Research Master Thesis

Increasing Readability with Disfluency Removal in Automatic Dutch Transcriptions

Martin Rorick Terlou

a thesis submitted in partial fulfillment of the requirements for the degree of

MA Linguistics
(Human Language Technology)

Vrije Universiteit Amsterdam

Computational Lexicology and Terminology Lab
Department of Language and Communication
Faculty of Humanities

Supervised by: Sophie Arnoult
2^{nd} reader: Luís de Passos Morgado da Costa

Submitted: June 30, 2023

Abstract

This thesis investigates the application of automatic labeling techniques and sequence-to-sequence (Seq2Seq) models for disfluency removal in Automatic Speech Recognition (ASR) transcriptions. The increasing consumption of audio and video media necessitates the need for accurate and accessible transcriptions. However, speech disfluencies often disrupt comprehension and readability, posing a significant challenge to transcription services. To address this, we explore two data labeling techniques for training token classification models and the use of Seq2Seq models for disfluency correction.

The first labeling technique generates disfluent elements and inserts them into clean data, while the second technique labels words based on the difference between a raw ASR transcript and its cleaned version. The models trained with these techniques were able to accurately identify disfluencies.

The Seq2Seq model, trained using a parallel corpus of raw and clean transcriptions, demonstrated strengths in correcting grammatical errors and enhancing stylistic elements, readability, and clarity. However, it struggled with removing certain types of disfluencies, particularly interjections, and was prone to hallucinations, especially with longer and noisier inputs.

In conclusion, this thesis highlights the potential of automatic labeling techniques for training token classification models and the promise of Seq2Seq models in disfluency removal. Despite some limitations, these methods offer promising possibilities for future research and development in disfluency removal in ASR applications.

Declaration of Authorship

I, Martin Rorick Terlou, declare that this thesis, titled *Increasing Readability with Disfluency Removal in Automatic Dutch Transcriptions* and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a Master's degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Date: 29 June 2023

Signed: Martin Rorick Terlou

Acknowledgments

I wish to express my deepest gratitude to all those who have made the completion of this thesis not only possible but an enriching journey of growth and discovery.

First and foremost, Justina, your support throughout my degree and this thesis has been invaluable. You've allowed me to dedicate myself entirely to my studies and, specifically, this thesis. I am forever grateful.

To Nithin Holla, my supervisor at Amberscript, your contribution to this work cannot be overstated. Your teachings, patience, and unmistakable passion for our work have allowed me to progress and learn a lot about NLP in both academia and business. Your deep knowledge and dedication have been nothing short of inspirational.

Professor Sophie Arnoult, my university supervisor, your understanding of linguistics, Natural Language Processing, and academic writing has been a great help throughout this process. You've guided me in the right direction and helped me avoid potential pitfalls.

I am incredibly grateful to the rest of the science team at Amberscript. Henry Turner and Henning Bartsch, from day one, you welcomed me as one of the team, creating an atmosphere of informality and intellectual curiosity. I appreciate your honest yet considerate feedback that helped me refine my work and broaden my understanding. I couldn't have asked for a more supportive team.

Lastly, I want to thank the entire Amberscript organization for providing me with the opportunity to do my internship in such a fun and educational environment.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Average weekly audio and video media consumption has increased by 6.5 hours over the past five years to 17 hours (Wyzowl, 2023). People are watching more videos, listening to more podcasts, and attending more virtual events. Local and national governmental debates must often be recorded and made public. To ensure such media is fully accessible, many broadcasting agencies and government bodies require media transcripts or subtitles. The European Accessibility Act even requires everyday products and services to be accessible to people with disabilities (European Parliament and Council of the European Union, 2019). Transcriptions and subtitles are one step in this process and allow those who are hard of hearing or have other linguistic impairments to access media and information.

The amount of media that needs to be transcribed is so large that it is too expensive and time-consuming to do so manually. For every audio or video media hour, many transcribers estimate a minimum of 4 hours required to create a transcript (Walford, 2001; Chen, 2022). Automatic speech recognition (ASR) software can speed up this process immensely. It can be used as an intermediate step that creates an initial draft of the transcript, which a human transcriber can then clean up.

Amberscript, the company hosting this thesis project, offers automatic and manual transcripts and subtitles. If a customer requests a manual transcript, the ASR output is only used to create an initial transcript of an audio source. A large team of transcribers then perfects this to ensure high-quality *clean-read* transcripts.

The automatic transcripts created by the ASR are verbatim. A verbatim transcript is a word-for-word transcript of a given audio source, which contains all kinds of speech-specific mannerisms and is not always as readable as one might like. Although not bothersome in spoken language, some words are usually avoided in written forms. Such speech disfluencies can make a text hard to follow. The reader needs to actively filter the interruptions to extract the meaning of the text. This can be especially true for complex texts, where the addition of disfluencies can make an already challenging text even more difficult to follow.

As a result, disfluencies often need to be omitted when transcribing audio for the purpose of reading along (such as subtitles) or for data archiving. To further enhance the usefulness of ASR systems, disfluencies that appear in the raw ASR transcripts should be automatically detected and removed.

Speech Disfluencies

Speaking is a wonderfully complex task that is easy for most people. It starts by collecting our thoughts to form a message we intend to convey. Then we have to structure this message to follow a logical pattern, form the sentences, and find the right words. To produce words, we need to vibrate our vocal cords and activate the muscles required to shape the oral tract. If done right, we create a sound sequence that makes up each word in our planned message (Levelt et al., 1999). As expected, much can go wrong in this process, and much does go wrong.

When people talk, they often use words or sounds that do not help them communicate their message. Someone might need some time to think before they can continue their sentence and might fill that silence with *uhm* or with actual words such as *like* or *you know*. Such speech disfluencies are usually caused by having an incomplete plan of our message. While speaking, we might realize we want to take the message in a different direction, and we need to adjust what has already been said (Lickley, 2015). This can cause us to restart or repair our utterances. Planning the new route could take some time, which can be filled with interjections. Disfluencies can also be caused simply by accidentally pronouncing a different sound than intended. Many disfluencies exist, but we focus on *same-turn* disfluencies in this thesis. These disfluencies are restricted to a single turn in a conversation (Shriberg, 1994). They can be grouped into several categories (Honal and Schultz, 2003):

- Filled pauses
  'Their cooking is amazing, yesterday they made ***uh*** *spaghetti'*
- Repetitions
  *'****I****, ****I****, I can do that'*
- Interjections
  'So he said ***like****, ****well****, I like you!'*
- Repairs
  'I went to the store... ***the supermarket****'*
- False starts
  'Would you... ***I really like the food****'*

Speech disfluencies are often viewed as a sign of poor linguistic ability, but they are a natural part of speech production. It is challenging to produce fully fluent speech in spontaneous conversations, and even people following a script may struggle to achieve it. Within Chomsky's theory of syntax (Chomsky, 1965), speech is affected by linguistic competence and performance. Linguistic competence is a speaker's subconscious knowledge of language rules like syntax and semantics. Linguistic performance is how well someone uses language in real-life situations. Different factors, such as memory limitations, distractions, and the social context of the conversation, influence it. Even those with a high level of competence may still produce speech disfluencies, making it more likely for speech disfluency to fall under linguistic performance. Better speakers don't necessarily have a better understanding of the language. Instead, they might better understand the social context or be better resistant to distractions.

Removing Disfluency

Some ASR systems directly ignore disfluencies. However, such models tend to lag behind the state-of-the-art pipeline where the ASR output is processed by a separate model from the disfluency removal (Mendelev et al., 2021). When seeing disfluency removal as an independent task, sequence tagging is one of the more straightforward approaches. Each word in a sequence is tagged as fluent/disfluent.

Another approach to the task uses sequence-to-sequence models, which can be used to create text free of disfluencies. Here, disfluency removal is treated as a translation problem, where an ASR transcript that contains disfluencies is considered one language, and a model is trained to translate it into a fluent version of the transcript. This method does not require word-level labels, but it does require sentence pairs containing both raw and cleaned ASR output. However, Wang et al. (2020c) have used this approach and relied on pseudo data for the disfluent sentence.

The present thesis focuses on sequence tagging, or *token classification*, and sequence-to-sequence text generation (Seq2Seq). Token classification models face a major limitation in disfluency removal due to the scarcity of labeled data. The Switchboard dataset Godfrey et al. (1992) offers disfluency labels for English, but such resources are lacking for low-resource languages. This thesis concentrates on Dutch, which may not be classified as low-resource overall but lacks direct disfluency-labeled datasets.

The Dutch dataset Spoken Dutch Corpus (Corpus Gesproken Nederlands, CGN) contains Dutch and Flemish audio fragments, including transcripts (Oostdijk et al., 2002). By combining the various annotations, it is possible to approximate speech disfluencies. Unfortunately, not all disfluencies can be extracted from this structure.

To overcome the labeled data shortage, Passali et al. (2022) generate disfluent elements and insert them into clean text. The proposed Large-scale Artificial Disfluency Generation (LARD) method allows them to generate repetitions, repairs, and false starts. While this approach can create a large amount of labeled data, the generated data may not be representative of real-world disfluencies, thus hindering generalization.

Seq2Seq models do not require word-level labels. They are usually used in machine translation. The task in machine translation is comparable to the present task of transforming disfluent text into fluent text. To train machine translation models, one requires a parallel corpus. Each sentence in the source language has a corresponding translation in the target language. These models can improve texts beyond just disfluency removal.

1.1 Motivation

Speech disfluencies present a significant challenge to providing accurate and readable transcripts. These disfluencies often disrupt the comprehension process, leading to confusion and misunderstanding. The issue becomes even more critical when considering individuals with hearing disabilities or other linguistic impairments who rely on transcripts or subtitles to understand and access the content. Therefore, it is crucial to establish an effective method to eliminate these disfluencies, ensuring readability in transcripts and subtitles.

Disfluencies pose an added challenge for companies like Amberscript that provide transcription services. Transcribing large volumes of audio and video content is time-consuming and costly. It becomes even more laborious when the transcripts have

to be cleaned up by human transcribers to remove the speech disfluencies. This is expensive and increases the turnaround time for delivering the transcripts to the clients. An automated approach to disfluency removal has the potential to improve efficiency, reduce costs, and enhance the quality of transcripts.

