128 tokens. The few-shot examples are sampled (without replacement) from the training set while creating the prompt for each datapoint in the test set. Since these are decoder-only models which essentially generate plausible completions of the input string, we preface each output with a word (e.g. “SUMMARY:”, “LABELS:”) in the few-shot examples and at the end of the prompt to trigger the generation of the required summary/labels.

GPT-3.5-turbo The formatting of input and output is exactly the same as for Llama/Vicuna for all tasks except Evidence Extraction and Extractive Summarization. For these two tasks, we found that this model performed much better if we prompted it to generate the source sentence ids which should be assigned the positive label, instead of generating a Yes/No prediction for each source sentence. So we changed the output formatting in our few-shot examples accordingly. For this model too, we choose a different set of instructions for the tasks by experimenting with different options on the validation set.

E.4 Human evaluation of model outputs

It is well-acknowledged that ROUGE [Lin, 2004a] is an imperfect automatic metric to assess summary quality, and may not accurately reflect human preferences [Nenkova, 2006, Cohan and Goharian, 2016, Goyal et al., 2022]. Hence, we also conducted human evaluation for some tasks, where we show summaries generated by the best fine-tuned model (Flan-T5-XL) and the best fewshot-prompted LLM (GPT-3.5-turbo) and ask annotators to choose the better one along different dimensions (Table 7.3).

For the tasks of Abstractive Summarization (ABS), Multi-sentence Compression (COMP), and Topic-based Summarization (TOPIC), we collected annotations for 50 pairs of summaries, with 3 annotators rating each pair. For these 3 tasks, we did not screen workers based on qualification tasks since evaluating overall summary quality is a subjective task and it is better to have a diverse opinion from a large population, rather than a small set of manually selected people.

Evaluating model outputs for the Fixing Factuality (FIX) task is a more difficult but objective job. The increased difficulty comes from the need to carefully note the edits made by the models on the original incorrect summary and then decide on the factual validity and necessity of each edit. So we screened annotators via a qualification task on Mechanical Turk and selected 2 annotators to conduct the human evaluation for this specific task. Each pair of model outputs was rated by both annotators.

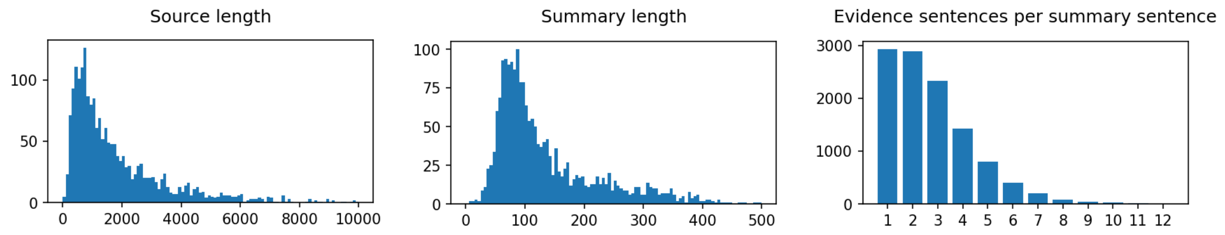


Figure 6: Distribution of number of words in the source and the summary, and the number of source sentences marked as evidence per summary sentence.

Task	Instruction
Multi-sentence Compression (COMP)	Summarize the following content in a single line.
Abstractive Summarization (ABS)	Summarize the following content.
Fixing Factuality (FIX)	Rewrite the given summary of the content to make it factually correct.
Unsupported Span Prediction (UNSUP)	Annotate parts of the summary which are not supported by evidence from the content.
Topic-based Summarization (TOPIC)	Summarize the given content for the following topic.
Factuality Classification (FAC)	Is there sufficient evidence for the summary in the content?
Extractive Summarization (EXT)	For each sentence, predict if it is important.
Evidence Extraction (EVEXT)	For each sentence in the content, predict if it provides any evidence for the claim.

Table 19: Instructions used in inputs to Flan-T5 models

Task	Instruction
Multi-sentence Compression (COMP)	Write a one-line summary of the content shown below.
Evidence Extraction (EVEXT)	Go over each sentence in the content, and decide if it supports the claim or not. Answer in Yes for a sentence if it supports the claim, and answer No otherwise.
Factuality Classification (FAC)	Is there sufficient evidence for the summary in the content?
Fixing Factuality (FIX)	Rewrite the given summary of the content to make it factually correct.
Abstractive Summarization (ABS)	Write a concise summary of the following paragraph
Topic-based Summarization (TOPIC)	Summarize the given content for the following topic.
Extractive Summarization (EXT)	For each sentence in the given content, label it as Yes if it is noteworthy enough to be included in a summary, or No otherwise.
Unsupported Span Prediction (UNSUP)	Regenerate the given summary, while surrounding those parts which do not have any supporting evidence in the content using [] and [/] tags

Table 20: Instructions used in inputs to the Llama-13B model

Task	Instruction
Multi-sentence Compression (COMP)	Write a single sentence summarizing the important points in the given content.
Evidence Extraction (EVEXT)	Predict which sentences in the given content can be used to infer facts in the claim.
Factuality Classification (FAC)	Decide if the following summary is consistent with the corresponding content. Note that consistency means all information in the summary is supported by the content. Explain your reasoning step by step then answer (yes or no) the question
Fixing Factuality (FIX)	Rewrite the following summary to make it factually accurate
Abstractive Summarization (ABS)	Draft a summary for the given document.
Topic-based Summarization (TOPIC)	Generate a summary of the given content covering the given topic.
Extractive Summarization (EXT)	For each sentence, predict if it is important.
Unsupported Span Prediction (UNSUP)	Annotate parts of the summary which are not supported by evidence from the content

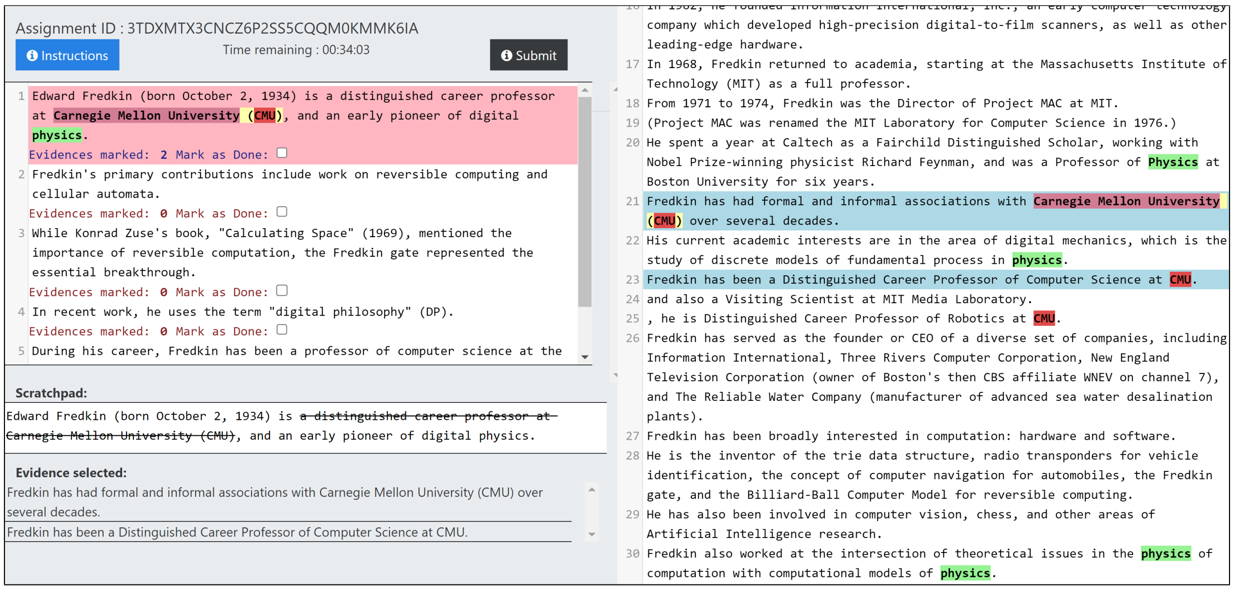
Table 21: Instructions used in inputs to the Vicuna-13B model

Task	Instruction
Multi-sentence Compression (COMP)	Summarize the following content in a single line.
Evidence Extraction (EVEXT)	Below is a claim along with its corresponding content. Identify and list all the sentences within the content that partially or entirely support the claim.
Factuality Classification (FAC)	Decide if the following summary is consistent with the corresponding content. Note that consistency means all information in the summary is supported by the content. Answer yes or no.
Fixing Factuality (FIX)	The summary might be incorrect. How would you rewrite it to make it factually accurate? Make as little changes as possible. Do not add any new information to the summary.
Abstractive Summarization (ABS)	Draft a summary for the given document.
Topic-based Summarization (TOPIC)	Create a short summary of the given content that touches upon information which fall under the specified topic.
Extractive Summarization (EXT)	For the task of extractive summarization, list all the SENTs of the content which would be included in its summary.
Unsupported Span Prediction (UNSUP)	Go over the given summary carefully, and regenerate it while surrounding any parts which are not supported by the content using [] and [/] tags

Table 22: Instructions used in inputs to the GPT-3.5-turbo model

Model	Task	Learning rate	Batch Size	Max input length	Max output length
Roberta-Large	FAC	1e-5	32	512	-
Roberta-Large	EXT	1e-5	32	$128\times128^{\psi}$	-
Roberta-Large	EVEXT	2e-5	2048	128	-
Roberta-Large	UNSUP	2e-5	32	512	-
T5-Large	(All)	5e-5	32	8192	768
FlanT5-Large	(All)	5e-5	32	8192	768
FlanT5-XL	(All)	5e-5	64	1536	512
Llama-13B	(All)	-	-	6144	512
Vicuna-13B	(All)	-	-	6144	512
GPT-3.5-turbo	(All)	-	-	Variable$^{\phi}$	Variable$^{\phi}$

Table 23: Hyperparameters used for training and inference with different models. ψ: 128 sentences each with maximum of 128 tokens fed into a hierarchical model. ϕ: GPT-3.5-turbo has a relatively small limit of 4096 tokens including both the input (with few-shot examples) and the output, and so we truncate the input on a per-task basis to leave token budget equal to the maximum output length in the train split for that task.


