Quantifying the Impact of Disfluency on Spoken Content Summarization

Maria Teleki, Xiangjue Dong, James Caverlee
Texas A&M University
College Station, Texas, USA
{mariateleki, xj.dong, caverlee}@tamu.edu

Abstract

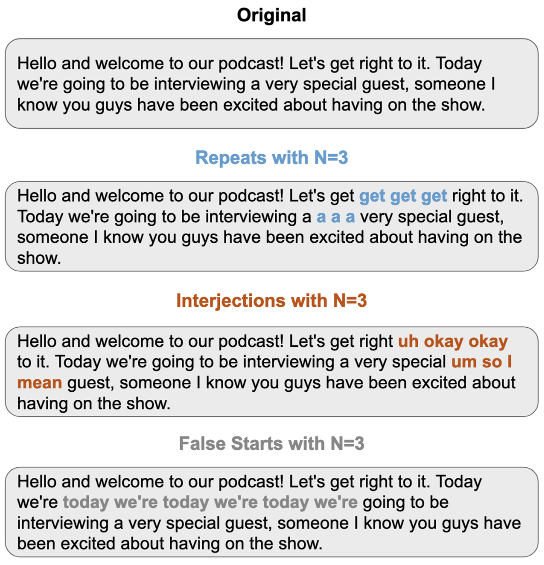
Spoken content is abundant – including podcasts, meeting transcripts, and TikTok-like short videos. And yet, many important tasks like summarization are often designed for written content rather than the looser, noiser, and more disfluent style of spoken content. Hence, we aim in this paper to quantify the impact of disfluency on spoken content summarization. Do disfluencies negatively impact the quality of summaries generated by existing approaches? And if so, to what degree? Coupled with these goals, we also investigate two methods towards improving summarization in the presence of such disfluencies. We find that summarization quality does degrade with an increase in these disfluencies and that a combination of multiple disfluency types leads to even greater degradation. Further, our experimental results show that naively removing disfluencies and augmenting with special tags can worsen the summarization when used for testing, but that removing disfluencies for fine-tuning yields the best results. We make the code available at `https://github.com/mariateleki/Quantifying-Impact-Disfluency`.

Keywords: disfluency, summarization, spoken content

1. Introduction

Spoken content is a popular and natural way to share information: consider podcasts, meeting transcripts, and voice texts, as well as the spoken portion of videos shared on platforms like YouTube. One of the key characteristics of spoken content – especially in comparison to written text – is the presence of *disfluencies*. Indeed, according to the American Speech-Language-Hearing Association, *all speakers are disfluent at times* (American Speech-Language-Hearing Association, 2023). In comparison to fluency – which refers to "continuity, smoothness, rate, and effort in speech production" (American Speech-Language-Hearing Association, 2023) – disfluency arises through word and phrase repetition, filler speech (e.g., "uh"), and conversational speech (e.g., compare reading a book out loud versus engaging in a normal conversation with its conversational and disorganized style).

At the same time, many important NLP tasks like summarization are often designed for written content rather than the looser, noiser, and more disfluent style of spoken content (Liu and Lapata, 2019; Lewis et al., 2020; Nenkova and McKeown, 2012; Nallapati et al., 2016). As a result, the quality of summaries generated from spoken content can be degraded. For example, a false start like "My friend, uh, my boss called …" could lead the summarizer to attribute an action (*calling*) to the wrong subject (*friend* instead of *boss*). Furthermore, it is recognized that systems which do not account for disfluency are particularly unfair to people who exhibit more disfluent speech (Lloreda, 2020). Together, these challenges mean that special care should be taken to understand and mitigate the impact of disfluencies on summarization.


Original

Hello and welcome to our podcast! Let's get right to it. Today we're going to be interviewing a very special guest, someone I know you guys have been excited about having on the show.

Repeats with N=3

Hello and welcome to our podcast! Let's get **get get get** right to it. Today we're going to be interviewing a **a a a** very special guest, someone I know you guys have been excited about having on the show.

Interjections with N=3

Hello and welcome to our podcast! Let's get right **uh okay okay** to it. Today we're going to be interviewing a very special **um so I mean** guest, someone I know you guys have been excited about having on the show.

False Starts with N=3

Hello and welcome to our podcast! Let's get right to it. Today we're **today we're today we're today we're** going to be interviewing a very special guest, someone I know you guys have been excited about having on the show.

Figure 1: The three basic disfluency transformations (repeats, interjections, and false starts) applied to an example transcript with $N = 3$.

Hence, we aim in this paper to quantify the impact of disfluency on spoken content summarization. Do disfluencies negatively impact the quality of summaries generated by existing approaches? And if so, to what degree? Are these impacts common across different summarization models? And what impact do different kinds of disfluency have on summarization? Coupled with these goals of

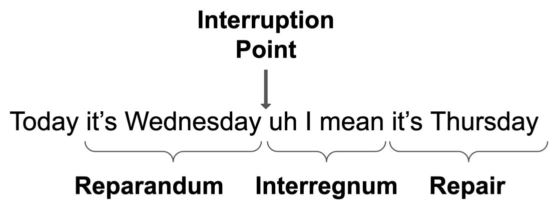


Figure 2: Disfluency structure as defined by Shriberg (1994). Disfluencies occur in the reparandum and the interregnum, hence these are deleted in order to form a fluent sentence from a disfluent one.

better understanding the impact of different kinds of disfluent speech, we also investigate two methods towards improving summarization in the presence of such disfluencies – (1) repairing the podcast transcripts through deletion of detected disfluencies, and (2) adding disfluency tags to detected disfluent words in the podcast transcripts.