In the context of Amberscript, the need for disfluency removal goes beyond manual transcripts. The company's automatic transcript service, which directly provides the raw output from ASR, also faces issues with speech disfluencies. These disfluencies often compromise the readability of the transcripts and, consequently, the overall customer experience. An effective method for automatic disfluency removal can dramatically improve the quality of these automatic transcripts, reducing the need for further manual intervention. Furthermore, it can expedite the production of subtitles, as the majority of the time spent on subtitle creation is on removing disfluent elements from the transcript.

1.2 Contributions

This thesis addresses several research questions related to disfluency removal and its applications in ASR systems. The research questions that guided this study are:

RQ1. Can automatic labeling techniques be reliably used to create datasets used to train effective transformer-based token classification models for disfluency removal?

- Can artificially generated disfluencies (following the LARD method proposed by Passali et al. (2022)) be used as an effective training dataset for token classification models?
- Can automatically labeled data using the raw and clean transcript pairs approach be used as an effective training dataset for token classification models?

RQ2. Can sequence-to-sequence models be reliably trained using automatically aligned fluent-disfluent data to create effective disfluency correction models?

To answer these questions, the thesis investigates the effectiveness of two data labeling techniques for generating training data for token classification models. One of these techniques is a novel approach to automatically label disfluencies, which is particularly useful in settings where only raw transcripts and human-perfected transcripts are available.

The first labeling approach generates disfluent elements and inserts them into clean data following the method by Passali et al. (2022). This input data lacks any ASR-specific errors. The second approach deals with this potential downside by using raw ASR output and labeling each word in the raw data by its existence in the cleaned-up version of the transcript. Significant edits can cause the two versions to be misaligned, and heuristics are applied to select candidate sentences that have enough overlap between the two versions. The unsupervised nature of this approach can result in imperfect training data.

The two labeling techniques result in various datasets used as input for a Large Language model, specifically the Dutch RobBERT (Delobelle et al., 2020). The model is fine-tuned with a binary classification head categorizing each token as fluent' or disfluent'.

The Seq2Seq model uses the raw and cleaned transcript pairs to fine-tune the T5 model (Raffel et al., 2020). Although the model may produce some hallucinations, the similarity in length and meaning between the two texts could minimize this effect. However, the model may still hallucinate if the input is nonsensical due to the noisy ASR output.

The results of this thesis contribute to the development of better disfluency removal data by comparing two labeling methods and providing insights into their strengths and weaknesses. In response to RQ1, our findings indicate that automatic labeling techniques are promising for training transformer-based token classification models. The Levenshtein-based models were particularly effective, with a precision rate of approximately 75% on the disfluent class and a recall rate of around 50%. However, the method proposed by Passali et al. (2022), despite offering reasonable precision, showed reduced effectiveness in recall. This highlights that the Levenshtein-based models outperformed those based on the LARD approach, stressing the benefits of realistic representation of conversational language in labeled data.

In response to RQ2, the results show that while the Seq2Seq models did not surpass the token classification models in terms of the BLEU score, they demonstrated an array of strengths, including the correction of grammatical errors and improvements in stylistic elements, readability, and clarity. Particularly, the model exhibited proficiency in handling single and multi-word repetitions and repairs, though it showed less efficiency in removing interjections. The model's tendency towards hallucinations under certain conditions warrants further exploration.

In conclusion, our research demonstrates the potential of automatic labeling techniques for training token classification models and the promise of Seq2Seq models in disfluency removal. These findings not only illustrate the potential of these methods but also highlight the inherent complexity of disfluency removal tasks, underlining the need for continued research and development in this field.

1.3 Outline

The coming chapters of this paper discuss different elements of the thesis. Chapter 2 presents a comprehensive review of previous works, discussing speech disfluencies and how transformer models aid in identifying them via token classification and sequence-to-sequence text generation. Chapter 3 explores the approaches used in this study, explaining the data sources, automatic labeling techniques, and the structure of the models. In Chapter 4, the results are shared, examining the fine-tuning steps and the effectiveness of the models. An in-depth error analysis for both model types is offered in Chapter 5, focusing on how input length and noise amount influence performance and on specific errors made by the models. Chapter 6 opens a dialogue about the results and their potential implications, as well as opportunities for further research. Finally, Chapter 7 concludes the research.

Chapter 2

Literature Review

Automatic speech recognition (ASR) has seen significant advancements over the past few years, with applications ranging from transcript services to voice assistants. One of the challenges in this field is to provide readable transcripts. This partly means providing accurate word-for-word transcripts of the spoken text. The next step is detecting and handling disfluencies - interruptions in the flow of speech that include repetitions, repairs, and interjections.

Despite the importance of this issue, there is little literature on disfluency removal, particularly in languages other than English. This gap is especially pronounced in Dutch, where no specific research on disfluency removal appears to have been conducted. This lack of research is needed to ensure the development of more readable Dutch ASR system outputs.

This chapter provides an overview of the literature on speech disfluency and its automatic removal using Large Language Models. We explore the nature of disfluencies and their impact on spoken and written communication. We delve into machine learning for disfluency removal, giving a brief insight into Large Language Models. Finally, we highlight the difficulties encountered in present disfluency removal research and potential solutions.

2.1 Speech Disfluency

Disfluencies in speech are a natural occurrence that can be observed during spontaneous conversation. Disfluencies are not necessarily indicative of a speech or language disorder, as even fluent speakers tend to use 6 to 10 disfluent words for every 100 spoken words (Rochester, 1973; Shriberg and Stolcke, 1996; Ferreira and Bailey, 2004). These disfluencies often serve communicative and cognitive functions such as helping the speaker retain the conversational floor, signaling uncertainty, or managing the social dynamics of the conversation.

This thesis will focus on same-turn disfluencies, specifically filled pauses, repetitions, interjections, repairs, and false starts. It is important to clarify that the scope does not include *dysfluencies*, which refer to speech that is not fluent due to speech disorders (Bloodstein et al., 2021). Additionally, we will not be discussing cross-speaker or cross-turn disfluencies, such as *uh huh* (affirmation) and *ooooh* (astonishment), which are vocal responses to other speakers and are often produced during another's turn or as turn-taking cues (Goodwin, 1986).

Disfluencies are influenced by various factors, including message structuring, utterance planning, socioeconomic circumstances, discourse factors, and the communication context (Clark and Tree, 2002; Shriberg, 1994; Holmes, 1988). They typically indicate issues with speech planning and can be impacted by elements such as sentence length, complexity, lexical context, and prosodic phrasing (Goldman-Eisler, 1958; Tannenbaum et al., 1965; Beattie and Butterworth, 1979; Bell et al., 2003; Nakatani and Hirschberg, 1994; Lickley, 2015).

Models of speech production often utilize disfluencies as evidence, suggesting a hierarchical process that encompasses stages such as syntactic planning, lexical selection and access, phonological planning, and motor control, with disfluencies potentially occurring at any stage (Levelt, 1983; Holmes, 1988; Ferreira and Pashler, 2002).

Contextual variables like the topic of conversation and familiarity with the speaker can also influence the rate of disfluent speech (Schachter et al., 1991; Moniz et al., 2014; Bortfeld et al., 2001; Shriberg, 1994; Arnold et al., 2007). Higher disfluency rates are often observed in more demanding contexts, such as speaking to a stranger or discussing complex topics. However, at the same time, disfluency rates drop in unfamiliar situations, such as human-machine communication. This could be due to sociolinguistic variables. Age, gender, register, and language variations systematically impact the use and frequency of filled pauses (Fruehwald, 2016; Tottie, 2011, 2014).

In short, sociological, neurological, and linguistic factors impact the realization of disfluency. The large variability can make systematic annotation challenging. Various approaches define and structure disfluency to simplify annotation and create helpful corpora for further research. In this thesis, we will discuss the methods and converge into a categorization we will use throughout this thesis.

2.1.1 Patterns in Speech Disfluencies

The classification of disfluencies mentioned in Section 1 is not the only possible grouping. Some early works regarding speech disfluencies focused on a clinical perspective to distinguish *normal* and disordered speech. Such categories were often highly specific. For example, Mahl (1956) mentions categories as *repetition of partial words* (stuttering), *intruding incoherent sound*, or *tongue slip*. Johnson (1961) had a similar goal and introduced eight categories, such as incomplete phrases and broken words. Although these categorizations effectively contrast speech with and without stuttering, they are not as suitable for characterizing typical speech.

Shriberg (1994) took a computational linguistics approach and developed a system that included five categories. The goal was to highlight the regularity found in speech disfluency. Due to technological advancements, there was a growing need to compare spontaneous spoken language with written language. In their paper, they stress the structured nature of speech disfluencies. A disfluent segment includes several parts: the word, partial word, phrase, or partial phrase that needs to be fixed (called the reparandum), the point where the interruption occurs, the pause between the interruption and the resumed speech (known as the interregnum), the word or phrase that is repeated or corrected, and the continuation of fluent speaking. Figure 2.1 visually represents speech disfluency components.

Any disfluent element follows this structure to some degree. Shriberg states that an interregnum is optionally filled with a word but can also be left silent. The repair is usually vocalized but can be left out under certain circumstances. For example, if the speaker is interrupted or assumes the repair is unnecessary to make their utterance's

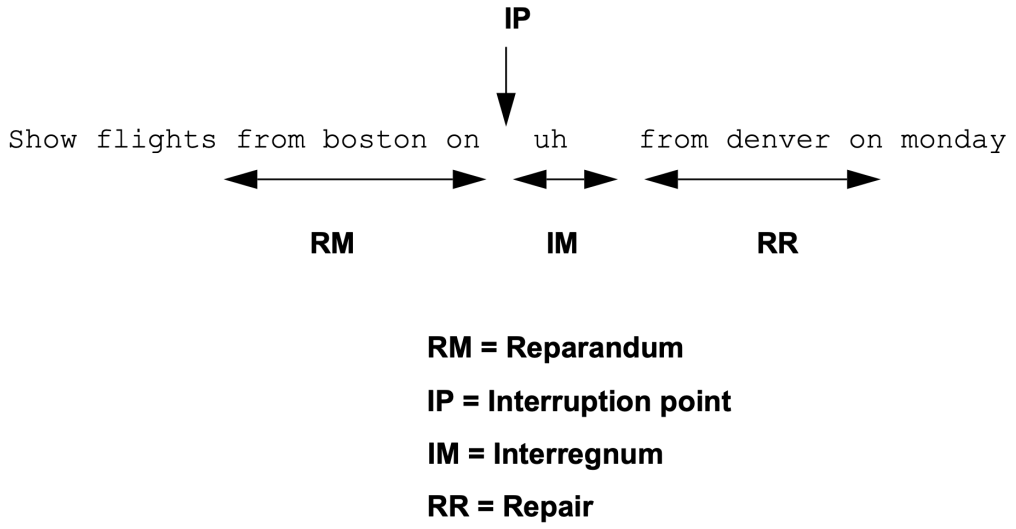

Figure 2.1: The structure of a disfluent phrase. Image from Shriberg (1994)

essence understood. Henceforth we will refer to the phrase that corrects the disfluent element as *correction*, not repair. We deviate from this since we will use the word repair as the specific class of disfluencies explained below.