Assignment ID : 3TDXMTX3CNCZ6P2SS5CQQM0KMMK6IA

1 Edward Fredkin (born October 2, 1934) is a distinguished career professor at **Carnegie Mellon University** (**CMU**), and an early pioneer of digital **physics**.
Evidences marked: 2 Mark as Done: ☐
2 Fredkin's primary contributions include work on reversible computing and cellular automata.
Evidences marked: 0 Mark as Done: ☐
3 While Konrad Zuse's book, "Calculating Space" (1969), mentioned the importance of reversible computation, the Fredkin gate represented the essential breakthrough.
Evidences marked: 0 Mark as Done: ☐
4 In recent work, he uses the term "digital philosophy" (DP).
Evidences marked: 0 Mark as Done: ☐
5 During his career, Fredkin has been a professor of computer science at the

Scratchpad:
Edward Fredkin (born October 2, 1934) is ~~a distinguished career professor at Carnegie Mellon University (CMU)~~, and an early pioneer of digital physics.

Evidence selected:
Fredkin has had formal and informal associations with Carnegie Mellon University (CMU) over several decades.
Fredkin has been a Distinguished Career Professor of Computer Science at CMU.

16 In 1962, he founded Information International, Inc., an early computer technology company which developed high-precision digital-to-film scanners, as well as other leading-edge hardware.
17 In 1968, Fredkin returned to academia, starting at the Massachusetts Institute of Technology (MIT) as a full professor.
18 From 1971 to 1974, Fredkin was the Director of Project MAC at MIT.
19 (Project MAC was renamed the MIT Laboratory for Computer Science in 1976.)
20 He spent a year at Caltech as a Fairchild Distinguished Scholar, working with Nobel Prize-winning physicist Richard Feynman, and was a Professor of **Physics** at Boston University for six years.
21 Fredkin has had formal and informal associations with **Carnegie Mellon University** (**CMU**) over several decades.
22 His current academic interests are in the area of digital mechanics, which is the study of discrete models of fundamental process in **physics**.
23 Fredkin has been a Distinguished Career Professor of Computer Science at **CMU**.
24 and also a Visiting Scientist at MIT Media Laboratory.
25 , he is Distinguished Career Professor of Robotics at **CMU**.
26 Fredkin has served as the founder or CEO of a diverse set of companies, including Information International, Three Rivers Computer Corporation, New England Television Corporation (owner of Boston's then CBS affiliate WNEV on channel 7), and The Reliable Water Company (manufacturer of advanced sea water desalination plants).
27 Fredkin has been broadly interested in computation: hardware and software.
28 He is the inventor of the trie data structure, radio transponders for vehicle identification, the concept of computer navigation for automobiles, the Fredkin gate, and the Billiard-Ball Computer Model for reversible computing.
29 He has also been involved in computer vision, chess, and other areas of Artificial Intelligence research.
30 Fredkin also worked at the intersection of theoretical issues in the **physics** of computation with computational models of **physics**.

Figure 7: Screenshot of the interface used for collecting annotations. The summary is shown on the left and the source on the right. Entities in the active summary line are highlighted to help find evidence quickly. A scratchpad is provided where users can keep track of the parts of the summary for which evidence has been marked.

Evidence Extraction					
	Accuracy	AUC	F1	Precision	Recall
SuperPAL [Ernst et al., 2021]	98.1	95.8	53.8	82.1	40.0
ROUGE [Chen and Bansal, 2018]	95.9	88.5	40.9	33.7	52.1
Entity overlap	95.7	92.5	47.0	35.6	69.4
Human annotations 100% (N=765)	98.8	99.0	77.7	77.0	78.4
Human annotations 20%	98.7	98.4	74.7	78.9	70.8
Human annotations 10%	98.5	98.1	72.4	73.0	71.8
Human annotations 5%	98.4	97.7	70.9	70.8	70.9
Factuality Classification					
	Accuracy	AUC	F1	Precision	Recall
FactEdit [Balachandran et al., 2022]	55.7	74.6	29.4	72.6	18.4
FactCC [Kryściński et al., 2020]	52.9	68.9	20.1	66.3	11.8
Human annotations 100%	88.1	95.1	87.5	92.3	83.2
Human annotations 20%	86.7	93.9	86.1	90.6	82.0
Human annotations 10%	83.4	91.8	81.6	91.7	73.5
Human annotations 5%	82.6	90.4	82.5	83.2	81.7
Fix factuality					
	Exact Match	Rouge-1	Rouge-2	Rouge-L	
FactEdit [Balachandran et al., 2022]	1.0	81.6	73.0	81.0	
FactCC [Kryściński et al., 2020]	0.8	81.9	73.6	81.4	
Human annotations 100%	32.9	91.9	86.5	91.4	
Human annotations 20%	28.8	90.3	84.3	89.8	
Human annotations 10%	15.3	85.7	78.5	85.1	
Human annotations 5%	11.2	83.9	76.1	83.3	

Table 24: Comparision between using human annotations vs heuristic annotations for training models—Flan-T5-Large. We also report performance when finetuning on smaller fractions of the training set with human annotations.

Abstractive Summarization (ABS)	INPUT	**DOCUMENT:** D'Vauntes Smith-Rivera High school career Smith-Rivera started high school at North Central High School in Indianapolis, and led his team to a state championship in his sophomore year. He transferred to the basketball specialty Oak Hill Academy in Virginia for his senior year, and he helped lead the team to the 2012 national championship He was recruited by Xavier, UCLA, Louisville, Memphis, NC State, and Georgetown. …
	TARGET	D'Vauntes Smith-Rivera is a professional basketball player who last played for Koroivos of the Greek Basket League. He played high school basketball for North Central in Indianapolis and Oak Hill Academy in Virginia. …
Multi-sentence Compression (COMP)	INPUT	**SOURCE SENTENCES:** Odenkirk was hired as a writer at "Saturday Night Live" in 1987 and worked there through 1991. Odenkirk's friendship with Ben Stiller, with whom he briefly shared an office at "SNL", would lead to his being hired for the cast of "The Ben Stiller Show" in 1992. Working as both a writer and actor on the show, he created and starred in the memorable sketch "Manson Lassie", and helped the show win an Emmy Award for writing.
	TARGET	From the late 1980s to 1990s, Odenkirk wrote for television shows "Saturday Night Live" and "The Ben Stiller Show", winning an Emmy Award for writing.
Extractive Summarization (EXT)	INPUT	**DOCUMENT:** **SENT0:** D'Vauntes Smith-Rivera **SENT1:** High school career **SENT2:** Smith-Rivera started high school at North Central High School in Indianapolis, and led his team to a state championship in his sophomore year. **SENT3:** He transferred to the basketball specialty Oak Hill Academy in Virginia for his senior year, and he helped lead the team to the 2012 national championship. **SENT4:** He was recruited by Xavier, UCLA, Louisville, Memphis, NC State, and Georgetown. …
	TARGET	SENT0 SENT2 SENT4…
Topic-based Summarization (TOPIC)	INPUT	**DOCUMENT:** Arkema S.A. Arkema was created when French oil major Total restructured its chemicals business. The restructuring was a gradual process that began many years earlier: … **TOPIC NAME:** Organization
	TARGET	Arkema is organized into three business segments: Coating Solutions, Industrial Chemicals, and Performance Products.

Figure 8: Sample input-output pairs for different tasks from the validation set of USB

Factuality Classification (FAC)	INPUT	**EVIDENCE:** In 2014 YG also expanded into the beauty industry with the creation of its cosmetics brand Moonshot. YG Plus Inc., previously named Phoenix Holdings Inc., is a publicly traded media and advertising company acquired by YG Entertainment in November 2014. **SUMMARY:** In addition, the company operates a number of subsidiary ventures under a separate public traded company, YG Plus, which includes a clothing line, a golf management agency, and a cosmetics brand.
	TARGET	Incorrect
Unsupported Span Prediction (UNSUP)	INPUT	**EVIDENCE:** David Martin McIntosh McIntosh was born in Oakland, California, the son of Jean Marie (Slough), a judge, and Norman McIntosh. He graduated with a B.A. (cum laude) in 1980, and later received a J.D. from University of Chicago Law School in 1983. … Incumbent Democrat U.S. Congressman Philip Sharp of Indiana's 2nd congressional district decided to retire. McIntosh decided to run and won the Republican primary with a plurality of 43% in a four candidate field. In the general election, he defeated Democratic Secretary of State of Indiana Joe Hogsett 54%-46%. **SUMMARY:** David Martin McIntosh (born June 8, 1958) is an American attorney and Republican Party politician who served as the U.S. representative for Indiana's 2nd congressional district from 1995 to 2001.
	TARGET	David Martin McIntosh ~~(born June 8 , 1958)~~ is an American attorney and Republican Party politician who served as the U.S. representative for Indiana 's 2nd congressional district ~~from 1995 to 2001~~ .
Fixing Factuality (FIX)	INPUT	**EVIDENCE:** In 2009, Jordan returned to the F1 scene as a pundit for BBC Sport F1 coverage alongside Jake Humphrey (who was later replaced by Suzi Perry) and David Coulthard. In March 2016 he was announced as Channel 4's lead analyst for C4F1. **SUMMARY:** He was the chief analyst for Formula One coverage on the BBC from 2009 to 2015 before joining Channel 4 after BBC pulled out in 2016.
	TARGET	He was the a pundit for Formula One coverage on the BBC from 2009 before joining Channel 4 in 2016.
Evidence Extraction (EVEXT)	INPUT	**DOCUMENT:** **SENT0:** 2012 Istanbul rally to commemorate the Khojaly massacre **SENT1:** "Justice for Khojaly" campaign. **SENT2:** "Justice for Khojaly", or "JFK" for short, is an International Awareness Campaign, initiated on 8 May 2008 under the motto of "Justice for Khojaly, Freedom for Karabakh". … **SENT6:** Around 200,000 participants for the 20th anniversary remembrance of the Khojaly Massacre victims, dozens of youth and student organizations, public unions, Turkish organizations and movements participated in the rally. … **SENT17:** Various slogans included, "We are all from Khojaly", "Stop Armenian aggression", "Do not forget Turkic people genocide by Armenian gangs in southern Azerbaijan", "One nation, two countries, Justice for Khojaly!", and "Stop Armenian lies". … **SUMMARY:** The demonstration with slogan "We are all from Khojaly" had around 200,000 participants.
	TARGET	SENT6 SENT17