Concretely, we focus in this paper on three types of speech disfluencies that are common (see Figure 1) and their impact on summarization: repeats, interjections, and false starts. A repeat occurs when a single word is repeated, an interjection occurs when a specific interjection word or two words (*"uh", "like", "I mean", etc.*) occurs, and a false start occurs when the first two words of a sentence are repeated. We also analyze the impact of the combinations of these disfluency types.

We ground our analysis in the largest spoken content summarization testbed – the Spotify Podcast Dataset – and analyze the impact of these disfluencies on the summarization of podcasts, wherein a summary is constructed that captures the key characteristics of the podcast transcripts (Karlbom and Clifton, 2020; Vartakavi et al., 2021; Rezapour et al., 2022). As part of our assessment, we consider the top performers at the TREC 2020 Podcasts Track (Jones et al., 2020; Rezapour et al., 2022; Manakul and Gales, 2020; Clifton et al., 2020) as well as popular approaches like Llama 2-Chat, Pegasus, T5, and BART.

In summary:

- By synthetically varying the presence of these disfluencies, we explore the spectrum of their impact on summarization quality over five state-of-the-art abstractive summarization approaches. We find that all summarizers are negatively impacted, even at low levels of disfluency. As disfluencies increase, we find that Llama 2-Chat, a chat-based model, is the most resilient off-the-shelf model to disfluency – however, it also has the worst summarization performance on the original text. BART, T5, and Pegasus suffer from worse drops in performance in the presence of disfluencies than Llama 2-Chat, but have better performance on the original text.
- Second, we explore two strategies to incorporate disfluency into summarization. We find that neither of these methods are more effective for inference than simply using the original podcast transcripts. However, when fine-tuning, utilizing a disfluency detection model (Jamshid Lou and Johnson, 2020b) to delete disfluent words, and essentially repair the podcast transcripts, at training time leads to an increase in summarization performance.

2. Related Work

Disfluency Structure. Toward modeling speech, there have been many efforts over the years to understand and address disfluent speech – usually, to repair the disfluencies in an effort to make the speech compatible with existing systems (Jamshid Lou and Johnson, 2020b; Chaudhury et al., 2024; Clark et al., 2020; Rocholl et al., 2021; Jamshid Lou and Johnson, 2020a; Dong et al., 2019; Wang et al., 2017). Disfluencies "are cases in which a contiguous stretch of linguistic material must be deleted to arrive at the sequence the speaker 'intended'," and occur within a broader disfluency structure (Shriberg, 1994). As shown in Figure 2, Shriberg (1994) defines the reparandum, interruption point, interregnum, and repair. The reparandum and the interregnum are comprised of the speech which must be deleted in order to form a fluent sentence. The repair contains the speech which "corresponds to" or is a "rough copy" of the speech in the reparandum (Shriberg, 1994; Charniak and Johnson, 2001). The repair is not deleted, and remains a part of the intended, fluent sentence. In this paper, we explore repeats, interjections, and false starts; in terms of the Shriberg (1994) disfluency structure: repeats occur within the reparandum, interjections occur within the interregnum, and false starts occur within the reparandum. In a general study of disfluency, it was found that "speech disfluencies occur at higher perplexities," and "disfluencies are more likely to occur before less predictable words" (Sen, 2020). It is also known that disfluencies can also be useful for human memory, as they can help "bring attentional focus to immediately upcoming material" (Diachek and Brown-Schmidt, 2023).

Disfluency Disorders. Fluency disorders (i.e. stuttering, cluttering) can be a cause of increased disfluency generation in speech (American Speech-Language-Hearing Association, 2023). In industry, Google (2023)'s Project Euphonia recruits people with disfluent, "non-standard" speech (such as from speakers with neurological diseases, including: Amyotrophic Lateral Sclerosis, Parkinson's Disease, or Cerebral Palsy) to record their speech in an effort to improve their automatic speech recognition (ASR) systems (Venugopalan et al., 2023). Note that while "[a]ll speakers are disfluent at times," the occasional disfluency does not constitute a fluency disorder (American Speech-Language-Hearing Association, 2023).

Summarization. Automatic summarization of text is an established natural language processing task (Liu and Lapata, 2019; Lewis et al., 2020; Nenkova and McKeown, 2012; Nallapati et al., 2016). In this work, we utilize the first minute of transcript text for the summaries as a simple baseline (Jones et al., 2020). We also use a specialized podcast summarization model from the TREC 2020 podcast track (Manakul and Gales, 2020). Additionally, we select four popular, more generic automatic summarization models: BART (Lewis et al., 2020), T5 (Raffel et al., 2020), Pegasus (Zhang et al., 2020), and Llama 2-Chat (Touvron et al., 2023).