2.1.2 Categorizing Disfluencies

Honal and Schultz (2003) introduced an intuitive categorization that follows the patterns described by Shriberg. Their paper discusses correcting errors in automatic speech recognition by implementing a disfluency correction system. They explain that their categorization method was based on the dataset annotation method used in their research. While Honal's classification system differs slightly from Shriberg's, as it uses broader categories, each disfluency type follows the same pattern as in Shriberg's system.

Filled Pauses Filled pauses are often non-semantic noises that fill a silence between utterances. Words like *uh* and *um* often give the speaker time to process or plan their utterance. On average, 1 to 4% of our speech comprises filled pauses (Lickley, 2015; Shriberg, 1994). Filled pauses are generally caused by high cognitive load, choices, or uncertainty (Arnold et al., 2007; Brennan and Williams, 1995; Smith and Clark, 1993). In Shriberg's structure, filled pauses are typically used as an interregnum.

Interjections Interjections share similarities with filled pauses and are commonly labeled as fillers. However, referring to them as such could lead to confusion with filled pauses. For clarity purposes, we will use the term interjections. Unlike filled pauses, interjections are standalone words. Although they may not have semantic meaning when used as interjections, they do have meaning in other contexts. For instance, the word *like* is commonly used as an interjection in English but can also hold significant meaning in typical usage. To illustrate this, consider the following example sentences:

(1) Examples of typical and interjectional use of *like*.

a. *I really like to go swimming*

b. *I really like want to go swimming.*

Example (1b) shows the disfluent realization of the word. Aside from a filled pause, the interjection can also fill the interregnum slot in Shriberg's structure.

Repetitions Repetitions are repeated words or phrases. It's important to differentiate between intentional and unintentional repetitions. Deliberate repetitions are used to emphasize a specific word. For instance, when saying *I really really like swimming*, the speaker wants to express their adoration for swimming. However, the use of *I* in *I I I really like swimming* does not emphasize that the speaker specifically likes swimming. This distinction is crucial as categorizing both types of repetitions as disfluency could result in losing important information if they're removed.

There is a ten times higher likelihood of unintentionally repeating a function word than a content word (Clark and Wasow, 1998). Function words may be more easily retrieved from memory when articulating a message due to their higher frequency, whereas content words may be more challenging to recall. However, repetitions may also indicate a speaker's attempt to maintain speech continuity when dealing with high cognitive demand (Hieke, 1981). This theory explains why repetition is more likely to occur at the beginning of complex clauses and more focused on content words appearing in that part of the sentence (Clark and Wasow, 1998).

Repairs The previous disfluencies are fairly straightforward and easily spotted. Repairs are disfluencies that are not as easily detected. Figure 2.1 shows a typical repair where the speaker intended to produce *from Denver on Monday* but instead uttered the name *Boston*. While *Boston* is the only error the speaker made, they correct it by also repeating surrounding the content words *from* and *on* as well. The entire section that needs to be corrected is called the reparandum, not just the one wrong word. We can determine what makes up the reparandum based on the repeated words in the correction.

Repairs can come in various forms. The disfluency type can include repetitions (as seen in Figure 2.1) or can include only a substitute for a single-word reparandum. Shriberg (1994) has identified multiple repair methods, including inserting, deleting, or substituting words. They show that a repair can also include interjections and filled pauses in the place of the interregnum. The many possible variations in repairs are also illustrated by Levelt's (Levelt, 1983) concept of covert repairs. Covert repairs refer to repairs where the reparandum is not vocalized. The assumption is that the speaker has the wrong word prepared in their mental plan of an utterance but corrects the error right before producing the word. Since this version of a repair is not audibly recognizable, we ignore it as it is realized as either a repetition (if surrounding words are repeated) or not as disfluency at all.

False starts A false start is when a speaker starts to speak, stops abruptly, and then resumes differently. False starts and repairs have similarities, and Shriberg does not differentiate them. The interrupted part follows the pattern of Shriberg's reparandums, while the restarted utterance speech is similar to corrections. However, false starts only

occur at the beginning of a sentence and can have longer reparandums than those found in repairs. Furthermore, false starts are usually not caused by an error in a content word, and the new sentence can be completely unrelated to the initial start. Another reason to differentiate false starts from repairs is that corrections in false starts involve a complete utterance, whereas corrections in repairs may only involve partial utterances (Tree, 1995).

2.2 Disfluency Detection

ASR systems typically provide a raw output consisting of a sequence of words that includes the speech disfluency in the audio. Having such verbatim transcripts can be useful since disfluency can carry pragmatic meaning, for example, in investigating linguistic phenomena or predicting sentiment in a text. However, a cleaner transcript is sometimes desired, and downstream models are used to improve the ASR output. A punctuation model can automatically predict sentence boundaries and capitalize sentence-initial words. A Named Entity Recognition model can identify and capitalize person names, company names, locations, and other entities. On top of adding punctuation and capitalization, detecting and removing speech disfluency from transcripts can improve their readability.

Rule-based approaches can target some disfluencies, but it soon becomes clear that the patterns of disfluency are difficult to capture. Repetitions can often be easily detected. However, not all repetitions are grammatically incorrect. Thus one would have to allow the model either to remove some repetitions falsely. Or one would have to specify which words would always be allowed to be repeated. Going into the territory of repairs and interjections, we realize that relying solely on rule-based approaches is no longer possible. All words that are considered an interjection in one utterance can be required in another. Any reparandum is also a correct word in its own right, and it is even hard for humans to identify them accurately. Machine learning is necessary for creating an automatic method of detecting and removing such disfluencies. Recent machine learning techniques offer a better semantic understanding of texts by machines, which is essential for detecting disfluencies.

2.2.1 Introduction to Machine Learning and Deep Learning

Machine Learning Machine learning focuses on developing algorithms and statistical models that enable machines to perform tasks otherwise only possible by humans. It draws on ideas from various fields, such as information theory, statistics, cognitive science, and mathematics (Cherkassky and Mulier, 2007).

Machine learning began with simple rule-based systems, where machines were programmed with specific rules to follow in processing data (Liu et al., 2015). Over time, researchers developed algorithms that allowed machines to identify patterns in data and make predictions without being explicitly engineered.

Machine learning can be categorized into three main types: supervised, unsupervised, and reinforcement learning (Qiu et al., 2016). Supervised learning involves training a model on a labeled dataset, where the correct answers (or labels) are provided. The model learns to predict the correct label for new, unseen data. Unsupervised learning does not require labeled data. The model learns to identify patterns and structure in the data without specific guidance about a correct answer. Reinforcement learning is

a type of machine learning where the model learns to make decisions by taking actions to maximize some cumulative reward (Shinde and Shah, 2018).

Common machine learning algorithms include linear regression, decision trees, support vector machines, and k-nearest neighbors. Each of these algorithms has its strengths and weaknesses and is used in different machine-learning tasks. However, they all rely on shallow architectures that involve only one layer of non-linear feature transformation. This single layer converts the raw input signals or features into the desired output, such as a class or label (Yu and Deng, 2010).

The architecture of shallow structures can be limiting because some features are too complex to transform with just one layer (LeCun et al., 2015). Deeper structures can overcome this issue by utilizing hierarchical structures to learn more complex representations automatically (Yu and Deng, 2010).

Deep Learning The start of deep learning marked a significant milestone in artificial intelligence. Deep learning leverages neural networks with many layers - hence the term *deep*. This approach was motivated by the need to overcome challenges that traditional machine learning techniques struggled with, such as image and speech recognition tasks (LeCun et al., 2015).

Several key factors facilitated the rise of deep learning. Firstly, the architecture of the models allows them to process large amounts and diverse types of data. Secondly, advancements in computing power, particularly the development of Graphics Processing Units (GPUs), made it feasible to process this data and train large neural networks (Dean et al., 2012).

Neural networks are at the center of deep learning. These are computational models inspired by the human brain, designed to recognize patterns. The networks consist of layers of nodes (or neurons), and they use a system of weights and bias adjustments in a process known as *training* to learn from data (Gardner and Dorling, 1998). The depth of these layers is what differentiates a deep neural network from a shallow one (Hornik et al., 1989).

While traditional machine learning and deep learning both fall under artificial intelligence, the two have fundamental differences. Traditional machine learning algorithms are often limited in their ability to process data in its raw form (LeCun et al., 2015). For instance, a piece of text, which is a string of characters, would require a feature extraction step to find words, syntactic features like word order, dependencies, and more. Conversely, deep learning models are designed to automatically learn these features, given enough training data.

Moreover, deep learning models are particularly well-suited to handling high-dimensional data, such as images, audio, and complex natural language processing (NLP) tasks. As we go further into the architectures of a specific deep learning architecture, a transformer, these advantages will become even more apparent (Vaswani et al., 2017).

2.2.2 Transformers

Before transformer models, much of the work in NLP with deep learning models used Recurrent Neural Networks (RNNs) and their extended variant, Long Short-Term Memory (LSTM) networks (Rumelhart et al., 1985; Hochreiter and Schmidhuber, 1997). These models processed input sequentially, making them suitable for handling the temporal dynamics of language. However, they had significant limitations, particularly when dealing with long-range dependencies in text. As the distance between relevant

pieces of information increased, RNNs and LSTMs struggled to maintain the necessary context (Bengio et al., 1994).