Figure 9: Sample input-output pairs for different tasks from the validation set of USB

F Appendix for Chapter 7

F.1 Details on human evaluation

We engaged annotators via Upwork[10], leveraging their expertise to evaluate model-generated summaries against source documents. Candidates were chosen through a qualifying round, focusing on their ability to identify inaccuracies in summaries. Ultimately, two proficient proofreaders and fact-checkers were selected based on their performance on the qualifying task. Each annotator was tasked with annotating all summaries for half of the documents in each of the 3 datasets used (see Section 7.4). Annotators received compensation at an average rate of $25 USD per hour for their contributions.

We use a slightly modified version of the GenAudit UI to collect the annotations, with two notable changes (Figure 10). First, we add a buttons next to every source sentence to provide a accept/reject feedback if the source sentence is highlighted as evidence for a summary sentence. For source sentences which are not highlighted as evidence, we provide an accept button to mark it as additional evidence if needed. Second, the UI enables cycling through multiple model-generated summaries for the same source document at once. This is done to save annotators' time, since otherwise the annotators would have to read the source document again each time they get a summary for it in the sequence of annotation jobs. The annotators were instructed to mark the following categories of generated summary sentences as *invalid* which we excluded from our analysis: (i) truncated sentences, which occur due to the maximum decode length limit being reached. (ii) incomprehensible sentences, such as when the models generate instructions instead of summary (e.g. Falcon-7B once generated *"when creating a summary, use the information given and avoid superfluous details."*)

F.2 Annotator Instructions

The following protocol was provided to annotators for assessing both the supporting evidence for summary sentences, and their factual consistency:

Evidence Annotation:

a) Evaluate each summary sentence by reviewing all linked evidence from the source, marking them as either accepted (by clicking a tick) or rejected (by clicking a cross) based on their validity.

b) Examine the summary for unsupported information with respect to suggested evidence, If any information supporting the summary is found in the source but is not already supported by the suggested evidence, mark it as "new" evidence. Note that "new" evidence pertains only to instances where the summary includes seemingly unsupported information compared to the suggested evidence.

Summary Annotation:

*a) First, accept all suggestions made by the tool that are **valid**. This encompasses deletions made by the tool of unsupported information and any recommended edits or replacements.*

[10] https://www.upwork.com/

b) If there is incorrect or contradictory information in the summary not satisfactorily addressed by the tool then make minimal edits to the summary to align it with the source. If the edit is based on source information not already included in suggested evidence, label it as "new evidence."

c) For cases where the summary introduces unsupported information not addressed by the tool, remove corresponding segments of the summary without altering the evidence.

Note if a summary sentence is incomplete or incomprehensible, mark the sentence as INVALID.

F.3 Inter-annotator agreement

For one-third of the documents in each dataset, we got their model-generated summaries annotated by both annotators to estimate inter-annotator agreement. The doubly-annotated data included 232 summaries consisting of a total of 989 sentences. We compute the agreement on error identification by comparing the words removed/replaced from the initial summary by the two annotators. The value of Cohen's Kappa for this is 62.7 indicating substantial agreement. Similarly, we compare the ratings (useful vs not) provided to each suggested evidence sentence by the two annotators, which yields a Cohen's Kappa value of 58.56 indicating moderate agreement.

F.4 Derivation of baseline precision-recall trade-off

Assume a binary classification problem with a dataset of T datapoints, out of which P have positive labels and N have negative labels. Let's assume a prediction model assigns positive labels to P' datapoints, and achieves a recall of α and precision of β. We want to boost the recall to a target of α' by flipping the predictions from negative to positive for some datapoints. In this section we derive the precision-recall trade-off achieved if we select the labels to flip uniformly at random, which is the baseline used in Section 7.5.

If we flip the labels of k datapoints from negative to positive, the expected number of them which would be true positives will be

$$\Delta = \frac{P(1-\alpha)}{(T-P')}k$$

For an expected target recall of α', we get

$$\alpha' = \frac{\alpha P + \Delta}{P}$$

$$\implies \alpha' = \alpha + \frac{k(1-\alpha)}{(T-P')}$$

$$\implies k = \frac{\alpha' - \alpha}{1-\alpha}(T-P')$$

The expected true positives is $\alpha' P$, which yields the new expected precision β' to be

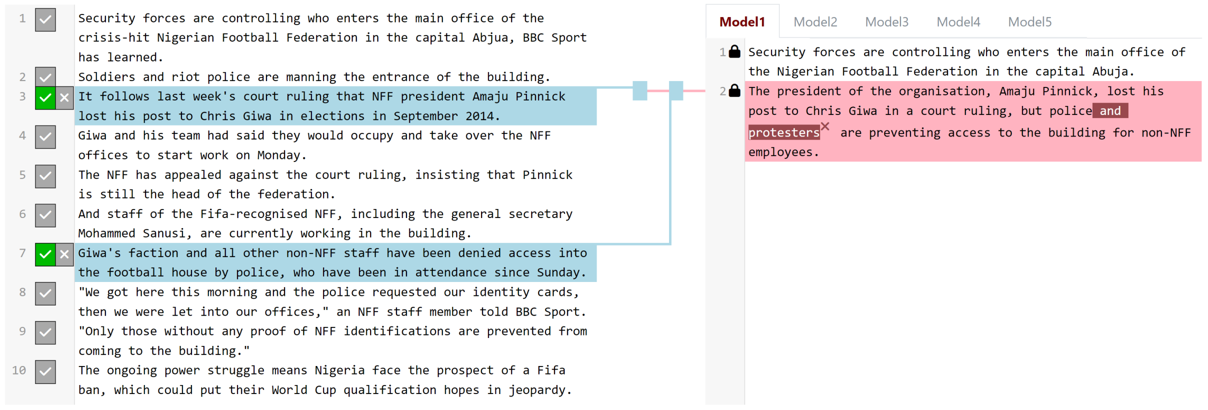
$$\beta' = \frac{\alpha' P}{(P'+k)}$$

$$\implies \beta' = \frac{\alpha' P}{P' + \frac{(\alpha'-\alpha)}{(1-\alpha)}(T - P')}$$

Note that $P = \gamma T$, where γ is the base rate for positive class, and the number of predicted positive labels $P' = \frac{\alpha P}{\beta}$. Substituting these, we get our final form

$$\beta' = \frac{\alpha' \gamma}{\frac{\alpha\gamma}{\beta} + \frac{(\alpha'-\alpha)}{(1-\alpha)}(1 - \frac{\alpha\gamma}{\beta})}$$

We showed the performance of this baseline against our proposed Algorithm 1 in Figure 7.2.



1 Security forces are controlling who enters the main office of the crisis-hit Nigerian Football Federation in the capital Abjua, BBC Sport has learned.
2 Soldiers and riot police are manning the entrance of the building.
3 It follows last week's court ruling that NFF president Amaju Pinnick lost his post to Chris Giwa in elections in September 2014.
4 Giwa and his team had said they would occupy and take over the NFF offices to start work on Monday.
5 The NFF has appealed against the court ruling, insisting that Pinnick is still the head of the federation.
6 And staff of the Fifa-recognised NFF, including the general secretary Mohammed Sanusi, are currently working in the building.
7 Giwa's faction and all other non-NFF staff have been denied access into the football house by police, who have been in attendance since Sunday.
8 "We got here this morning and the police requested our identity cards, then we were let into our offices," an NFF staff member told BBC Sport.
9 "Only those without any proof of NFF identifications are prevented from coming to the building."
10 The ongoing power struggle means Nigeria face the prospect of a Fifa ban, which could put their World Cup qualification hopes in jeopardy.

1 Security forces are controlling who enters the main office of the Nigerian Football Federation in the capital Abuja.
2 The president of the organisation, Amaju Pinnick, lost his post to Chris Giwa in a court ruling, but police and protesters ✕ are preventing access to the building for non-NFF employees.

Figure 10: Interface used for collecting feedback on suggested evidence and edits from GENAUDIT. Annotators can accept/reject each suggested evidence sentence, and can also mark additional sentences as evidence if needed. Suggested edits can be accepted/rejected by clicking on the button on the top-right of the highlighted span. if needed, users can also make freeform edits to fix more errors. Annotators cycle through the summaries generated by different models, whose names are anonymized and their order is shuffled.