3. Research Questions

Our focus in this paper is the impact on the downstream task of summarization of three types of disfluencies in spoken content: repeats, interjections, and false starts, as shown in Figure 1. To evaluate the influence of disfluencies in spoken content, we aim to answer the following two research questions in our experiments:

- ***RQ1: How Do Disfluencies Impact Summarization Quality?*** We synthetically inject disfluency events (repeats, interjections, false starts, and their combinations) at a range of severity levels and measure their impact on summarization quality.
- ***RQ2: Can Summarization Quality be Improved By Directly Modeling Disfluency?*** We explore the use of a state-of-the-art disfluency detection model (Jamshid Lou and Johnson, 2020b) to improve the summarization quality by either (1) removing the disfluencies, or (2) tagging the disfluencies.

4. Preliminaries

Podcast Dataset. For all of our experiments, we adopt the Spotify Podcast Dataset (Clifton et al., 2020), a collection of around 100,000 podcasts from Spotify.[1] This dataset was originally used for two tasks from the TREC 2020 Podcasts Track (Jones et al., 2020): the summarization task and the segment retrieval task. Specifically, we use the test set for the summarization task, which consists of 1,027 podcasts. For each, we have the podcast transcript and metadata. The metadata includes the Show ID, Episode ID, creator-provided show description, and creator-provided episode description. We keep podcasts which have text occurring in their transcript in the first 60 seconds, which leaves us with 1,020 podcasts.

Disfluency Injection. While the podcasts have naturally occurring disfluencies, we inject additional disfluencies towards stress-testing the capabilities of summarization methods. Similar to other recent works (Wang et al. (2020); Passali et al. (2022)), we use a rule-based approach to generate disfluencies. We inject additional instances of repeats, interjections, and false starts. For each disfluency transformation, we control the number of times it occurs with a special parameter N that we can vary experimentally (see Section 3). The use of this method means that we can isolate the impact of various disfluency severity levels on the summarization systems, as we are simply shifting the existing disfluency distribution.

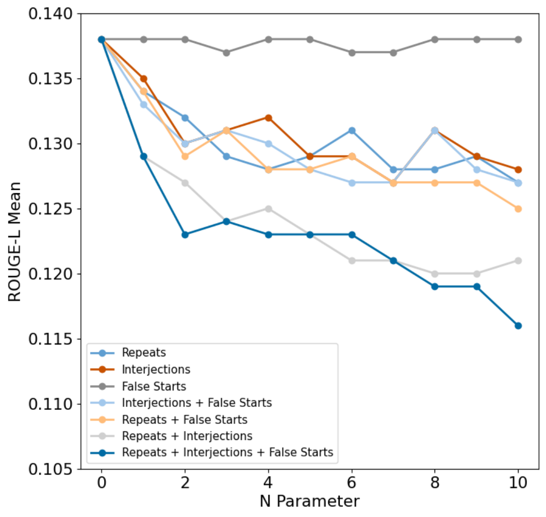
The podcast transcripts are timestamped on the word-level from the ASR system (Clifton et al., 2020) (Google Cloud, 2023). The summarization models have a maximum amount of tokens on which they can operate; in order to ensure that minimal original text is pushed out of the token range as a result of our various transformation operations, we experimented with the lengths of the original transcripts. We found that limiting all of the podcasts to one minute of text (based on their word-level timestamps) led to minimal information loss for feeding into the summarization models.

For the *repeats* and *interjections*, we construct consecutive random subsets of the podcast transcript text by iteratively drawing a sample from $X \sim \mathcal{N}(\mu = 10, \sigma = 1)$ to determine the position at which the repeat or interjection term should be injected into the transcript N times.[2] Additionally, in the case of the repeats transformation, the word is cleaned to maintain correct grammar and punctuation within the context. The cleaning process involves removing punctuation and handling the casing (uppercasing/lowercasing) of the word. The various interjections are uniformly randomly selected from the following list: *"uh", "um", "well", "like", "so", "okay", "I mean", "you know"*.

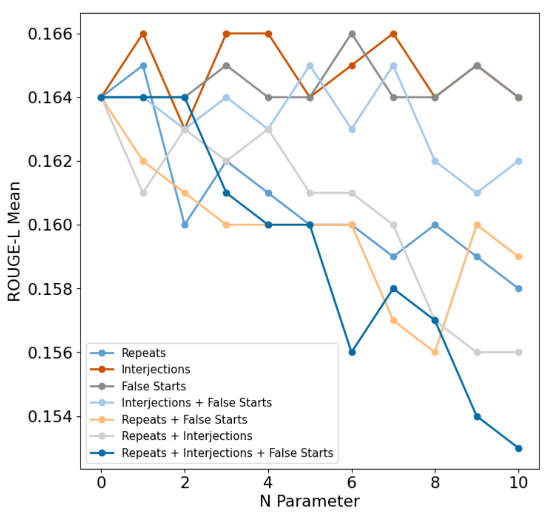
For the *false starts*, we construct a new transcript text by first breaking the original transcript text

[1] https://podcastsdataset.byspotify.com/

[2] Let's consider an example: Sentence 1 is 3 words long; sentence 2 is 16 words long. If the first sample drawn is 10, then sentence 1 will not contain any injected disfluencies. Thus, not all sentences may receive a repeat or interjection amplification.



(a) ROUGE-L over increased N on **BART** model.



(b) ROUGE-L over increased N on **cued_speechUniv2** model.