Attention The concept of attention was a possible solution to this problem. For neural networks, attention was first introduced by Bahdanau et al. (2014) in the context of neural machine translation. The idea was to allow the model to focus on only the relevant parts of the input sequence when producing each word in the output sequence. This would reduce the bottleneck of having to look at the entirety of a fixed-length vector. The *soft-search* was achieved by assigning a weight to each word in the input sequence for each word in the output sequence. These weights were learned during the training process, allowing the model to automatically learn which parts of the input were relevant for each part of the output.

The transformer model, introduced by Vaswani et al. (2017) took this concept of attention and extended it with the idea of *multi-headed self-attention*. Not only do the words in the target sentence attend to the words in the source sentence, but they also attend to each other. This is referred to as self-attention. Furthermore, instead of having a single self-attention mechanism, the transformer model has multiple attention *heads*. Each head learns a different set of attention weights, allowing the model to focus on different types of information. For example, one head might learn to pay attention to syntactic information, like the subject of a sentence, while another head might focus on semantic information, like the overall sentiment of the sentence. This multi-head attention mechanism allows the transformer model to capture a richer set of dependencies in the input data than possible with a single attention mechanism (Vaswani et al., 2017).

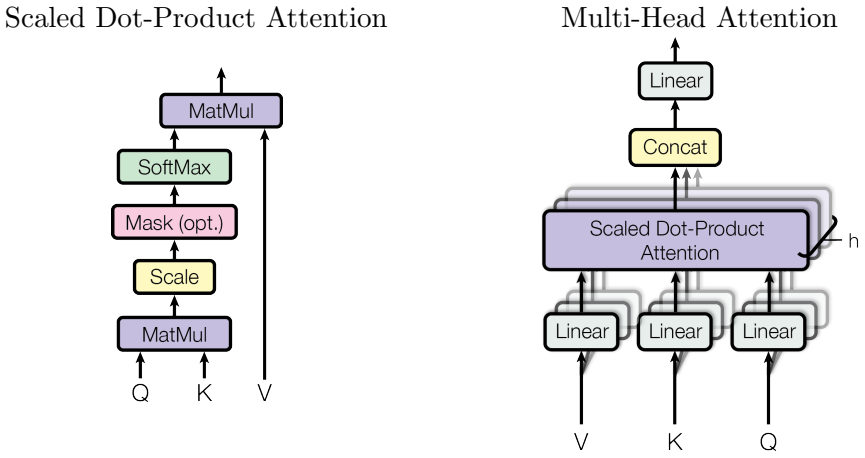

Figure 2.2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel. Taken from Vaswani et al. (2017)

In Figure 2.2, Q, K, and V stand for Query, Key, and Value, respectively. They are essential elements in the attention mechanism, which allows the model to focus on different parts of the input when generating the output. The Query represents the current word or context the model focuses on. The Key can be considered a set of

possible words or contexts that could answer the Query. Finally, the Value is the word or context corresponding to each Key. When a Key is selected as the answer to the Query, the corresponding Value is returned.

In the transformer model, Q, K, and V are vectors that are learned from the data during the training process. Specifically, they are created by multiplying the input (e.g., word embeddings for a given sentence) by weight matrices that are learned during training.

The multi-headed self-attention mechanism is used in the model's encoder and decoder parts. Multi-head attention allows the model to focus on different parts of the input simultaneously, which can capture various aspects of the data. The attention mechanism (Scaled Dot-Product Attention) is applied to the input (Q, K, V) h times (where h is the number of heads) using different learned weight matrices. This produces h different sets of Q, K, and V. These output vectors are concatenated and then linearly transformed once more to produce the final output.

Vaswani et al. (2017) used stacks of multi-head attention and fully connected feed-forward neural networks to create the transformer. The model used the more common encoder-decoder structure following the example of competitive models at the time (Bahdanau et al., 2014; Sutskever et al., 2014; Cho et al., 2014)

The introduction of the transformer model was a breakthrough in the field of NLP. It paved the way for a new generation of models that could handle complex language tasks more effectively than their predecessors. Since the publication of *Attention is All You Need*, transformer models have rapidly evolved, each iteration seeking to refine and improve upon the original concept.

2.2.3 Pre-training and Transfer Learning

This evolution has led to the development of models like BERT (Bidirectional Encoder Representations from Transformers), which used bi-directional attention architecture and presented and split the encoding layer from the decoding layer (Devlin et al., 2018). It introduced a novel pre-training method on a large corpus of text, enabling the model to function as a language model. This could then be fine-tuned for specific tasks.

BERT was a significant development in this area (Devlin et al., 2018). BERT was pre-trained on a large corpus of text using two tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP). The MLM task involved randomly masking words in a sentence and training the model to predict the masked words based on their context. The NSP task involved training the model to predict whether one sentence follows another. Through these tasks, BERT learned a rich representation of language that captured both the meaning of individual words and the relationships between words in a sentence.

Once a model like BERT has been pre-trained, it can be fine-tuned on a specific task with a smaller amount of task-specific data. This process is known as transfer learning. The idea is that the model can transfer the knowledge it gained during pre-training to the new task.

2.2.4 Large Language Models

After BERT, many models appeared that used similar architectures and included pre-training on large amounts of data. One popular adaptation was the *Robustly Optimized BERT* (RoBERTa) (Liu et al., 2019). RoBERTa is a variant of BERT designed to

further optimize the model's performance on NLP tasks. Like BERT, RoBERTa is pre-trained on a large corpus of text, but it uses a longer training time and a larger batch size, among other tweaks, to improve upon BERT's performance.

The architecture of Roberta is similar to that of BERT, consisting of a multi-layer bidirectional transformer encoder. However, Roberta removes the Next Sentence Prediction (NSP) objective used in BERT's pre-training and trains with much larger mini-batches and learning rates. Roberta also uses byte-level Byte-Pair Encoding (BPE), which allows for a more robust handling of word pieces. BERT uses a technique called WordPiece tokenization (Wu et al., 2016). This method breaks words into smaller units, or *WordPieces*, which allows the model to handle rare and out-of-vocabulary words. However, WordPiece tokenization is not without its limitations. It can struggle with languages that do not use whitespace to separate words and can sometimes split words in ways that do not align with linguistic boundaries.

BPE is a method that iteratively merges the most common pair of consecutive bytes in the data Sennrich et al. (2015). This method allows Roberta to handle a broader range of linguistic units without needing to split them in potentially unnatural ways. BPE is also language-agnostic, as it operates on raw bytes rather than language-specific characters or words. This makes Roberta more versatile and capable of handling a wider range of languages and linguistic phenomena compared to BERT.

RoBERTa achieved state-of-the-art performance on various benchmarks, demonstrating the effectiveness of its optimizations over BERT. It has since been used in a wide range of NLP tasks, including sentiment analysis and named entity recognition (Tian et al., 2020; Wang et al., 2020d).

Another significant development was the Text-to-Text Transfer Transformer (T5) model, which adopted a unified text-to-text format for a multitude of language tasks (Raffel et al., 2020). This approach allowed the model to handle tasks as diverse as translation, summarization, and question answering with a single, consistent framework.

The architecture of T5 is similar to other transformer models, with an encoder-decoder structure. However, it introduces several unique features and improvements. For example, T5 uses a causal masking strategy in its encoder, unlike the bidirectional encoders used in models like BERT and Roberta. This strategy only allows the T5 to use words prior to the mask being used (and attended to) in the prediction of the masked word. This limitation allows the model to be used for generative tasks, where the output is a sequence of tokens that can be generated one at a time. This type of model is also often referred to as a sequence-to-sequence (Seq2Seq) model.

T5 has been used in a wide range of NLP tasks, including translation, summarization, and question-answering. It has achieved state-of-the-art performance on several benchmarks, demonstrating the effectiveness of the text-to-text framework and the versatility of the T5 model.

Models like BERT, RoBERTa, and T5 motivated others to develop language-specific versions and multilingual versions. Usually, such adaptations stay consistent with the original structurally. Instead, they use alternate training data. For example, the Dutch version of RoBERTa, aptly named with a common Dutch name, RobBERT, is trained on Dutch data. The model was pre-trained on the Dutch section of the OSCAR corpus. This large multilingual corpus was extracted from the Common Crawl corpus (Suárez et al., 2019). The authors follow the exact method with which RoBERTa was trained.

Inspired by the success of language models like BERT, RoBERTa, and T5, re-

searchers started creating language-specific and multilingual versions of these models. These adaptations typically maintain the original structural design but are distinguished by the unique datasets they are trained on. For example, the Dutch adaptation of RoBERTa, fittingly named RobBERT, a common Dutch name. RobBERT's training is based on Dutch data, specifically pre-training on the Dutch segment of the OSCAR corpus (Suárez et al., 2019). This expansive multilingual corpus was extracted from the Common Crawl corpus. The creators of RobBERT followed the original RoBERTa training methodology to create this Dutch-focused adaptation.

The multilingual T5, or mT5, is designed to handle tasks in multiple languages. It leverages the same transformer-based architecture as the standard T5 but is trained on a multilingual dataset containing over 100 languages. This makes it capable of understanding and generating text in a wide variety of languages, thereby broadening the applicability of the T5 model beyond English and into a global context.

2.3 Disfluency Detection with LLMs

Token classification The pre-trained strength of transformers has also been applied to the task of detecting disfluencies. In a study by Bach and Huang (2019), a BERT model was fine-tuned on the Switchboard dataset, which is a large collection of English-language telephone conversations from the 1990s that includes annotations for various types of disfluencies (Godfrey et al., 1992). The researchers trained BERT to classify each word in a sentence as either fluent or falling into one of several disfluency categories, such as filled pauses, repetitions, repairs, and false starts. Results showed that the fine-tuned BERT model outperformed other models used in their experiments.

In their study, Wang et al. (2020a) enhanced the BERT model by fine-tuning it on the Switchboard dataset and incorporating an extra training layer. They implemented a *self-training* technique that exposed the model to both fluent and non-fluent data. The non-fluent data was generated by randomly adding or removing a word from a sentence. They found that this additional training step resulted in better performance compared to solely fine-tuning BERT.

Rocholl et al. (2021) also suggest that it's important to adjust a language model for conversational data before fine-tuning it. They believe that the BERT model's pre-training data doesn't have enough conversational data, making it less effective in handling this type of data. To address this issue, they conduct additional pre-training on the model using the same tasks as its original pre-training (MLM and NSP). Their study shows that this approach enhances the model's performance.