F.5 Training details

Each model fine-tuned on the USB dataset, was trained for 10 epochs and the best checkpoint was selected based on validation performance at the end of each epoch. The effective batch size was kept at 128. The optimizer used was 8-bit AdamW with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, without weight decay, and with a learning rate of $5e - 5$. The maximum input and output sequence lengths were set to 3050 and 150 tokens respectively. We used 4-bit quantization with low-rank adapters [Dettmers et al., 2023] with parameters $r = 8$ and $\alpha = 32$.

Each training run was carried out on $2\times$ Nvidia A6000 GPUs each of which has 49GB of memory. We used gradient accumulation and gradient checkpointing to reduce the peak memory requirements during training. The implementation was done using the Huggingface transformers [Wolf et al., 2019], Deepspeed [Rasley et al., 2020], Peft [Mangrulkar et al., 2022] and Bitsandbytes[11] libraries.

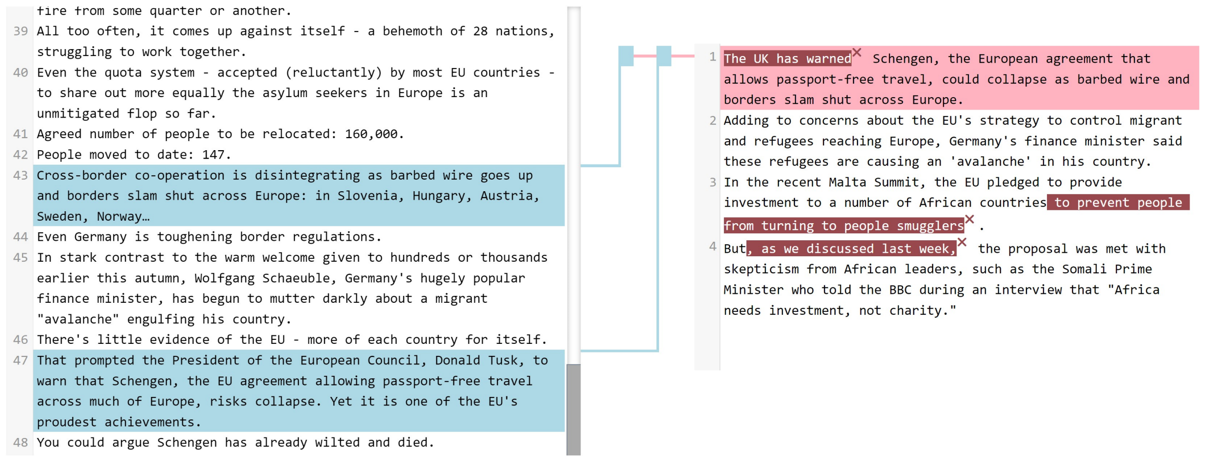
[11]https://github.com/TimDettmers/bitsandbytes

Model	URL
Llama-7B	https://huggingface.co/meta-llama/Llama-2-7b-chat-hf
Llama-70B	https://huggingface.co/meta-llama/Llama-2-70b-chat-hf
Mistral-7B	https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1
Falcon-7B	https://huggingface.co/tiiuae/falcon-7b-instruct
Flan-T5-XL	https://huggingface.co/google/flan-t5-xl
Flan-T5-XXL	https://huggingface.co/google/flan-t5-xxl
Flan-UL2	https://huggingface.co/google/flan-ul2
GPT-3.5-turbo	https://platform.openai.com/docs/models/gpt-3-5-turbo (version gpt-3.5-turbo-16k-0613)
GPT-4	https://platform.openai.com/docs/models/gpt-4-and-gpt-4-turbo (gpt-4-0613)

Table 25: Links to models used in Section 7.3 and Section 7.4

F.6 Sample model outputs

We show some sample corrections suggested by GENAUDIT using the Flan-UL2 backend to summaries generated by LLMs in Figures 11, 12, 13, 14.



fire from some quarter or another.
39 All too often, it comes up against itself - a behemoth of 28 nations, struggling to work together.
40 Even the quota system - accepted (reluctantly) by most EU countries - to share out more equally the asylum seekers in Europe is an unmitigated flop so far.
41 Agreed number of people to be relocated: 160,000.
42 People moved to date: 147.
43 Cross-border co-operation is disintegrating as barbed wire goes up and borders slam shut across Europe: in Slovenia, Hungary, Austria, Sweden, Norway…
44 Even Germany is toughening border regulations.
45 In stark contrast to the warm welcome given to hundreds or thousands earlier this autumn, Wolfgang Schaeuble, Germany's hugely popular finance minister, has begun to mutter darkly about a migrant "avalanche" engulfing his country.
46 There's little evidence of the EU - more of each country for itself.
47 That prompted the President of the European Council, Donald Tusk, to warn that Schengen, the EU agreement allowing passport-free travel across much of Europe, risks collapse. Yet it is one of the EU's proudest achievements.
48 You could argue Schengen has already wilted and died.

1 ~~The UK has warned~~ Schengen, the European agreement that allows passport-free travel, could collapse as barbed wire and borders slam shut across Europe.
2 Adding to concerns about the EU's strategy to control migrant and refugees reaching Europe, Germany's finance minister said these refugees are causing an 'avalanche' in his country.
3 In the recent Malta Summit, the EU pledged to provide investment to a number of African countries ~~to prevent people from turning to people smugglers~~ .
4 But ~~, as we discussed last week,~~ the proposal was met with skepticism from African leaders, such as the Somali Prime Minister who told the BBC during an interview that "Africa needs investment, not charity."

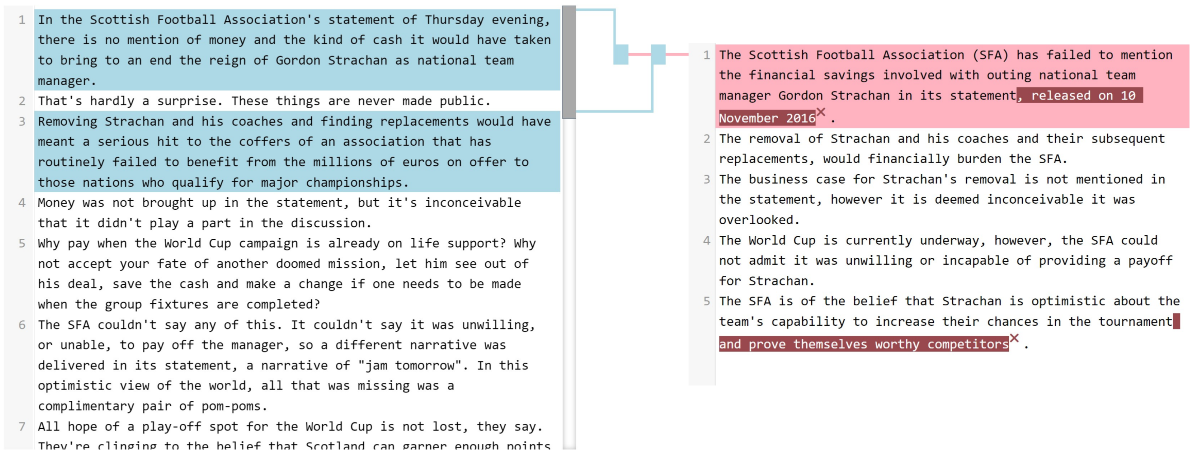
Figure 11: GENAUDIT suggested edits for a GPT4-generated summary of a news article from the XSum dataset. The first sentence contains a statement attributed to the UK which was actually made by the president of the European Union

Model	Prompt
GPT-4	You are provided a document and its summary. The summary may potentially contain facts which contradict with the document or are not supported by any evidence in the document. The last sentence of the summary is marked as a claim. Find and list sufficient sentences in the document to provide evidence for the claim. Make sure to provide evidence for all the supported facts in the claim. Then, revise the claim to remove or replace facts which are not supported by the document or are contradicted by it. Only make changes to the text of the claim when necessary. When you add new information to the claim, it must be only to fix a contradictory fact in the claim, and not for changing the style of the text.
GPT-3.5-turbo	You are provided a document and its summary. The summary may potentially contain facts which contradict with the document or are not supported by any evidence in the document. The last sentence of the summary is marked as a claim. Find and list sufficient sentences in the document to provide evidence for the claim, and then revise the claim to remove or replace facts which are not supported by the document or are contradicted by it. When you add new information to the claim, it must be only to fix a contradictory fact in the claim, and not for changing the style of the text.
Others	You are provided a document and its summary. The summary may potentially contain factual errors. The last sentence of the summary is marked as a claim. Find all sentences in the document providing evidence for the claim, and then revise the claim to remove or replace unsupported facts.