Figure 3: Disfluency Spectrum Results (increasing from $N = 0$ to $N = 10$): Synthetically increasing the number of disfluencies generally leads to a degradation in summarization quality, as measured by ROUGE-L.

into sentences using NLTK's sentence tokenizer.[3] Then, sentences which are longer than 4 words are non-uniformly sampled with 80/20 probability with replacement, and the selected sentences have a false start interjected N times. The false start consists of the first 2 words of the sentence, cleaned for casing (uppercasing/lowercasing) and punctuation.

5. Experimental Settings

5.1. Summarization Models

We consider six summarization models. The first two are from TREC 2020:

- **1min** is a simple baseline that truncates the podcast transcript to one minute in duration and directly uses it as a summary. As demonstrated in (Jones et al., 2020), this method works well as a baseline.
- **cued_speechUniv2** is an ensemble of three BART models plus a hierarchical model which filters the transcript text; at TREC 2020 this was the top-performing summarization model in the podcast track (Manakul and Gales, 2020).

The next four are widely used and comprehensively evaluated summarization models:

- **BART** is a sequence-to-sequence model with a bidirectional encoder and a left-to-right autoregressive decoder. It is trained by corrupting documents with an arbitrary noising function and then reconstructing the original documents by optimizing a reconstruction loss (Lewis et al., 2020). We use *facebook/bart-large-cnn* from Hugging Face.
- **T5** is a language model trained in a "text-to-text" setting – that is feeding text as input and producing new text as output. It uses unsupervised pre-training to enable the model to learn from a large amount of unlabeled text data available on the internet (Raffel et al., 2020). We use *t5-large* from Hugging Face. We prepend the prompt *"summarize: "* on all inputs.
- **Pegasus** is a transformer-based encoder-decoder model designed for text summarization tasks. It includes a new self-supervised pre-training objective – gap sentence generation – that encourages the model to generate accurate and informative summaries (Zhang et al., 2020). We use *pegasus-large* from Hugging Face.
- **Llama 2-Chat** is a transformer-based large language model (LLM) which is "optimized for dialogue use cases" (Touvron et al., 2023). It is pretrained over a vast amount of cleaned public data (2 trillion tokens) and with Reinforcement Learning with Human Feedback (RLHF). We use *Llama-2-7B-chat* from Meta and Hugging Face. We prepend the prompt *"summarize:"* on all inputs.

For each of the summarization models, we used the *large* version of them on Hugging Face, and

[3]https://www.nltk.org/api/nltk.tokenize.html

Table 1: Disfluency Transformation Results (fixed at $N = 0$ and $N = 2$): difference in means and percent change in ROUGE-L F1 score (R: Repeats; I: Interjections; F: False Starts. Original: original transcript without any disfluency transformations). We designate the largest and smallest drops in percent change (aside from the 1min baseline): **bold** indicates the *largest drop (worst)* for each transformation, and underline indicates the *smallest drop (best)* for each transformation.

Model	N=0	N=2 vs. N=0	R	I	F	I+F	R+F	R+I	R+I+F
1min	0.124	$N_2 - N_0$	-0.013	-0.014	-0.004	-0.017	-0.016	-0.026	-0.029
		Δ	(-10.3%)	(-11.6%)	(-3.2%)	(-14.0%)	(-13.0%)	(-21.0%)	(-23.7%)
BART	0.138	$N_2 - N_0$	-0.006	-0.008	0.000	-0.008	-0.008	-0.011	-0.015
		Δ	(-4.6%)	(-5.5%)	(-0.3%)	(-5.7%)	(-6.1%)	(-7.6%)	(-11.1%)
T5	0.134	$N_2 - N_0$	-0.018	-0.010	-0.003	-0.013	-0.018	-0.025	-0.032
		Δ	**(-13.7%)**	(-7.4%)	(-2.4%)	(-9.9%)	**(-13.7%)**	**(-19.0%)**	**(-23.7%)**
Pegasus	0.131	$N_2 - N_0$	-0.011	-0.014	-0.003	-0.017	-0.014	-0.023	-0.026
		Δ	(-8.8%)	**(-10.4%)**	**(-2.6%)**	**(-12.9%)**	(-10.7%)	(-17.2%)	(-19.9%)
cued_speechUniv2	0.164	$N_2 - N_0$	-0.004	-0.001	-0.001	-0.001	-0.003	-0.002	-0.001
		Δ	(-2.5%)	(-0.8%)	(-0.5%)	(-0.8%)	(-1.9%)	(-1.0%)	(-0.5%)
Llama 2-Chat	0.129	$N_2 - N_0$	-0.001	-0.002	-0.001	-0.002	-0.001	-0.001	-0.002
		Δ	(-0.6%)	(-1.2%)	(-1.0%)	(-1.6%)	(-1.1%)	(-1.1%)	(-1.5%)

controlled their *min_length* and *max_length* generation parameters, setting them to 56 and 144, respectively, so that the summaries are directly comparable with the summaries produced by the winning models from the TREC Podcasts Track (Jones et al., 2020; Manakul and Gales, 2020). Additionally, we set their input *max_length* to be 1,024, allowing truncation and padding.