Previous methods have a disadvantage due to the lack of disfluency datasets annotated by humans. Most of these methods utilize the Switchboard data since it is one of the few datasets with human-annotated disfluencies. Although unsupervised approaches have been suggested, they tend to be less robust (Wang et al., 2020b; Saini et al., 2021).

In their additional pre-training step,Wang et al. (2020a) included artificial data by randomly adding or removing words from sentences to create disfluent sentences. Building on this idea, Passali et al. (2022) used more complex rules to generate disfluency, which closely follow the patterns described by Shriberg (Shriberg, 1994). They achieved near state-of-the-art results on the Switchboard using their method.

Sequence-to-sequence translation Aside from token classification approaches, disfluency removal can also be approached as a translation task, where a disfluent sentence is translated into a fluent one. Saini et al. (2021) used an unsupervised method to train a translation model based on previous research in style transfer models (He et al., 2020). Instead of labeling disfluent words and phrases, this method corrects them when generating a fluent version of the sentence. The unsupervised approach achieved a 79.39 BLEU score on the switchboard dataset. However, fine-tuning the model on a small set of parallel data improved the performance significantly, reaching an 85.28 BLEU score.

However, the BLEU score does not tell us much about one of the biggest risks in such generative models. Hallucination is defined as the generated content that is nonsensical or unfaithful to the provided source content (Filippova, 2020; Zhou et al., 2020). Ji et al. (2022) categorizes hallucinations into two types.

- Intrinsic Hallucinations: The generated output that contradicts the source content.
- Extrinsic Hallucinations: The generated output that cannot be verified from the source content

Even within intrinsic hallucinations, not all novel generations necessarily contradict the source content. When paraphrasing, one might generate a sentence that contains only novel words yet not change the source content. Similarly, simply adding one negation can flip the meaning of the generated text. Hence, as opposed to the assumption underlying the BLEU score, not all deviations from the source text are equal.

Fine-tuning a Seq2Seq model can reduce its hallucinatory tendencies. To do so, one requires parallel data. Parallel data consists of text in one language and its corresponding translation in another. The sentences need to be semantically aligned to make the parallel corpus useful for machine learning algorithms. Sentence alignment then refers to the process of finding corresponding sentences in two different language versions of a text. For example, if you have a book in English and its French translation, sentence alignment would involve identifying which sentence in the French version corresponds to each sentence in the English version.

The quality of the sentence alignment directly affects the quality of the resulting MT system. If the alignment is incorrect, the MT system might learn incorrect translations. Therefore, much effort is put into creating accurate sentence alignment methods. These methods often use linguistic knowledge (like punctuation or sentence length) and statistical methods to identify the most likely correspondences between sentences.

Past approaches to sentence alignment follow a dynamic programming approach. Gale and Church use the lengths of sentences in characters as the main criterion for alignment (Gale and Church, 1993). This method assumes that the length of the translated sentence in characters will be proportional to the length of the source sentence. It can also handle one-to-many and many-to-one alignments (where one sentence in the source language corresponds to several sentences in the target language and vice versa).

Another well-known method is Moore's, which relies on a greedy algorithm (Moore, 2002). Like Gale and Church's method, it uses the lengths of sentences as a criterion for alignment, but it also incorporates a lightweight translation model to further improve the alignment.

More recent approaches use LLMs to calculate the semantic similarity between two sentences. First, this method calculates the sentence embedding by pooling the word

embeddings extracted from an LLM. Then it measures the cosine similarity between the two embeddings, where a higher score refers to more similar sentences. One such approach is popularized by Reimers and Gurevych (2019).

All methods offer distinct sentence alignment approaches, each with unique strengths and weaknesses. The Gale and Church, and Moore methods rely heavily on the lengths of sentences as a primary criterion for alignment, making them simple and broadly applicable to a wide range of languages. However, this reliance on sentence length can lead to inaccuracies when sentence length doesn't correspond well across languages. This is also highlighted in disfluency removal as the targeted languages are the same, yet one of the key differences between a fluent and disfluent sentence is its length.

On the other hand, Reimers and Gurevych's approach focuses on semantic similarity between sentences, providing potentially more accurate alignments by going beyond mere sentence length and incorporating deeper linguistic information. However, this approach is more computationally intensive and might be slower when dealing with large corpora. This approach is also less sensitive to single words. Adding a few words will not change the embedding of an entire sentence much, while the alignment might decrease significantly.

Most of these approaches might not be necessary when creating parallel data for disfluency removal. Simple word-level comparisons might suffice since the source and target text are in the same language. Edit distance metrics can compare two sequences against each other and highlight what edits need to be made to go from one to the other. If no edits need to be made, we can assume the sequences are aligned. Usually, such metrics, like the Levenshtein edit distance (Vladimir, 1966), output a single score which is often used as a similarity score of the two texts (Wubben et al., 2010; Kutuzov, 2013). Texts with high similarity can be seen as good candidates for paraphrasing datasets since there is much word-for-word overlap.

Disfluency Removal in Dutch The field of disfluency removal has yet to be widely explored in Dutch, with few studies addressing this language specifically. Notably, none of these Dutch-focused studies incorporate deep learning methods, and all utilize the Corpus Gesproken Nederlands (CGN) dataset, which lacks predefined labels for disfluencies, thus constraining the scope and generalizability of their findings.

The first study investigated spontaneous speech recognition, highlighting disfluencies such as filled pauses, abbreviations, and repetitions. They proposed a method for detecting fillers prior to speech recognition, aiming to improve recognition accuracy by eliminating detected filled pauses from the recognizer input. This methodology, however, did not address other forms of disfluencies (Stouten and Martens, 2004).

A more comprehensive approach to disfluency removal was taken in another study, which defined disfluencies as any element that is not part of the main syntactic tree of a sentence as annotated in the CGN dataset. Beyond filled pauses, this included fragmented words, laughter, self-corrections, repetitions, abandoned constituents, and hesitations. Utilizing a memory-based learning algorithm (with a k-nearest neighbor algorithm), the researchers aimed to detect disfluent chunks based on a relatively small set of low-level features (Lendvai et al., 2003).

Although the existing research on Dutch disfluency removal has identified certain important aspects, such as filled pauses and a broader spectrum of disfluencies, it is evident that utilizing deep learning and conducting further research would be advantageous.

2.4 Summary

The primary purpose of this chapter is to define and categorize disfluencies and introduce previous attempts to detect them using transformer models. The discussed disfluency categories include filled pauses, repetitions, repairs, and false starts. These categories are based on the grouping by Honal Honal and Schultz (2003) and the disfluency patterns identified by Shriberg (Shriberg, 1994).

In this chapter, transformers were introduced, and their importance in recent NLP advancements was emphasized. Their most significant advantage is their ability to understand text in context, both semantically and syntactically. This makes them particularly effective in identifying disfluencies.

This chapter demonstrates two approaches to detecting disfluencies: token classification and sequence-to-sequence. Token classification trains the model to label each word in a sentence as fluent or one of several disfluency categories. Sequence-to-sequence treats disfluency removal as a translation task, transforming a disfluent sentence into a fluent one. Both approaches show promising results in disfluency removal and correction.

To achieve high-performing models, we learned that labeled or parallel data is necessary regardless of the approach used. However, due to the shortage of data in this field, researchers have explored unsupervised techniques and data augmentation methods as alternatives.

These methods are the basis for the approaches used in this thesis. The next section will explain the data used for the experiments and the labeling techniques applied to transform the data into datasets ready for training. We will also provide more details on the specific models used as a foundation and the evaluation setup done to evaluate the resulting models.

Chapter 3

Data and Methods

This chapter outlines this thesis's methodologies and data sources for disfluency removal. Our approach comprises two methods: token classification and Seq2Seq translation. Each method involves unique data processing and evaluation procedures, explained in the upcoming subsections.

The data for our experiments are derived from both Amberscript's proprietary data and the publicly accessible Spoken Dutch Corpus (CGN). These diverse data sources bring about specific challenges and benefits, which we discuss in this chapter.

We fine-tune the Dutch Large Language Model variant of RoBERTa (RobBERT) and T5 model on our datasets. Detailed information about the model selection and the evaluation metrics will be further elaborated in the subsequent sections.

3.1 Data

Both sources of data used for this thesis have opposing downsides. The in-house data, henceforth *Amberdata*, comprises raw transcripts generated by the company's ASR. Training any model on this means that there is no dissimilarity between training and future inference data. However, this data is not labeled, which requires automatic labeling techniques and heuristics to prepare it for training token classification models.

The CGN data, comprised of human-perfected verbatim transcripts, can be labeled for disfluency using the various existing annotations. It is openly available, which helps the reproducibility of the experiments. However, the transcripts differ from future inference data, likely causing the resulting models to perform worse than those trained on the *Amberdata*.

For Seq2Seq models, labels are not required, but we must collect correct sentence pairs of fluent and disfluent sentences. Here, automatic methods are also used to extract such pairs from the dataset.

3.1.1 Amberdata

When an Amberscript customer requests a manual transcript of their audio file, the file first gets processed by the ASR system. A professional transcriber then perfects the automatic transcript. Both versions of the transcript, the *raw* and *clean-read* transcript are stored separately. This allows us to extract both versions of transcripts whenever customers request a perfected transcript.

Professional transcribers correct any recognition errors made by the ASR system. If the customer requests a *clean-read* transcript, as opposed to a *verbatim* transcript, the transcriber will also clean up the texts by removing disfluencies and grammatical errors. Sometimes, a transcriber might even rephrase what was said to capture the speaker's intentions. Since verbatim transcripts maintain any speaker errors, including disfluencies, these transcripts are not suitable for training. Table 3.1 shows an example of the *Amberdata*.

Table 3.1: Example of the *Amberdata*. The first column shows a snippet of an automatic transcript, and the second column shows its *clean-read* version.