Table 26: Prompts used for fact-checking using GPT models in zero-shot setting, and using other models with fine-tuning (as described in Section 7.3)

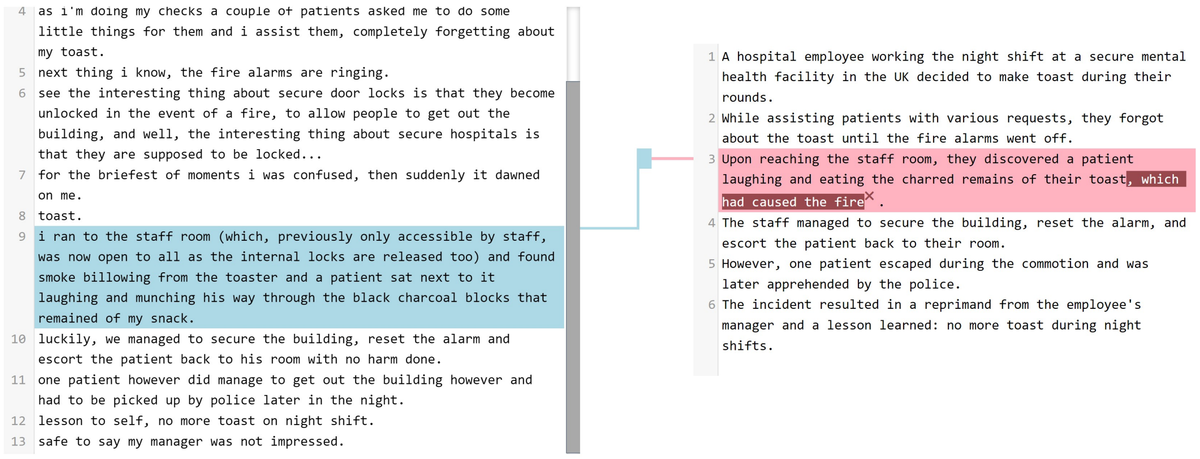
Model	Prompt
Gemini-pro, GPT-3.5-turbo, GPT-4	Generate a summary for the following document in brief. When creating the summary, only use information that is present in the document. Generate the summary in free-form text without using bullet points.
Others	Generate a summary for the following document in brief. When creating the summary, only use information that is present in the document.

Table 27: Prompts used for summary generation using different models for human evaluation experiment (Section 7.4)


1 In the Scottish Football Association's statement of Thursday evening, there is no mention of money and the kind of cash it would have taken to bring to an end the reign of Gordon Strachan as national team manager.
2 That's hardly a surprise. These things are never made public.
3 Removing Strachan and his coaches and finding replacements would have meant a serious hit to the coffers of an association that has routinely failed to benefit from the millions of euros on offer to those nations who qualify for major championships.
4 Money was not brought up in the statement, but it's inconceivable that it didn't play a part in the discussion.
5 Why pay when the World Cup campaign is already on life support? Why not accept your fate of another doomed mission, let him see out of his deal, save the cash and make a change if one needs to be made when the group fixtures are completed?
6 The SFA couldn't say any of this. It couldn't say it was unwilling, or unable, to pay off the manager, so a different narrative was delivered in its statement, a narrative of "jam tomorrow". In this optimistic view of the world, all that was missing was a complimentary pair of pom-poms.
7 All hope of a play-off spot for the World Cup is not lost, they say. They're clinging to the belief that Scotland can garner enough points

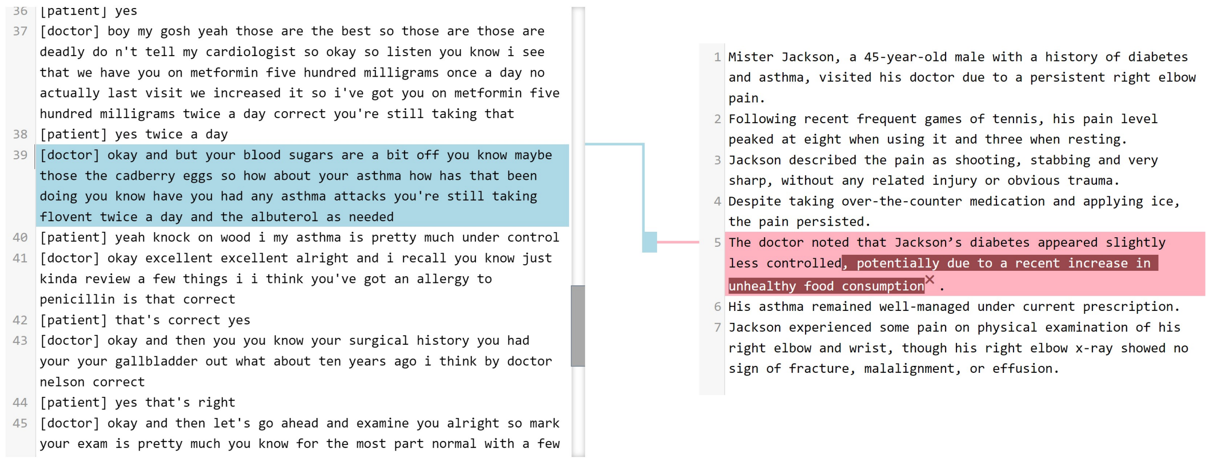
1 The Scottish Football Association (SFA) has failed to mention the financial savings involved with outing national team manager Gordon Strachan in its statement, released on 10 November 2016 ✕ .
2 The removal of Strachan and his coaches and their subsequent replacements, would financially burden the SFA.
3 The business case for Strachan's removal is not mentioned in the statement, however it is deemed inconceivable it was overlooked.
4 The World Cup is currently underway, however, the SFA could not admit it was unwilling or incapable of providing a payoff for Strachan.
5 The SFA is of the belief that Strachan is optimistic about the team's capability to increase their chances in the tournament and prove themselves worthy competitors ✕ .

Figure 12: GENAUDIT suggested edits for a GPT4-generated summary of a news article from the XSum dataset. The reference document does not contain the date on which the said statement was made by SFA. Interestingly though, GPT-4 almost got it right by sheer memorization. The statement was released on 17 November 2016 whereas GPT-4 mentioned 10 November 2016. (Ref: https://www.bbc.com/sport/football/38021627,https://www.bbc.com/sport/football/38019477)


4 as i'm doing my checks a couple of patients asked me to do some little things for them and i assist them, completely forgetting about my toast.
5 next thing i know, the fire alarms are ringing.
6 see the interesting thing about secure door locks is that they become unlocked in the event of a fire, to allow people to get out the building, and well, the interesting thing about secure hospitals is that they are supposed to be locked...
7 for the briefest of moments i was confused, then suddenly it dawned on me.
8 toast.
9 i ran to the staff room (which, previously only accessible by staff, was now open to all as the internal locks are released too) and found smoke billowing from the toaster and a patient sat next to it laughing and munching his way through the black charcoal blocks that remained of my snack.
10 luckily, we managed to secure the building, reset the alarm and escort the patient back to his room with no harm done.
11 one patient however did manage to get out the building however and had to be picked up by police later in the night.
12 lesson to self, no more toast on night shift.
13 safe to say my manager was not impressed.

1 A hospital employee working the night shift at a secure mental health facility in the UK decided to make toast during their rounds.
2 While assisting patients with various requests, they forgot about the toast until the fire alarms went off.
3 Upon reaching the staff room, they discovered a patient laughing and eating the charred remains of their toast, which had caused the fire ✕ .
4 The staff managed to secure the building, reset the alarm, and escort the patient back to their room.
5 However, one patient escaped during the commotion and was later apprehended by the police.
6 The incident resulted in a reprimand from the employee's manager and a lesson learned: no more toast during night shifts.

Figure 13: GENAUDIT suggested edits for a Geminipro-generated summary of a Reddit post where a person describes fire alarm going off at a workplace due to smoke from burnt toast. The summary suggests that there was a fire caused which doesn't seem to be the case from the reference.



[patient] yes
[doctor] boy my gosh yeah those are the best so those are those are deadly do n't tell my cardiologist so okay so listen you know i see that we have you on metformin five hundred milligrams once a day no actually last visit we increased it so i've got you on metformin five hundred milligrams twice a day correct you're still taking that
[patient] yes twice a day
[doctor] okay and but your blood sugars are a bit off you know maybe those the cadberry eggs so how about your asthma how has that been doing you know have you had any asthma attacks you're still taking flovent twice a day and the albuterol as needed
[patient] yeah knock on wood i my asthma is pretty much under control
[doctor] okay excellent excellent alright and i recall you know just kinda review a few things i i think you've got an allergy to penicillin is that correct
[patient] that's correct yes
[doctor] okay and then you you know your surgical history you had your your gallbladder out what about ten years ago i think by doctor nelson correct
[patient] yes that's right
[doctor] okay and then let's go ahead and examine you alright so mark your exam is pretty much you know for the most part normal with a few

1 Mister Jackson, a 45-year-old male with a history of diabetes and asthma, visited his doctor due to a persistent right elbow pain.
2 Following recent frequent games of tennis, his pain level peaked at eight when using it and three when resting.
3 Jackson described the pain as shooting, stabbing and very sharp, without any related injury or obvious trauma.
4 Despite taking over-the-counter medication and applying ice, the pain persisted.
5 The doctor noted that Jackson's diabetes appeared slightly less controlled~~, potentially due to a recent increase in unhealthy food consumption~~ ✕ .
6 His asthma remained well-managed under current prescription.
7 Jackson experienced some pain on physical examination of his right elbow and wrist, though his right elbow x-ray showed no sign of fracture, malalignment, or effusion.

Figure 14: GenAudit suggested edits for a GPT4-generated summary of a conversation between a doctor and patient. Here, the doctor briefly mentions that the patient's blood sugar problems may be caused by eating chocolates, but they don't suggest that such unhealthy consumption has increased recently (as the summary claims).