5.2. Summary Quality Evaluation

Following the TREC evaluation framework, we use the creator-provided episode description as the ground truth summaries[4] and evaluate the performance of the summarization models using ROUGE scores (Lin, 2004), which are commonly used in summarization (Fabbri et al., 2021); variations of ROUGE scores include ROUGE-1, ROUGE-2, and ROUGE-L. A low ROUGE indicates that there is little overlap between the model-generated summary and the human-created summary, while a high ROUGE means there is more overlap, and therefore that the summarization quality is overall better (Lin, 2004). The TREC 2020 Podcasts Track found that the ROUGE-L score was correlated with aggregate manual evaluation scores, as assessed by human NIST evaluators, on a weighted Excellent/Good/Fair/Bad (EGFB) scale (Jones et al., 2020), hence we report only the F1 scores for ROUGE-L here.

6. RQ1: How Do Disfluencies Impact Summarization Quality?

First, how much impact do the three disfluency types have on the performance of automatic summarization models?

6.1. Disfluency Spectrum Results (increasing from $N = 0$ to $N = 10$)

We begin in Figure 3 by showcasing the impact of disfluencies across the extreme spectrum of disfluency. We consider the three disfluency types – repeats (R), interjections (I), and false starts (F) – and their combinations. Here, we vary the N parameter that controls the degree of disfluency from $N = 0$ (the original transcript with no transformations applied) to $N = 10$, to reflect scenarios of extreme disfluency toward stress testing summarization. We consider two representative methods: BART and cued_speechUniv2. Qualitatively similar results hold for the other methods.

In Figure 3a, we observe that for low levels of synthetic disfluency augmentation (e.g., for $N = 1$ and $N = 2$) there is a significant drop in summary quality across nearly all types of disfluency. Increasing the number of disfluencies continues to degrade performance with some variability across specific types of disfluency. We observe that the combination of all of the transformations, R + I + F, has the worst overall performance, as it is mostly below all of the other transformations.

In Figure 3b, we notice that at low levels of disfluency augmentation, there is a drop for many types of disfluency – however, a few types have a small increase in performance, and a few types retain the same performance. As noted below in Table 1, cued_speechUniv2 is most resistant to increases in disfluencies. Yet here we observe that as the level of disfluency increases (to $N = 3, 4, 5, \ldots, 10$), adding additional disfluencies does lead to drops in summary quality for over half of the disfluency types. Further, in Figure 3b, we observe that there are cyclical patterns which occur

[4]We perform two key data cleaning steps on both the summaries and the transcripts: (1) we remove URLs, and (2) we remove non-ascii characters.

with all of the disfluency transformations. We hypothesize that this may be related to the hierarchical filtering model which runs before the ensemble summarization component of this approach (Manakul and Gales, 2020) and deserves further study.

6.2. Disfluency Transformation Results (fixed at $N = 0$ and $N = 2$)

Coupled with this extreme stress-testing, we show in Table 1 the results for all six different summarizers at more moderate levels of disfluency (here we set $N = 2$). We again consider the three disfluency types – repeats (R), interjections (I), and false starts (F) and their combinations versus the original ROUGE score.

We first observe in Table 1 the values for each of the six summarizers at $N = 0$ (no disfluency augmentation). 1min performs the worst, then Llama 2-Chat, Pegasus, T5, BART, and finally cued_speechUniv2 performs the best.

Next, we see that all of the disfluency transformations (R, I, F, and all of their combinations) lead to a degradation in summarization performance, with varying impacts across summarizers. We look at all the summarizers aside from the 1min baseline. Focusing on the individual disfluency types, we see that repeats (R) and interjections (I) lead to larger drops in ROUGE than false starts (F). For example, the variation of the models' drop in performance for repeats (R) ranges from a 0.6% drop to a 13.7% percent drop. For interjections (I), the drops range from 0.8% to 10.4%. For false starts (F), on the other hand, the drops range much smaller: from 0.3% to 2.6%. We attribute this partially to our strict definition of false starts (which follow on work could explore further), but also to the ability of multiple repeats and interjections to confuse the summarization models (e.g., in some cases generating summaries almost entirely comprised of repeated words that are less fluent). Looking at the combinations of disfluencies (I + F, R + F, R + I, R + I + F), we observe that combinations generally lead to even worse summarization quality than only individual disfluencies. Inserting both repeats and interjections (R + I) leads to the biggest drops in performance across all models, when considering pairs of disfluencies. Combining all disfluencies (R + I + F) leads to the most degradation for all five summarizers in terms of summarization performance, ranging from 0.5% to 23.7% drop.

Now turning to specific summarization models, we see that the TREC 2020 top performer, cued_speechUniv2, and Llama 2-Chat are the most resilient to the various disfluency transformations, as they incur the least drop in performance – both in terms of percentage drop and absolute ROUGE-L score. T5 and Pegasus are the least resilient, as they incur the largest drops. BART sustains moderate drops in performance across the disfluency transformations. We hypothesize that this difference in performance has to do with the data that these models were trained on: Llama 2-Chat was trained on chat data, and cued_speechUniv2 was fine-tuned on the podcast transcripts. Hence, they are more resilient to disfluencies, as their pre-training data is in-domain.

Overall, disfluencies degrade the quality of the summaries, even at a moderate level of disfluency (i.e., $N = 2$), and models which are trained on in-domain, spoken or chat data are more resilient to disfluencies, both in terms of percentage dropped and absolute ROUGE-L score.