	Raw ASR	**Clean-Read**
1	Het is krijg ik een. Ik geloof ik. Ja, volgens mij moet ik het dan ook ergens zien. Ik zie nog niks. Juist. Ik zie het nu wel een ja, ja. Oké, nou hartstikke fijn. Ehm. Dr zijn een aantal vragen waarvan we afgesproken hebben dat we die allemaal zullen stemmen, maar die heb je volgens mij ook al gekregen in de mail. Nobetter u, misschien heb ik schrok gekregen, maar ik heb ze niet opgeslagen in mn hoofd. In ieder geval. Ah, nou, ik heb ze hier bij de hand.	Dus krijg ik een melding, geloof ik. Ja, volgens mij moet ik het dan ook ergens zien. Ik zie nog niks. Ja, ik zie het nu wel een rood knopje, ja. Ja. Oké, nou hartstikke fijn. Er zijn een aantal vragen waarvan we afgesproken hebben dat we die allemaal zullen stellen, maar die heb je volgens mij ook al gekregen in de mail. Nou, misschien heb ik ze wel gekregen, maar ik heb ze niet opgeslagen in mijn hoofd in ieder geval. Oh, nou, ik heb ze hier bij de hand.
2	En ehm, hoe vindt zij dat je je ex vrouw? Hoe noem ik haar niet of je vriendin? Mijn vrouw of vriendin, acht jaar dus, ze zegt ook: ze zegt: het went wel een beetje.	Hoe vindt zij dat, je ex-vrouw? Hoe moet ik haar nu noemen, je vriendin? Mijn vrouw of vriendin, wij kennen elkaar al acht jaar, dus ze zegt ook: "Het went wel een beetje"

The first thing that stands out in Table 3.1 is that the *clean-read* version is slightly longer than the raw transcript. The *clean-read* transcript contains some words that do not appear in the raw transcript, like `melding` (notification) in the first sentence of the first example. Perhaps this word was mumbled, not uttered, or missed by the ASR. There are very few instances where the ASR transcribed a word that the transcriber removed. Only one instance of disfluency removal can be found in this example. In the fourth line, the ASR transcribes `Ehm`, which is considered a filled pause and is removed in the *clean-read* version.

Often the transcriber needs to edit more of the ASR transcript than seen in the first example. The second example shows a transcript where a transcriber had to remove multiple repetitions and correct some grammar due to left-out words. The second sentence in this example also shows how transcribers sometimes rephrase a text to improve its readability. Freely translated, the raw ASR transcript reads: *"How do I call her not or your girlfriend?"*. This is changed to be: *"How should I call her, your girlfriend?"*. The result of this edit may not match the exact words spoken in the audio. However, if the transcriber can be certain about the intent, they have this freedom. Overall, the *clean-read* transcripts contain more changes than fixing ASR

errors. They fill in words a speaker skipped, correct punctuation, and remove words that are repeated or repaired.

All data for which customers had given their permission to be used for training was collected. This data includes the raw ASR and *clean-read* transcripts. The ASR system used by the company has improved over time, affecting the transcript quality. Not all examples given accurately represent the ASR's current quality. This gathering process resulted in a full dataset of 4709 texts. The data was split into a training, validation, and test set using an 80/10/10 split. Each text is a full transcript of an audio file. The lengths of these files range from 5 to 80611 words, with an average of 6374 words. For token classification models, each text is split into sentences. Since the data is unlabeled, 2000 semi-random sentences are taken from the validation and test set to be labeled manually. We explain more about the annotation process in section 3.1.2.

The sentences were selected to correctly represent various levels of *noise* found in real-life data. The ASR used in the company can sometimes easily transcribe the speech in an audio source, whereas other times, it has a harder time. Background sounds, mumbled speaking, and cross-speech all influence the accuracy of the ASR. We calculated the *noise* of a sentence based on its edit distance between the raw transcript and the *clean-read* transcript (for the full explanation of this process, please see Section 3.2.1). We created five noise levels using various thresholds and extracted sentences on each level. Level AA refers to verbatim text perfected by human transcribers, then levels A through D are increasingly noisy raw transcripts. Please refer to Appendix B for a further explanation of each noise level and the heuristic used to create it.

The *Amberdata* does not contain any labels, making it impossible to be used to train token classification models. Due to the data size, it is not feasible to label disfluency manually through annotation projects. Thus two different automatic methods are employed to label disfluency in the *Amberdata*. In the next section, we explain the process of each method and elaborate on the training datasets that result from them.

The nature of the *Amberdata*, in that for each text, a raw automatic transcript exists alongside a professionally cleaned version, makes the data ideal for training Seq2Seq models. However, due to the size of the transcripts, they need to be segmented into chunks that have the same meaning in both versions. The training dataset is created by selecting the best-aligned pairs using a similarity metric and other heuristics. For more information about the preprocessing method and data exploration, please refer to Section 3.3.

3.1.2 Annotation Study

To annotate the data, the help of four professional transcribers with significant experience working for Amberscript was enlisted. Each editor was asked to edit a total of 1000 sentences. Before beginning the annotation process, the editors participated in a training session conducted via video call. During this session, the editors were provided with detailed explanations of the task and annotation guidelines. Afterward, a trial round was conducted to see if the instructions were clear or if the guidelines needed to be adapted.

The trial round consisted of annotating 200 sentences following the guidelines provided. After completing the trial round, each annotator received feedback on their work and was able to give feedback on the task and guidelines.

Table 3.2: Inter Annotator Agreement of the annotation on the Trial Round.

Annotator Pair	Krippendorff's Alpha
1 - 2	0.417
1 - 3	0.523
1 - 4	0.661
2 - 3	0.287
2 - 4	0.379
3 - 4	0.572
Average	0.473

The task assigned to the annotators was to remove any disfluent words from the sentences they were given. The annotators were provided with a Google Sheets file containing two identical columns. While one column was provided for reference purposes, the annotators were instructed to use the second column to make necessary edits. They were only allowed to remove words and not replace them. Nor were they allowed to correct grammar or spelling. The annotators were told what disfluencies to look for following the disfluency types explained in Section 2.1.1. The full guidelines can be found in Appendix C.1 and C.2 for English and Dutch, respectively.

During the trial round, all annotators were given the same 200 sentences to annotate. An Inter Annotator Agreement (IAA) score was calculated with these annotations using Cohen's Kappa and Krippendorff's Alpha score (Krippendorff, 1970). The value of this score is not entirely representative of the final performance since the guidelines were adjusted based on the outcomes of the trial round. However, it does represent an early assessment of the task's difficulty.

Table 3.2 shows the IAA scores between each annotator and the average of those scores. Unfortunately, Krippendorff's Alpha has relatively low scores. A score of at least 0.67 is considered acceptable, but a good score should be around 0.80 Krippendorff (2004). Even the most aligned annotators (1 and 4) did not reach the minimum score during the trial round, indicating the complexity of the task. The data provided to the annotators is noisy, and the sentences they need to correct are cut pieces from a larger text, which can contain numerous word recognition errors. Some of the sentences are also incomprehensible. The low scores make it challenging to expect high performance from any model when evaluating them on the human-annotated validation set. Although excluding incomprehensible sentences from the validation and test sets would improve the scores, it would also reduce the validity of the evaluation since the datasets would no longer accurately represent real-life data. Therefore, it is crucial to maintain the integrity of the evaluation to avoid falsely positive expectations.

Additionally, we found that the annotators would often remove words that improved readability but did not match the patterns found in disfluencies.

(2) *Original sentence: And then we were able to also train them up better*
Cleaned sentence: Ø Ø we were able to Ø train them Ø Ø

While it is true that the edited sentence in this example is easier to read, the deleted words do not fully follow speech disfluency patterns. Since there is little context available, it could be possible that the words `then` and `also` served a semantic purpose.

The guidelines were adapted to address the issue found during the trial to clarify the differences between disfluencies and non-disfluent words and emphasize the importance of only removing disfluencies. The transcribers were given feedback on their work during the trial round and were given the new guidelines.

For the main annotation task, each transcriber received the 1000 sentences they were to annotate. The setup was identical to the trial round. Throughout the project, close contact was maintained with the annotators to answer any questions they had.

The annotators were only instructed to remove disfluent words and not to label each word individually. This approach was chosen to keep the annotators' task as simple as possible. However, this means that a different method must be employed to label the disfluencies in the original sentences. To do this, a comparison is made between the original and edited sentence word by word, starting at the end of the sentence. If the word is identical in both sentences, the corresponding word in the disfluent sentence is marked as *fluent*. Next, in both sentences, we move backward by one word, and the similarity between the words is assessed again. If the words differ, the corresponding word in the disfluent sentence is marked as *disfluent*, and only the disfluent sentence moves backward by one word. This process is illustrated in Figure 3.1.

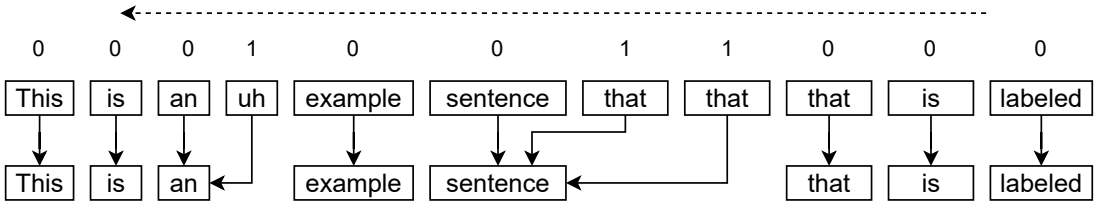

Figure 3.1: An illustration of the labeling process of disfluency. Labeling is done from right to left. Words in the source text (above) are labeled with *1* if the reference is identical, else *0*. In case of *0*, the target in the reference text moves one position forward.

This labeling process only works when the annotators follow the rule of only removing words. The process is consistently applied to all datasets for token classification, ensuring that the same disfluency pattern is always followed. For instance, in repetitions, the last word of the repeated sequence is marked as fluent, while the others are marked as disfluent. If the first word were marked as fluent, the sentence would remain the same, but the tokens would be labeled differently. This pattern also applies to repairs and false starts.

After the annotators completed their work on the full validation and test sets, the results were inspected for accidental mistakes, such as skipped sentences and the removal of non-disfluent words. Then the data is labeled using the method described above. The resulting sets are henceforth referred to as *Amber-val* and *Amber-test*. For inspection, we only use *Amber-val*. The 2000 sentences in this set have an average length of 16.52 words. Each sentence consists of around 12.8% disfluent words. However, this disfluency ratio depends significantly on the length of the sentence. Figure 3.2 shows the ratio for the sentence lengths that appear in the data.