Bibliography

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 3, 6, 69, 82

Griffin Adams, Jason Zuckerg, and Noémie Elhadad. A meta-evaluation of faithfulness metrics for long-form hospital-course summarization. In *Machine Learning for Healthcare Conference*, pages 2–30. PMLR, 2023. 67

Jaimeen Ahn and Alice Oh. Mitigating language-dependent ethnic bias in bert. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 533–549, 2021. 3, 52

Nadeem Akhtar, Nashez Zubair, Abhishek Kumar, and Tameem Ahmad. Aspect based sentiment oriented summarization of hotel reviews. *Procedia Computer Science*, 115(C):563–571, 2017. 65

Miltiadis Allamanis, Hao Peng, and Charles Sutton. A convolutional attention network for extreme summarization of source code. In *International conference on machine learning*, pages 2091–2100. PMLR, 2016. 100

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, et al. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*, 2023. 72, 73

Philip Arthur, Graham Neubig, and Satoshi Nakamura. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2016. 78

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023. 82

Vidhisha Balachandran, Hannaneh Hajishirzi, William Cohen, and Yulia Tsvetkov. Correcting diverse factual errors in abstractive summarization via post-editing and language model infilling. *arXiv preprint arXiv:2210.12378*, 2022. 53, 54, 55, 63, 64, 65, 112

Joshua Bambrick, Minjie Xu, Andy Almonte, Igor Malioutov, Guim Perarnau, Vittorio Selo, and Iat Chong Chan. Nstm: Real-time query-driven news overview composition at bloomberg. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics:*

System Demonstrations, pages 350–361, 2020. 2

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, 2021. 3

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 481–490. Association for Computational Linguistics, 2011. 10

Rishi Bommasani and Claire Cardie. Intrinsic evaluation of summarization datasets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8075–8096, 2020. 64

Shikha Bordia and Samuel Bowman. Identifying and reducing gender bias in word-level language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 7–15, 2019. 34

Ronald Brandow, Karl Mitze, and Lisa F Rau. Automatic condensation of electronic publications by sentence selection. *Information Processing & Management*, 31(5):675–685, 1995. 36

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 3, 52

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023. 2

Meng Cao, Yue Dong, Jiapeng Wu, and Jackie Chi Kit Cheung. Factual error correction for abstractive summarization models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6251–6258, 2020. 53

Meng Cao, Yue Dong, and Jackie Chi Kit Cheung. Hallucinated but factual! inspecting the factuality of hallucinations in abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3340–3354, 2022. 83

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. Faithful to the original: Fact aware neural abstractive summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 78

Jean Carletta. Unleashing the killer corpus: experiences in creating the multi-everything ami meeting corpus. *Language Resources and Evaluation*, 41(2):181–190, 2007. 1, 9, 10

Nicholas Carlini, Florian Tramer, Eric Wallace, et al. Extracting training data from large language models. 2021. 3, 33

Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. *arXiv preprint arXiv:2301.13188*, 2023. 3

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 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)*, pages 1662–1675, 2018. 10

Sky CH-Wang, Benjamin Van Durme, Jason Eisner, and Chris Kedzie. Do androids know they're only dreaming of electric sheep? *arXiv preprint arXiv:2312.17249*, 2023. 84

Tuhin Chakrabarty, Christopher Hidey, and Kathleen McKeown. Imho fine-tuning improves claim detection. In *Proceedings of NAACL-HLT*, pages 558–563, 2019. 44

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016. 71

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)*, pages 675–686, 2018. 10, 53, 55, 63, 64, 65, 112

Cheng-Han Chiang and Hung-yi Lee. Pre-training a language model without human language. *arXiv preprint arXiv:2012.11995*, 2020. 34, 35, 42, 44

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/. 60

Eric Chu and Peter Liu. Meansum: a neural model for unsupervised multi-document abstractive summarization. In *International Conference on Machine Learning*, pages 1223–1232. PMLR, 2019. 100

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv:2309.03883*, 2023. 79

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022. 57, 60, 72

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2019. 41, 42, 45, 46, 50, 102

Arman Cohan and Nazli Goharian. Revisiting summarization evaluation for scientific articles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 806–813, 2016. 60, 106

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short*

Papers), pages 615–621, 2018. 100

Apache Commons. Refinedsoundex. URL https://commons.apache.org/proper/commons-codec/apidocs/org/apache/commons/codec/language/RefinedSoundex.html. 16

Paresh Dave. Stack Overflow will charge AI giants for training data. 4 2023. URL https://www.wired.com/story/stack-overflow-will-charge-ai-giants-for-training-data/. 83

Thomas Davidson, Debasmita Bhattacharya, and Ingmar Weber. Racial bias in hate speech and abusive language detection datasets. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 25–35, 2019. 34

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023. 71, 117

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186, 2019. 3, 5, 41, 42, 81, 101

Alaaeldin El-Nouby, Gautier Izacard, Hugo Touvron, Ivan Laptev, Hervé Jegou, and Edouard Grave. Are large-scale datasets necessary for self-supervised pre-training? *arXiv preprint arXiv:2112.10740*, 2021. 43, 44

Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479, 2004. 1, 10

Ori Ernst, Ori Shapira, Ramakanth Pasunuru, Michael Lepioshkin, Jacob Goldberger, Mohit Bansal, and Ido Dagan. Summary-source proposition-level alignment: Task, datasets and supervised baseline. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 310–322, 2021. 53, 63, 64, 65, 112

Alexander Richard Fabbri, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. Qafacteval: Improved qa-based factual consistency evaluation for summarization. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2587–2601, 2022. 53, 64, 69, 78

Katja Filippova. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 23rd international conference on computational linguistics*, pages 322–330. Association for Computational Linguistics, 2010. 10

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023. 82

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, 2020. 3, 33, 52

Sebastian Gehrmann, Hendrik Strobelt, Robert Krüger, Hanspeter Pfister, and Alexander M

Rush. Visual interaction with deep learning models through collaborative semantic inference. *IEEE transactions on visualization and computer graphics*, 26(1):884–894, 2019. 54

Robert Geirhos, Kristof Meding, and Felix A Wichmann. Beyond accuracy: quantifying trial-by-trial behaviour of cnns and humans by measuring error consistency. *Advances in Neural Information Processing Systems*, 33:13890–13902, 2020. 49

Shima Gerani, Giuseppe Carenini, and Raymond T Ng. Modeling content and structure for abstractive review summarization. *Computer Speech & Language*, 53:302–331, 2019. 1, 100

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79, 2019. 1, 8, 22, 25, 36, 54, 100

Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009, 2009. 45

Raphael Gontijo-Lopes, Yann Dauphin, and Ekin D Cubuk. No one representation to rule them all: Overlapping features of training methods. In *International Conference on Learning Representations (ICLR)*, 2022. 49, 50

Tanya Goyal and Greg Durrett. Annotating and modeling fine-grained factuality in summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1449–1462, 2021. 53, 64, 69, 78

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. News summarization and evaluation in the era of gpt-3. *arXiv preprint arXiv:2209.12356*, 2022. 1, 2, 54, 64, 106

Matt Grenander, Yue Dong, Jackie Chi Kit Cheung, and Annie Louis. Countering the effects of lead bias in news summarization via multi-stage training and auxiliary losses. In *EMNLP-IJCNLP*, pages 6021–6026, 2019. 2, 36

Max Grusky, Mor Naaman, and Yoav Artzi. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. *arXiv preprint arXiv:1804.11283*, 2018. 8

Michael Grynbaum and Ryan Mac. The times sues openai and microsoft over a.i. use of copyrighted work. *The New York Times*, 2023. URL `https://www.nytimes.com/2023/12/27/business/media/new-york-times-open-ai-microsoft-lawsuit.html`. 3, 83

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017. xviii, 50

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, 2020. 41, 42, 44, 46, 81, 102

Xiaochuang Han and Jacob Eisenstein. Unsupervised domain adaptation of contextualized embeddings for sequence labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Lan-*

guage Processing (EMNLP-IJCNLP), pages 4238–4248, 2019. 44

Hiroaki Hayashi, Prashant Budania, Peng Wang, Chris Ackerson, Raj Neervannan, and Graham Neubig. Wikiasp: A dataset for multi-domain aspect-based summarization. *Transactions of the Association for Computational Linguistics*, 9:211–225, 2021. 65

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022. 43

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701, 2015. 1, 2

Sepp Hochreiter and Jürgen Schmidhuber. Lstm can solve hard long time lag problems. *Advances in neural information processing systems*, 9, 1996. 1

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2019. 73

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021. 71

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. Codesearchnet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436*, 2019. 24

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2073–2083, 2016. 100

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 2022. 64

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023. 69, 72, 73

Hongyan Jing, Daniel Lopresti, and Chilin Shih. Summarization of noisy documents: a pilot study. In *Proceedings of the HLT-NAACL 03 Text Summarization Workshop*, pages 25–32, 2003. 22, 23

Anirudh Joshi, Namit Katariya, Xavier Amatriain, and Anitha Kannan. Dr. summarize: Global summarization of medical dialogue by exploiting local structures. *arXiv preprint arXiv:2009.08666*, 2020. 10

Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 31–39, 2014. 64

Kaggle.com. Toxic comment classification challenge: Identify and classify toxic online comments. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge, 2018. 45

Ryo Kamoi, Tanya Goyal, Juan Diego Rodriguez, and Greg Durrett. WiCE: Real-world entailment for claims in Wikipedia. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7561–7583, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.470. URL https://aclanthology.org/2023.emnlp-main.470. 78

Masahiro Kaneko, Aizhan Imankulova, Danushka Bollegala, and Naoaki Okazaki. Gender bias in masked language models for multiple languages. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2740–2750, 2022. 3

Neel Kanwal and Giuseppe Rizzo. Attention-based clinical note summarization. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, pages 813–820, 2022. 67

Vladimir Karpukhin, Omer Levy, Jacob Eisenstein, and Marjan Ghazvininejad. Training on synthetic noise improves robustness to natural noise in machine translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 42–47, 2019. 3, 23