7. RQ2: Can Summarization Quality be Improved By Directly Modeling Disfluency?

Given the impact of disfluency on summarization quality, we next investigate two straightforward approaches to incorporate evidence of disfluency into the summarizer. Specifically, we investigate the following two approaches as ways to improve BART, T5, and Pegasus' ROUGE-L summarization performance on the podcast transcripts based on the output of the SOTA disfluency annotation model from Jamshid Lou and Johnson (2020b):

- **Repairing: Removal of Disfluent Words.** This approach involves simply removing words marked as disfluent by a disfluency annotation model. Because this approach is a pre-processing step, it can be easily integrated into existing summarization systems.

- **Tagging: Adding Disfluency Tokens.** This approach adds disfluency tags (`<DIS>` and `<\DIS>`) around words marked as disfluent by the disfluency annotation model. We hypothesize that with more information about disfluencies – in the form of tags – the model can better understand which words are more or less important for conveying the key information to the summarizer.

We select BART, T5, and Pegasus due to their time and memory efficiency. Additionally, they sustain the largest decreases in RQ1 (Table 1), so investigating their repair is valuable.

7.1. Disfluency Annotation Model

Both of these approaches (repairing and tagging) involve modification at the word-level. From Jamshid Lou and Johnson (2020b), we use their state-of-the-art disfluency annotation model that is a self-attentive, self-trained constituency parser,

performing both disfluency annotation and syntactic parsing.
The model works by performing span classification on all possible spans in a string, for all possible positions in the string starting from position i to position j. Each span, from position i to position j, is assigned a label l, based on which label l receives the highest score $s(i,j,l)$ for that span. $s(T)$, then, is the overall score for the parse tree (Stern et al., 2017; Kitaev and Klein, 2018), as shown in Equation 1.

$$s(T) = \sum_{(i,j,l)\in T} s(i,j,l) \tag{1}$$

The best parse tree for a sentence, then, (out of all the possible parse trees) is the parse tree which receives the highest score for the sentence, as shown in Equation 2. Thus, the model outputs disfluency annotations at the parse tree level and at the word level. For our experiment, we consider these word-level annotations,[5] where each word is either *fluent* or *not fluent*.

$$\hat{T} = \underset{T}{\text{argmax}}\ s(T) \tag{2}$$

There are three types of parsing nodes which are considered disfluent: EDITED, PRN (parenthetical), and INTJ (interjection). The model is scored in terms of node labels–the model achieves the following scores for EDITED nodes ($P = 87.5, R = 93.8, F = 90.6$), and for EDITED, INTJ, and PRN nodes ($P = 92.5, R = 97.2, F = 94.8$) (Jamshid Lou and Johnson, 2020b). In terms of our disfluency definitions, "[f]illed pauses and discourse markers belong to a finite set of words and phrases, so INTJ and PRN nodes are trivial to detect" (Johnson and Charniak, 2004; Jamshid Lou and Johnson, 2020b).

7.1.1. Experimental Details

For both fine-tuning approaches, we fine-tune BART, T5, and Pegasus on the summarization task across 5 different seed values for 4 epochs, with a batch size of 4. All experiments are run on a single NVIDIA TITAN Xp GPU, and a single NVIDIA RTX A5000 GPU. We report the average results across the 5 different seed values.

Training Set. We use a randomly selected 1% of the podcasts from the Spotify Podcast Dataset (Clifton et al., 2020) to fine-tune our summarization models. For the dataset size of 105,360, this leaves us with approximately 1,053 podcasts in our uniformly randomly-selected 1% training sample (there may be less, due to some podcasts not having audio, and hence not having a transcript in their first minute). For our test set, we maintain consistency by using the same test set (of size 1,020) from the TREC 2020 Podcasts Track (Jones et al., 2020).

[5]We modify the code from Jamshid Lou and Johnson (2020b) to annotate *all* EDITED, PRN, and INTJ nodes as disfluent.

7.2. Results

7.2.1. Impact on Quality at Inference Time

How do these two methods impact summarization quality at inference-time only? In Table 2, we examine the impact of *repairing* (removal of disfluent words) and *tagging* (adding tags `<DIS>` and `<\DIS>` around disfluent words).

We see that simply using the test set as-is (i.e. without either modification, repairing or tagging) yields the best ROUGE-L, ROUGE-1, and ROUGE-2 scores in most cases. However, for Pegasus, utilizing the output of Jamshid Lou and Johnson (2020b) for the purpose of removing disfluencies increases the summarization performance in terms of the ROUGE-L and ROUGE-1 score at inference time. Thus, Pegasus is more robust in the face of missing information, and benefits from having the disfluencies removed.

We hypothesize that *repairing* the transcripts may not produce the best results in all cases due to the disfluency annotation model (Jamshid Lou and Johnson, 2020b) removing words which are not disfluencies, and therefore removing salient information from the models' input. It makes sense then that *repairing* is second to simply leaving the transcripts as-is. We can also see how *tagging* may add additional noise to the input, and therefore not yield the best results. All-in-all, leaving the test data as-is is the best option for BART and T5, and *repairing* yields good results for Pegasus.

Table 2: **Inference Results. Bold** indicates the best performance for each ROUGE metric. *R* indicates that disfluencies were removed. *T* indicates that disfluencies were tagged at the word-level using `<DIS>` and `<\DIS>`.