We can mainly see that in sentences containing only three words, 60% are disfluent. In sentences of more than three words, the ratio is mostly between 0.1 and 0.2.

Figure 3.3 shows the normalized disfluency placement in a sentence. In the validation set, the disfluency predominantly appears at the start and gradually decreases as

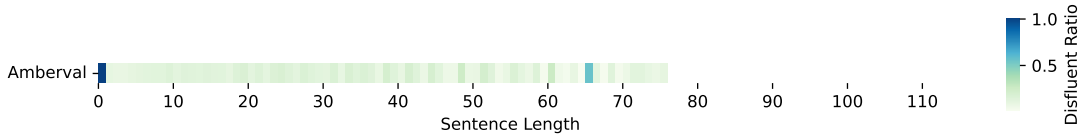

Figure 3.2: Disfluency ratio per sentence length in *Amber-val*.
The x-axis denotes the lengths of texts in the dataset. The colored blocks show the average ratio of disfluent words in the texts.

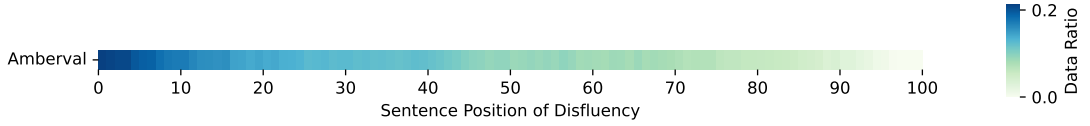

Figure 3.3: Normalized placement of disfluency in *Amber-val*.
The x-axis represents the length of a sentence where at 0, the sentence just started, and the first word would be placed. At 100, the last word is placed, and the sentence ends. Any length sentence is transformed to fit this range. The colored blocks show the ratio of sentences containing disfluencies at various places in the sentence.

we move to the end of the sentence. We also see that final disfluency is not occurring. This fits disfluency patterns, as any disfluency is followed by fluent words (either as a repair or repetition).

3.1.3 CGN

The Corpus Gesproken Nederlands (CGN) is a spoken language corpus containing recordings of spoken Dutch conversations from various regions in the Netherlands and Belgium (Schuurman et al., 2003). The corpus comprises scripted and unscripted speech, including monologues, dialogues, and spontaneous conversations. The CGN dataset is annotated with various linguistic information, such as parts of speech, dependency, and morphological tags. It has been used extensively in linguistic research, including studies on prosody, syntax, and speech recognition. The CGN is one of the largest spoken Dutch language corpora available, and it provides a valuable resource for researchers studying Dutch language and speech.

The full dataset consists of 900 hours of speech where every word was transcribed, including repetitions and incomplete words. Background noises and non-linguistic sounds, such as laughter, were marked separately. This verbatim transcript was done manually using the PRAAT software (Boersma and Weenink, 1992). A part of the data was annotated to show the dependency relations between words and phrases.

The data in CGN is categorized into 15 domains based on the content type. These domains include conversations between an interviewer and a teacher, live commentaries on sports, and political debates. However, not all of the domains contain unscripted speech. Scripted speech is less likely to have disfluencies, so only a portion of the data was chosen based on the audio source.

The dataset was refined by selecting Dutch texts (excluding Flemish) annotated with dependency relations. The resulting dataset consists of 71961 texts. This was further reduced using heuristics relying on the many layers of annotations as explained

below.

Labeling repetitions and repairs The CGN project has annotation guidelines available for all levels of annotation, including dependency tagging. In the syntax annotation guidelines, Hoekstra et al. (2003) explain that spoken language contains constructions that can be seen as malformed in written language. The authors highlight that annotators should create a separate dependency tree for the first instance of the repetition and the reparandum for repetitions and repairs. This smaller tree is not connected to the main tree. This means marking any word not attached to the main tree will capture all repairs and repetitions. No other information from the syntax annotations was needed.

Labeling filler words Having captured repetitions and repairs, we still need a way to label filler words. Even though filler words can be explicitly considered spoken language constructions, these words were added to the dependency trees. Luckily, such words are partly marked in the part-of-speech (POS) tagging procedure. In the guidelines of the POS tagging procedure, the authors categorize three types of interjection: onomatopoeia (expressing a dog's bark with *woof*), expression of emotions (describing pain with *ouch*), and specific formulas of social interaction (such as greetings, apologies) (Van Eynde, 2004). Each of these was tagged with *'TSW()'*. Tagging these words as disfluent also captures most filler words, including filled pauses.

The guidelines also explain the usage of 'special labels'. Eight special labels denote partial words, unintelligible words, background noises, and more. The ASR system employed by Amberscript is not capable of transcribing partial words or denoting background noises. So texts containing such elements are invalid candidates for model training. Specifically, any text containing the special labels: *T002*, *T003*, *T008*, and *T009* were discarded. They represent partial words, unintelligible words, descriptions of background noise, and annotator comments, respectively.

This process tags all three categories of disfluency: repetitions, repairs, and filler words (or pauses). However, inspection shows that the process is sometimes too harsh or lenient. In the examples below, words that are marked with Ø are labeled as *disfluent* following the process described above.

(3) Examples of disfluency marking in CGN.

a. *nou ze eten ze eten hier ontzettend veel vis*
Ø Ø

'well they eat they eat a lot of fish here'

b. *ze heeft voorsprong, tien meter.*
Ø Ø Ø

'She is in the lead, ten meters.'

Example (3a) shows an example where the tagging process missed a word that could be considered disfluency. The first word, `nou` (well), is often seen as a filler and here has no additional value to the text. The text preceding this may explain the decision not to mark this word, but any resulting model will not have this contextual insight either. However, such words remain unmarked in the CGN annotations and can thus not be automatically labeled as disfluency. The Amberscript transcribers do often remove such

words when editing the transcripts. This discrepancy is likely to affect performance models trained on the CGN data.

The opposite is shown in example (3b), which shows how too many words are labeled as disfluent. The entire phrase 'she is in the lead' is marked as disfluent since this part is not attached to the main dependency tree. This could again be due to contextual information. However, the aim is to limit ourselves to disfluency following the patterns of filler words, repetitions, or repairs. Using such data for training could result in a model that marks too many words as disfluent.

Applying heuristics Since the aim of CGN was not to annotate disfluencies directly, imperfect results are expected. To ensure the sentences used for training and testing are as valid as possible, some additional heuristics were implemented to select good candidate sentences. First, texts with less than one fluent word and fewer than two total words were removed. These texts are shorter than what is expected in future unseen data. Second, texts similar to the one in example (3b) were also removed, where more than 60% of the words are marked as disfluent. While somewhat arbitrary, the specific threshold of 60% was chosen to balance between removing too much data (and potentially useful learning examples) and retaining data that might negatively impact the learning of the model. Time constraints prevented empirical testing of different versions of the CGN data.

Finally, any duplicate found in the data was removed. Applying the heuristics further reduced the amount of data to 35939 texts. The number of remaining texts gathered from each of the relevant domains is shown in Table 3.3.

Table 3.3: Number of texts per domain in the processed CGN data

Domain	**Count**
Live commentaries	1949
Political discussions/debates/meetings	816
Spontaneous telephone dialogues	5145
Interviews/discussions/debates	3857
Spontaneous conversations	24172

All texts are then labeled in the same labeling method explained in Figure 3.1. Within the resulting data, there exists no predetermined split for training, validation, and test data. Thus the resulting dataset is randomly split into a train, validation, and test dataset using an 80/10/10 split. Figure 3.4 shows some details on the distribution of disfluent words in the training split of the selected CGN data.

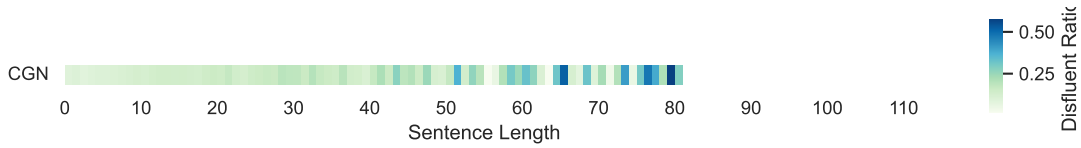

Figure 3.4: The ratio of disfluent words per sentence length in the training split of preprocessed CGN data.

The average percentage of disfluent words in a text is 11%, which is reflected in short sentences seen in Figure 3.4. For texts longer than approximately 40 words, we

see that percentage becomes higher. Meaning that in longer texts, on average more words are deleted. Longer sentences contain more information and thus allow for more instances of disfluency to arise. Furthermore, some disfluency types, like repairs and especially false starts, can only appear in long sentences. These multi-word disfluencies will cover more of a text than single-word disfluencies.

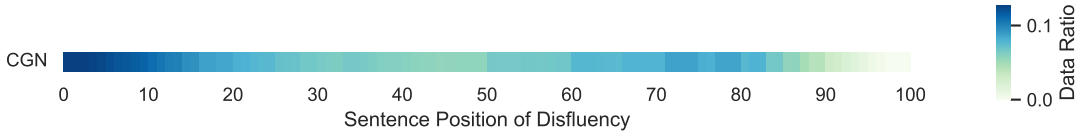

Figure 3.5: The normalized placement of disfluencies in a sentence in the CGN dataset.

Figure 3.5 shows that the distribution of disfluency is slightly more uniform across the sentence compared to the human-annotated data from Amberscript. A second difference is the second peak toward the end of the sentence, between 70%-80% of the sentence. However, like the *Amber-val* set, the CGN data also does not have sentence-final disfluency.

3.2 Token Classification

Token classification is the task of categorizing individual words in a sentence into predefined categories. Disfluency removal is one such task, where the goal is to identify disfluent tokens in a sentence.

Each automatic labeling technique is discussed in detail, including its advantages and disadvantages. The resulting datasets are analyzed and compared. Finally, the model that is fine-tuned using these datasets is explained. The process of fine-tuning the model is discussed, including the hyperparameter tuning and the main training process. Finally, the evaluation metric that is used for assessing the model's performance is presented.

3.2.1 Automatic Labeling

The first method uses the approach by Passali et al. (2022) to insert automatically generated disfluencies into *clean-read* transcripts. This approach runs the risk of producing unrealistic examples of spoken language or examples that are dissimilar to ASR transcripts. However, this method allows for the creation of large datasets with ease, potentially overcoming the aforementioned risk.