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, 2020. 82

Atif Khan, Muhammad Adnan Gul, Mahdi Zareei, RR Biswal, Asim Zeb, Muhammad Naeem, Yousaf Saeed, and Naomie Salim. Movie review summarization using supervised learning and graph-based ranking algorithm. *Computational intelligence and neuroscience*, 2020, 2020. 1

Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48, 2020. 82

Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. Abstractive summarization of reddit posts with multi-level memory networks, 2018. 25

Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. Abstractive summarization of Reddit posts with multi-level memory networks. 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)*, pages 2519–2531, Minneapolis, Minnesota, June 2019a. Association for Computational Linguistics. doi: 10.18653/v1/N19-1260. URL https://www.aclweb.org/anthology/N19-1260. 1, 22

Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. Abstractive summarization of reddit posts with multi-level memory networks. In *Proceedings of NAACL-HLT*, pages 2519–2531, 2019b. 64, 73

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. Text generation from knowledge graphs with graph transformers. 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), pages 2284–2293, 2019. 100

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*, pages 48–56, 2019. 1

Kundan Krishna and Balaji Vasan Srinivasan. Generating topic-oriented summaries using neural attention. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1697–1705, 2018. 65

Kundan Krishna, Jeffrey P Bigham, and Zachary C Lipton. Does pretraining for summarization require knowledge transfer? In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3178–3189, 2021a. 5, 42, 44

Kundan Krishna, Sopan Khosla, Jeffrey P Bigham, and Zachary C Lipton. Generating soap notes from doctor-patient conversations using modular summarization techniques. 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)*, pages 4958–4972, 2021b. 1, 4, 105

Kundan Krishna, Amy Pavel, Benjamin Schloss, Jeffrey P Bigham, and Zachary C Lipton. Extracting structured data from physician-patient conversations by predicting noteworthy utterances. In *Explainable AI in Healthcare and Medicine*, pages 155–169. Springer, 2021c. 8

Kundan Krishna, Saurabh Garg, Jeffrey P Bigham, and Zachary C Lipton. Downstream datasets make surprisingly good pretraining corpora. *arXiv preprint arXiv:2209.14389*, 2022. 5

Kundan Krishna, Prakhar Gupta, Sanjana Ramprasad, Byron Wallace, Jeffrey Bigham, and Zachary Lipton. USB: A unified summarization benchmark across tasks and domains. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8826–8845, 2023a. 68, 70, 78

Kundan Krishna, Prakhar Gupta, Sanjana Ramprasad, Byron C Wallace, Jeffrey P Bigham, and Zachary C Lipton. Usb: A unified summarization benchmark across tasks and domains. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8826–8845, 2023b. 5

Kundan Krishna, Yao Zhao, Jie Ren, Balaji Lakshminarayanan, Jiaming Luo, Mohammad Saleh, and Peter J Liu. Improving the robustness of summarization models by detecting and removing input noise. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1324–1336, 2023c. 4, 64

Wojciech Kryściński, Bryan McCann, Caiming Xiong, and Richard Socher. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, 2020. 55, 63, 64, 78, 112

Gogi Kumar and Adam Mezoff. Physician burnout at a children's hospital: Incidence, interventions, and impact. *Pediatric Quality & Safety*, 5(5), 2020. 8

Philippe Laban, Tobias Schnabel, Paul N Bennett, and Marti A Hearst. Summac: Re-visiting nli-based models for inconsistency detection in summarization. *Transactions of the Association for Computational Linguistics*, 10:163–177, 2022. 78

Philippe Laban, Wojciech Kryściński, Divyansh Agarwal, Alexander Richard Fabbri, Caiming Xiong, Shafiq Joty, and Chien-Sheng Wu. Summedits: Measuring llm ability at factual reasoning through the lens of summarization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9662–9676, 2023. 69, 77, 78

Frederic Lardinois. Adobe starts paying bonuses to stock contributors whose content is being used to train firefly. *TechCrunch*, 2023. URL https://techcrunch.com/2023/09/13/adobe-starts-paying-bonuses-to-stock-contributors-whose-content-is-being- 83

Logan Lebanoff, Kaiqiang Song, Franck Dernoncourt, Doo Soon Kim, Seokhwan Kim, Walter Chang, and Fei Liu. Scoring sentence singletons and pairs for abstractive summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2175–2189, 2019. 65

Rémi Lebret, David Grangier, and Michael Auli. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213, 2016. 100

Yoav Levine, Noam Wies, Daniel Jannai, Dan Navon, Yedid Hoshen, and Amnon Shashua. The inductive bias of in-context learning: Rethinking pretraining example design. In *International Conference on Learning Representations*, 2021. 101

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703. 21, 23

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 2020b. 1, 33, 35, 64

Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. Halueval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6449–6464, 2023a. 67

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *arXiv preprint arXiv:2306.03341*, 2023b. 79

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023c. 83

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004a. 59, 60, 106

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004b. Association for Computational Linguistics. URL `https://aclanthology.org/W04-1013`. 25

Chin-Yew Lin and Eduard Hovy. Manual and automatic evaluation of summaries. In *Proceedings of the ACL-02 Workshop on Automatic Summarization-Volume 4*, pages 45–51, 2002. 93

Yongjie Lina, Yi Chern Tana, and Robert Frankb. Open sesame: Getting inside bert's linguistic knowledge. *ACL 2019*, page 241, 2019. 34, 42, 44

Chunyi Liu, Peng Wang, Jiang Xu, Zang Li, and Jieping Ye. Automatic dialogue summary generation for customer service. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1957–1965, 2019a. 10

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*, 2018. 100

Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In *Proceedings of EMNLP-IJCNLP 2019*, pages 3721–3731, 2019. 1, 64, 100

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. 2019b. 42, 45, 60, 81

Yixin Liu, Budhaditya Deb, Milagro Teruel, Aaron Halfaker, Dragomir Radev, and Ahmed H Awadallah. On improving summarization factual consistency from natural language feedback. *arXiv preprint arXiv:2212.09968*, 2022. 65

Yixin Liu, Budhaditya Deb, Milagro Teruel, Aaron Halfaker, Dragomir Radev, and Ahmed Hassan Awadallah. On improving summarization factual consistency from natural language feedback. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15144–15161, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.844. URL `https://aclanthology.org/2023.acl-long.844`. 78

Zhengyuan Liu, Jia Hui Hazel Lim, Nur Farah Ain Suhaimi, Shao Chuen Tong, Sharon Ong, Angela Ng, Sheldon Lee Shao Guang, Michael Ross Macdonald, Savitha Ramasamy, Pavitra Krishnaswamy, et al. Fast prototyping a dialogue comprehension system for nurse-patient conversations on symptom monitoring. In *NAACL-HLT (2)*, 2019c. 8

Zhengyuan Liu, Angela Ng, Sheldon Lee, Ai Ti Aw, and Nancy F Chen. Topic-aware pointer-generator networks for summarizing spoken conversations. *arXiv preprint arXiv:1910.01335*, 2019d. 10

Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. Zero-shot entity linking by reading entity descriptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3449–3460, 2019. 44

Shayne Longpre, Gregory Yauney, Emily Reif, Katherine Lee, Adam Roberts, Barret Zoph, Denny Zhou, Jason Wei, Kevin Robinson, David Mimno, et al. A pretrainer's guide to training data: Measuring the effects of data age, domain coverage, quality, & toxicity. *arXiv preprint arXiv:2305.13169*, 2023. 83

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018. 102

Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*, 2021. 44

Hartmut Maennel, Ibrahim Alabdulmohsin, Ilya Tolstikhin, Robert JN Baldock, Olivier Bousquet, Sylvain Gelly, and Daniel Keysers. What do neural networks learn when trained with random labels? 2020. 35

Pratyush Maini, Skyler Seto, He Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. Rephrasing the web: A recipe for compute and data-efficient language modeling. *arXiv preprint arXiv:2401.16380*, 2024. 83

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022. 117

Christopher D Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054, 2020. 34, 44

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, 2020. 2, 53

Ailsa Meechan-Maddon. The effect of noise in the training of convolutional neural networks for text summarisation, 2019. 23

Cade Metz. Lawsuit takes aim at the way a.i. is built. *The New York Times*, 2022. URL https://www.nytimes.com/2022/11/23/technology/copilot-microsoft-ai-lawsuit.html. 3

Cade Metz. Chatbots may'hallucinate'more often than many realize. *International New York Times*, pages NA–NA, 2023. 2

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.741. URL https://aclanthology.

org/2023.emnlp-main.741. 67, 78

Abhika Mishra, Akari Asai, Vidhisha Balachandran, Yizhong Wang, Graham Neubig, Yulia Tsvetkov, and Hannaneh Hajishirzi. Fine-grained hallucination detection and editing for language models. *arXiv preprint arXiv:2401.06855*, 2024. xix, 74, 75, 78

Amit Moryossef, Yoav Goldberg, and Ido Dagan. Step-by-step: Separating planning from realization in neural data-to-text 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)*, pages 2267–2277, 2019. 100

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, 2016. 1, 8, 10, 54

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-first AAAI conference on artificial intelligence*, 2017. 64

Feng Nan, Ramesh Nallapati, Zhiguo Wang, Cicero dos Santos, Henghui Zhu, Dejiao Zhang, Kathleen Mckeown, and Bing Xiang. Entity-level factual consistency of abstractive text summarization. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2727–2733, 2021. 64

Shashi Narayan, Shay Cohen, and Maria Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807. Association for Computational Linguistics, 2018a. 22, 25, 73

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. *ArXiv*, abs/1808.08745, 2018b. 1, 2, 38, 54, 64