Model	Test	Rouge-L	Rouge-1	Rouge-2
BART	test$_R$	0.137	0.211	0.053
	test	**0.138**	**0.212**	**0.054**
	test$_T$	0.137	0.209	0.052
Pegasus	test$_R$	**0.131**	**0.200**	0.047
	test	**0.131**	0.198	**0.049**
	test$_T$	0.113	0.169	0.038
T5	test$_R$	0.133	0.194	0.050
	test	**0.134**	**0.199**	**0.051**
	test$_T$	0.126	0.181	0.048

Table 3: **Fine-Tuning.** *R* indicates that disfluencies were removed. *T* indicates that disfluencies were tagged at the word-level using `<DIS>` and `<\DIS>`. **Bold** indicates the best performance for each ROUGE metric (ROUGE-L, ROUGE-1, and ROUGE-2).

train	test	BART			T5			Pegasus		
		R-L	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2
train$_R$	test$_R$	0.172	0.240	0.085	0.145	0.197	0.059	0.129	0.174	0.049
	test	**0.177**	**0.244**	**0.090**	0.146	0.196	0.060	**0.131**	0.177	**0.052**
	test$_T$	0.174	0.241	0.086	0.148	0.198	0.063	0.096	0.133	0.037
train	test$_R$	0.170	0.236	0.083	0.146	0.198	0.060	0.122	0.165	0.045
	test	0.175	0.242	0.088	**0.149**	**0.200**	0.062	0.126	0.169	0.049
	test$_T$	0.172	0.238	0.085	0.147	0.194	**0.065**	0.090	0.124	0.032
train$_T$	test$_R$	0.172	0.238	0.083	0.142	0.193	0.057	0.129	**0.193**	0.048
	test	0.173	0.240	0.085	0.143	0.194	0.057	0.127	**0.193**	0.047
	test$_T$	0.169	0.235	0.081	0.145	0.196	0.058	0.115	0.146	0.038

7.2.2. Impact on Quality with Fine-Tuning

How do these two methods impact summarization quality with additional fine-tuning? In Table 3 we examine the impact of *repairing* (removal of disfluent words) and *tagging* (adding tags `<DIS>` and `<\DIS>` around disfluent words) the podcast transcripts by ablating the original train and test set, their repaired versions, and their tagged versions.

First, looking to the rows, we see that there are two approaches which stand out as being the most effective: (1) train$_R$/test, training on the *repaired* transcripts and testing on the *original*, unaltered transcripts, and (2) train/test, training and testing on the *original* transcripts.

In the case of train$_R$/test in the second row, we see that BART and Pegasus obtain their maximum scores across every ROUGE metric – with the exception of Pegasus' ROUGE-1 score. We hypothesize that training on the *repaired* transcripts is beneficial because the removal of disfluencies allows the model to focus on learning how to summarize spoken content (rather than the written content that these models were trained on) without the additional noise of disfluencies. However, at testing time, we hypothesize that the disfluency annotation model (Jamshid Lou and Johnson, 2020b) marks important, salient information as being disfluent, and it is this information that is removed, leading to degradation in the summarization quality. Hence, the best choice for the test set is simply to use it as-is, as to retain all important information.

Next, in the case of train/test in the fifth row, we see that this combination yields top scores as well. This makes sense, as *repairing* the transcripts leads to the loss of information.

Finally, we notice that there are two situations where utilizing *tags* is beneficial: the ROUGE-2 score of T5 and the ROUGE-1 score of Pegasus. While these specific disfluency tags may not have been seen in the models' vast training data, these models have certainly seen tags before. As such, we hypothesize that they may have captured knowledge of the generic tagging structure, and thus are able to interpret tags in this context.

8. Conclusion

In this paper, we have studied the impact of disfluencies in spoken content on automatic summarization methods. By amplifying the presence of three types of disfluency – repeats, interjections, and false starts – we have explored the impact of an increasing number of disfluencies on six different summarization approaches in an extreme stress-test. We find that summarization quality degrades with an increase in these disfluencies and that a combination of multiple disfluency types leads to even greater degradation. We also investigate two approaches to improve the performance based on a state-of-the-art disfluency detection model from Jamshid Lou and Johnson (2020b), and find that this tool can be helpful for improving performance in the face of regular, non-augmented disfluencies.

By diving deeper into the interplay of disfluency and summarization, we aim in our continuing work to identify new directions going forward toward improving the quality of summarization as well as enabling better models of speech itself. Recent research has indicated that disfluencies can be helpful for memory (Diachek and Brown-Schmidt, 2023): that is, interjecting "ums" and "ahs" can bring extra attention on upcoming material. Hence, there may be an opportunity to revisit existing summarization models in the context of better incorporating disfluency as a "feature" for identifying critical aspects of spoken content.

9. Limitations

This work considers a limited set of disfluency types. There are other types of disfluencies which could be considered as well in addition to the three we considered in this work. A wider range of disfluency types being systematically studied could be helpful for creating inclusive technology.

This work considers a limited subset of the podcasts from (Clifton et al., 2020). The Spotify Podcast Dataset is a large, heterogeneous dataset. Therefore, it may be beneficial to investigate performance on different parts of the dataset.

10. Ethical Considerations

One approach investigated in this work involves deleting disfluencies completely from text data, which in many applications, could end up treating disfluent speakers unfairly. In certain systems, it may be more useful to honor the disfluencies (e.g., create a variable which indicates the level of disfluency for a given speaker), and potentially use them for certain purposes (e.g., so the technology is accessible to a wide range of speakers). The treatment of disfluencies should be carefully considered in the context of the overall system and done on a case-by-case basis. However, simply filtering out disfluencies is a cheap alternative that could make many systems more inclusive.