Ani Nenkova. Summarization evaluation for text and speech: issues and approaches. In *Ninth International Conference on Spoken Language Processing*, 2006. 106

Ani Nenkova, Kathleen McKeown, et al. Automatic summarization. *Foundations and Trends® in Information Retrieval*, 5(2–3):103–233, 2011. 10

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? In *Advanes in Neural Information Processing Systems (NeurIPS)*, 2020. 35

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, et al. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9844–9855, 2022. 82

Xing Niu, Prashant Mathur, Georgiana Dinu, and Yaser Al-Onaizan. Evaluating robustness to input perturbations for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8538–8544, 2020. 23

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. The e2e dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, 2017. 100

Artidoro Pagnoni, Vidhisha Balachandran, and Yulia Tsvetkov. Understanding factuality in abstractive summarization with frank: A benchmark for factuality metrics. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4812–4829, 2021. 2

Isabel Papadimitriou and Dan Jurafsky. Learning music helps you read: Using transfer to study linguistic structure in language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6829–6839, 2020. 34, 35, 44

Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*, 2018. 100

Denis Peskov, Joe Barrow, Pedro Rodriguez, Graham Neubig, and Jordan Boyd-Graber. Mitigating noisy inputs for question answering. *arXiv preprint arXiv:1908.02914*, 2019. 23

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL `https://aclanthology.org/N18-1202`. 42

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, 2019. 42, 44

Vivek Podder, Valerie Lew, and Sassan Ghassemzadeh. Soap notes. In *StatPearls [Internet]*. StatPearls Publishing, 2021. 4

Ratish Puduppully, Li Dong, and Mirella Lapata. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6908–6915, 2019. 100

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 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*, pages 2401–2410, 2020. 33

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023. 79

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024. 83

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020a. 1, 2, 3, 13, 21, 23, 24, 33, 35, 39, 42, 60, 64

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020b. URL http://jmlr.org/papers/v21/20-074.html. 26

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016. 51

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506, 2020. 117

Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. Can wikipedia help offline reinforcement learning? *arXiv preprint arXiv:2201.12122*, 2022. 44

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024. 82

Jie Ren, Jiaming Luo, Yao Zhao, Kundan Krishna, Mohammad Saleh, Balaji Lakshminarayanan, and Peter J. Liu. Out-of-distribution detection and selective generation for conditional language models, 2022. URL https://arxiv.org/abs/2209.15558. 22, 26, 27

Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, 2015. 1, 2

Mobashir Sadat, Zhengyu Zhou, Lukas Lange, Jun Araki, Arsalan Gundroo, Bingqing Wang, Rakesh Menon, Md Parvez, and Zhe Feng. Delucionqa: Detecting hallucinations in domain-specific question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 822–835, 2023. 67

Erik Tjong Kim Sang and Sabine Buchholz. Introduction to the conll-2000 shared task chunking. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*, 2000. 60

Benjamin Schloss and Sandeep Konam. Towards an automated soap note: Classifying utterances from medical conversations. In *Machine Learning for Healthcare Conference*, pages 610–631. PMLR, 2020. 8

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, Jacopo Staiano, Alex Wang, and Patrick Gallinari. Questeval: Summarization asks for fact-based evaluation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6594–6604, 2021. 54, 64

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)*, pages 1073–1083, 2017. 1, 2, 10, 13, 22, 25, 38, 40, 53, 64, 78, 100

Umar Shakir. Reddit's new developer API terms enable monetization avenue from AI makers. 4 2023. URL https://www.theverge.com/2023/4/18/23688463/reddit-developer-api-terms-change-monetization-ai. 83

Eva Sharma, Chen Li, and Lu Wang. Bigpatent: A large-scale dataset for abstractive and coherent summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2204–2213, 2019. 100

Damien Sileo, Tim Van De Cruys, Camille Pradel, and Philippe Muller. Mining discourse markers for unsupervised sentence representation learning. 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)*, pages 3477–3486, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N19-1351. 45

Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. *arXiv preprint arXiv:2104.06644*, 2021. 34, 35, 44

Christine Sinsky, Lacey Colligan, Ling Li, Mirela Prgomet, Sam Reynolds, Lindsey Goeders, Johanna Westbrook, Michael Tutty, and George Blike. Allocation of physician time in ambulatory practice: a time and motion study in 4 specialties. *Annals of internal medicine*, 2016. 7

Aviv Slobodkin, Paul Roit, Eran Hirsch, Ori Ernst, and Ido Dagan. Controlled text reduction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5699–5715, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.385. 65

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. A graph-to-sequence model for amr-to-text generation. *arXiv preprint arXiv:1805.02473*, 2018. 100

Jonathan Stempel. Nvidia is sued by authors over ai use of copyrighted works. *Reuters*, 2024. URL https://www.reuters.com/technology/nvidia-is-sued-by-authors-over-ai-use-copyrighted-works-2024-03-10/. 3, 83

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020. 1

Sandeep Subramanian, Raymond Li, Jonathan Pilault, and Christopher Pal. On extractive and abstractive neural document summarization with transformer language models. *arXiv preprint arXiv:1909.03186*, 2019. 10

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014. 10

Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Steven Zheng, et al. Ul2: Unifying language learning paradigms. In *The Eleventh International Conference on Learning Representations*, 2022. 3, 72, 73

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 6, 69

Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. In *ACL*, pages 4593–4601, 2019. 34, 42, 43, 44

Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher Manning, and Chelsea Finn. Fine-tuning language models for factuality. In *ICLR*, 2024. 79, 83

Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari Morcos. D4: Improving llm pretraining via document de-duplication and diversification. *Advances in Neural Information Processing Systems*, 36, 2024. 83

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 3, 6, 60, 69, 72, 73

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. In *ACL*, pages 76–85, 2016. 87

Vaibhav Vaibhav, Sumeet Singh, Craig Stewart, and Graham Neubig. Improving robustness of machine translation with synthetic noise. 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)*, pages 1916–1920, 2019. 3, 23

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1

James Vincent. Getty images is suing the creators of ai art tool stable diffusion for scraping its content. *The Verge*, 2023. URL https://www.theverge.com/2023/1/17/23558516/ai-art-copyright-stable-diffusion-getty-images-lawsuit. 3, 83

Pavlos Vougiouklis, Hady Elsahar, Lucie-Aimée Kaffee, Christophe Gravier, Frédérique Laforest, Jonathon Hare, and Elena Simperl. Neural wikipedian: Generating textual summaries from knowledge base triples. *Journal of Web Semantics*, 52:1–15, 2018. 100

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural*

Networks for NLP, pages 353–355, 2018. 45

Lu Wang and Claire Cardie. Domain-independent abstract generation for focused meeting summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1395–1405, 2013. 10, 100

Lu Wang and Wang Ling. Neural network-based abstract generation for opinions and arguments. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 47–57, 2016. 38, 54, 64

Alex Warstadt, Yian Zhang, Xiaocheng Li, Haokun Liu, and Samuel Bowman. Learning which features matter: Roberta acquires a preference for linguistic generalizations (eventually). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 217–235, 2020. 50, 102

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*, 23, 2013. 51

Jason Weston and Sainbayar Sukhbaatar. System 2 attention (is something you might need too). *arXiv preprint arXiv:2311.11829*, 2023. 83

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)*, pages 1112–1122, 2018. 45

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, 2017. 38

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019. 117

Kam-Fai Wong, Mingli Wu, and Wenjie Li. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of the 22nd international conference on computational linguistics (Coling 2008)*, pages 985–992, 2008. 1, 10, 64

Yuhuai Wu, Felix Li, and Percy Liang. Insights into pre-training via simpler synthetic tasks. *arXiv preprint arXiv:2206.10139*, 2022. 42, 43, 44

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2023. 82

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, 32:5753–5763, 2019. 33, 42

Xingcheng Yao, Yanan Zheng, Xiaocong Yang, and Zhilin Yang. Nlp from scratch without large-scale pretraining: A simple and efficient framework. In *International Conference on Machine*

Learning, pages 25438–25451. PMLR, 2022. 44, 101

Michihiro Yasunaga, Jungo Kasai, Rui Zhang, Alexander R Fabbri, Irene Li, Dan Friedman, and Dragomir R Radev. Scisummnet: A large annotated corpus and content-impact models for scientific paper summarization with citation networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7386–7393, 2019. 100

Wen-wai Yim, Yujuan Fu, Asma Ben Abacha, Neal Snider, Thomas Lin, and Meliha Yetisgen. Aci-bench: a novel ambient clinical intelligence dataset for benchmarking automatic visit note generation. *Scientific Data*, 10(1):586, 2023. 73

Lin Yuan and Zhou Yu. Abstractive dialog summarization with semantic scaffolds. *arXiv preprint arXiv:1910.00825*, 2019. 10

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020. 82

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *EMNLP*, 2018. 51

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, 2019. 51

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR, 2020. 1, 2, 21, 22, 23, 25, 33, 35, 64, 81, 100

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015. 45

Yong Zhang, Joo Er Meng, and Mahardhika Pratama. Extractive document summarization based on convolutional neural networks. In *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 918–922. IEEE, 2016. 64

Yuan Zhang, Jason Baldridge, and Luheng He. Paws: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1298–1308, 2019. 45, 103

Chenguang Zhu, Ruochen Xu, Michael Zeng, and Xuedong Huang. End-to-end abstractive summarization for meetings. *arXiv preprint arXiv:2004.02016*, 2020. 10

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015. 42

Yanyan Zou, Xingxing Zhang, Wei Lu, Furu Wei, and Ming Zhou. Pre-training for abstractive document summarization by reinstating source text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3646–3660, 2020. 34, 35, 36, 95