11. Bibliographical References

American Speech-Language-Hearing Association. 2023. Fluency disorders (practice portal). Accessed: 2023-10-19.

Eugene Charniak and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In *North American Chapter of the Association for Computational Linguistics*.

Rohan Chaudhury, Maria Teleki, Xiangjue Dong, and James Caverlee. 2024. DACL: Disfluency augmented curriculum learning for fluent text generation. In *Joint International Conference on Computational Linguistics, Language Resources and Evaluation*.

Leigh Clark, Benjamin R Cowan, Abi Roper, Stephen Lindsay, and Owen Sheers. 2020. Speech diversity and speech interfaces: Considering an inclusive future through stammering. In *Conference on Conversational User Interfaces*, pages 1–3.

Evgeniia Diachek and Sarah Brown-Schmidt. 2023. The effect of disfluency on memory for what was said. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 49(8):1306.

Qianqian Dong, Feng Wang, Zhen Yang, Wei Chen, Shuang Xu, and Bo Xu. 2019. Adapting translation models for transcript disfluency detection. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 6351–6358.

Alexander R Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. SummEval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409.

Google. 2023. Project Euphonia. Accessed: 2023-10-19.

Google Cloud. 2023. Speech-To-Text. Accessed: 2023-09-24.

Paria Jamshid Lou and Mark Johnson. 2020a. End-to-end speech recognition and disfluency removal. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2051–2061.

Paria Jamshid Lou and Mark Johnson. 2020b. Improving disfluency detection by self-training a self-attentive model. In *Association for Computational Linguistics*, pages 3754–3763.

Mark Johnson and Eugene Charniak. 2004. A tag-based noisy-channel model of speech repairs. In *Association for Computational Linguistics*, pages 33–39.

Rosie Jones, Ben Carterette, Ann Clifton, Maria Eskevich, Gareth J.F. Jones, Jussi Karlgren, Aasish Pappu, Sravana Reddy, and Yongze Yu. 2020. TREC 2020 Podcasts Track Overview. In *Text Retrieval Conference*.

Hannes Karlbom and Ann Clifton. 2020. Abstract podcast summarization using BART with longformer attention. In *Text Retrieval Conference*, volume 1266.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Association for Computational Linguistics*, pages 2676–2686.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Association for Computational Linguistics*, pages 7871–7880.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*, pages 3730–3740.

Claudio Lopez Lloreda. 2020. Speech recognition tech is yet another example of bias.

Potsawee Manakul and Mark Gales. 2020. Cued_speech at TREC 2020 podcast summarisation track. In *Text Retrieval Conference*.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Conference on Computational Natural Language Learning*, pages 280–290.

Ani Nenkova and Kathleen McKeown. 2012. A survey of text summarization techniques. *Mining text data*, pages 43–76.

Tatiana Passali, Thanassis Mavropoulos, Grigorios Tsoumakas, Georgios Meditskos, and Stefanos Vrochidis. 2022. LARD: Large-scale artificial disfluency generation. In *Language Resources and Evaluation Conference*, pages 2327–2336.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Rezvaneh Rezapour, Sravana Reddy, Rosie Jones, and Ian Soboroff. 2022. What makes a good podcast summary? In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2039–2046.

Johann C. Rocholl, Vicky Zayats, Daniel D. Walker, Noah B. Murad, Aaron Schneider, and Daniel J. Liebling. 2021. Disfluency detection with unlabeled data and small BERT models. In *Interspeech*, pages 766–770.

Priyanka Sen. 2020. Speech disfluencies occur at higher perplexities. In *Workshop on the Cognitive Aspects of the Lexicon*, pages 92–97.

Elizabeth Ellen Shriberg. 1994. *Preliminaries to a theory of speech disfluencies*. Ph.D. thesis.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Association for Computational Linguistics*, pages 818–827.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Aneesh Vartakavi, Amanmeet Garg, and Zafar Rafii. 2021. Audio summarization for podcasts. In *European Signal Processing Conference*, pages 431–435.

Subhashini Venugopalan, Jimmy Tobin, Samuel J. Yang, Katie Seaver, Richard J.N. Cave, Pan-Pan Jiang, Neil Zeghidour, Rus Heywood, Jordan Green, and Michael P. Brenner. 2023. Speech intelligibility classifiers from 550k disordered speech samples. In *International Conference on Acoustics, Speech and Signal Processing*, pages 1–5.

Shaolei Wang, Wangxiang Che, Qi Liu, Pengda Qin, Ting Liu, and William Yang Wang. 2020. Multi-task self-supervised learning for disfluency detection. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 9193–9200.

Shaolei Wang, Wanxiang Che, Yue Zhang, Meishan Zhang, and Ting Liu. 2017. Transition-based disfluency detection using LSTMs. In *Conference on Empirical Methods in Natural Language Processing*, pages 2785–2794.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

12. Language Resource References

Clifton, Ann and Reddy, Sravana and Yu, Yongze and Pappu, Aasish and Rezapour, Rezvaneh and Bonab, Hamed and Eskevich, Maria and Jones, Gareth and Karlgren, Jussi and Carterette, Ben and others. 2020. *100,000 podcasts: A spoken English document corpus*. [link].