The second approach avoids this issue. It uses ASR transcripts and *clean-read* transcripts and uses the Levenshtein distance methodology to find all minimal edits needed to transform one text into another. This method provides realistic disfluencies similar to future inference data. However, the process could also mark words as disfluent that do not follow disfluency patterns.

LARD - Generate Disfluencies

The LARD method allows for the inclusion of repetitions, false starts, and reparandums Passali et al. (2022). The authors of the method explain how they add disfluencies of

each type to a sentence. We only use the *clean-read* transcript from the *Amberdata* as a base to insert disfluencies into. However, since the *clean-read* transcript can be very long, it was first split into sentences. This resulted in approximately 1.7 million sentences in the training data split of the Amberdata. Sentences with fewer than three words were ignored as they are unlikely to contain valid disfluencies.

This dataset was divided into four equal parts, with three parts utilized for generating various disfluency types and one part left untouched. As a result, the dataset includes an equal representation of fluent sentences and sentences with one of the three disfluency types. The process by Passali et al. for generating each type of disfluency is explained below.

Repetitions Following the LARD process, given a fluent sentence, 1 to 3 consecutive random words are picked. This word or phrase is then repeated a maximum of three times.

(4) Examples of repetitions generated with the LARD method. S_f/S_{dis} refer to fluent and disfluent sentences, respectively. words between '[ ]' are inserted.

a. *S_f: She was working very hard to finish the project*
   S_{dis}: She was working very hard [to to] to finish the project

b. *S_f: She was working very hard to finish the project*
   S_{dis}: She was working very hard [to finish] to finish the project

c. *S_f: She was working very hard to finish the project*
   S_{dis}: She was working [very] very hard to finish the project

Example (4a) shows a first-degree repetition, a single-word repetition, and is repeated two times. Example (4b) shows a second-degree repetition, a two-word phrase, and is repeated once. While this approach works well and is straightforward, there are some downsides. Example (4c) shows one downside. In most languages, words can be repeated to emphasize a message, which is more common in speech than in writing but is not considered spoken disfluency. Additionally, there are occasions where word repetition is acceptable, such as in the sentence, "I can't believe that that movie was so popular". This holds true for Dutch as well, especially the words: `je` (*you*) and `dat` (*that*).

False Starts False starts occur when an utterance is abruptly stopped and restarted differently. To create a false start, a fluent sentence is selected, and a second sentence is randomly chosen from the dataset. If the second sentence is not identical to the target sentence, it is cut at a random point in the first half of the sentence and added to the beginning of the target sentence. To minimize the chance of fluent words being outnumbered by disfluent words in the resulting disfluent sentence, only sentences that are at least four words long are eligible for a false start.

(5) Examples of false starts generated with the LARD method.

a. *S_{f1}: Yesterday he said some strange things.*
   S_{f2}: I will go to the store.
   S_{dis}: [Yesterday he] I will go to the store.

b. *S_{f1}:* *Yesterday he said some strange things.*
S_{f2}: Yesterday he went to the store.
S_{dis}: [Yesterday he] Yesterday he went to the store.

The authors note it is important to ensure that the start of the two fluent candidate sentences is not identical. Otherwise, the resulting disfluent sentence could be identical to a repetition, as shown in example (5b).

Reparandums The third type of disfluency is a reparandum, which involves selecting a random noun, verb, or adjective from a sentence as a repair candidate. Using the Python NLTK package (Bird et al., 2009), a list of synonyms and antonyms is obtained from WordNet (Fellbaum, 1998). A random synonym or antonym is then selected and, optionally, a random selection of 0 to 3 words that precede the candidate may also be chosen. Together with the antonym or synonym, these words form the reparandum and are inserted before the repair candidate.

(6) Examples of reparandums generated with the LARD method.

a. *S_f:* *That is very nice!*
S_{dis}: That [are] is very nice

b. *S_f:* *He went dancing with a friend*
S_{dis}: He went dancing [with a buddy] with a friend

c. *S_f:* *Then I wish you all the best!*
S_{dis}: Then I wish you [good luck {I mean}] all the best!

Example (6a) shows a sentence where only a single synonym was added in front of the original word. Note that examples are not limited to the official synonym or antonym definition. The process following Passali et al. also allows for different verb conjugations.

Example (6b) shows a sentence where, in addition to the synonym, one word was repeated. Finally, example (6c) shows that aside from the reparandum, it is possible to add a filler between the reparandum and the repair. It is randomly decided whether a filler is to be added. If one should be added, they randomly pick one from a list.

The LARD method is specifically designed to process English language data and search for synonyms and antonyms on the English WordNet. However, the Dutch WordNet (Postma et al., 2016) is less extensive, resulting in fewer recorded synonyms and antonyms for Dutch words. This limitation resulted in a reduced pool of replacement options, and some sentences had to be excluded during the generation process due to the lack of suitable candidate words. In cases where no reparandums can be generated, the entire sentence is ignored.

For each disfluency type, the generated elements are labeled as disfluent and all original elements as fluent. Each part of the data is collected back into a single dataset, and the order is then randomized.

The synthetic nature of the dataset is visible when examining the distribution of disfluent words per sentence in Figure 3.6. As sentences become longer, the ratio of disfluent words decreases. This is because each sentence only contains one disfluent element, which takes up more space in a shorter sentence than in a longer one. In actual speech, disfluencies can occur more frequently and unpredictably in longer sentences, which may not be fully captured by the LARD disfluency generation process.

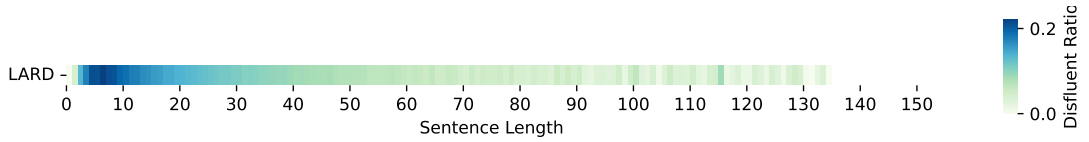

Figure 3.6: The number of disfluent words per sentence length in LARD data.

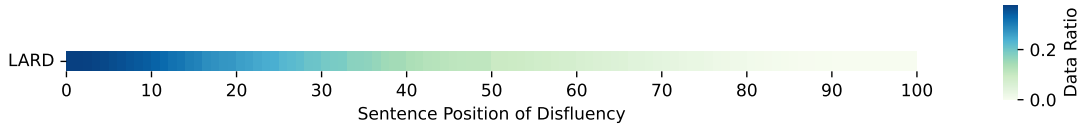

Figure 3.7: The normalized placement of disfluencies in a sentence in the LARD data.

In Figure 3.7, a similar trend to that in the human-annotated set can be seen, but much steeper. More sentences have disfluent words at the beginning of the sentence, whereas fewer are in the middle and end. This is likely due to a large number of false starts present in the dataset resulting from this method. In real data, false starts do not occur as often.

Levenshtein - Label Disfluencies

Levenshtein distance is a metric used to measure the difference between two sequences of characters. It is the minimum number of single-character edits (insertions, deletions, or replacements) required to transform one sequence into another. Using the standard Levenshtein example, the distance between the words `kitten` and `sitting` is 3, since three edits are needed to change `kitten` to `sitting`: replace `s` for `k`, replace `i` for `e`, and insert a `g` at the end. There are other ways to change the original string, but those will require more edits.

We use the Levenshtein Distance to label tokens in a sentence as either *delete*, *replace*, or *insert* based on the difference between the original ASR transcript sentence and the reference *clean-read* transcript. To do this, we used the Levenshtein package by Max Bachmann (Bachmann, 2022). The module labels elements in an original list according to a reference list using the shortest Levenshtein Distance. In the present case, we use a list of tokens that comprise a full text.

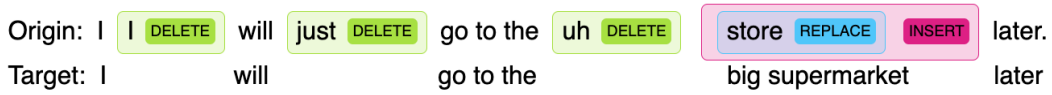

Figure 3.8: Example of a sentence tagged using the Levenshtein Distance.

Figure 3.8 shows an example of an original sentence from the Amberdata tagged with the Levenshtein labels. The words `i`, `just`, and `uh` are all labeled as delete since they are not present in the reference sentence. The word `store` received two labels: *insert* and *replace*. The *insert* label is given due to the word `big` in the reference sentence. Then, `store` would need to be replaced with `supermarket` as the next word `later` is present in both sentences.

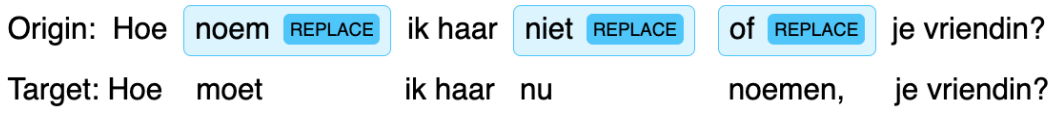

Figure 3.9: Example of an Amberdata sentence tagged using the Levenshtein Distance. *Target translation: 'How should I call her, your girlfriend?'*

Figure 3.9 shows an example using a sentence from the Amberdata. Here we see that there are no deletions but only replacements. We can see that such replacements serve no use for labeling disfluency. Problematic speech is caused by more than speech disfluencies. Grammatical errors are often left unrepaired because perhaps our conversational partners understand our intent or we do not realize the error.

Sometimes a transcriber judges an entire sentence as unnecessary and removes it completely. Perhaps the sentence is identical in meaning to the next but phrased differently. When we split the ASR transcript and the *clean-read* version into sentences, it is unlikely for the sentences to align. Thus, labeling the text on a sentence level is likely not going to work well. Instead, the text is labeled before splitting into sentences which can cause a sentence to only receive labels as *delete* or *replace*.

As seen in table 3.1, the transcriber also adjusts punctuation. This can affect the labeling method since words that have differing punctuation attached to them between the original and reference sentence are no longer identical. For example, the final word in Figure 3.9 is `vriendin?`. If the reference were not seen as a question, it would receive a *replace* label because the reference word would be `vriendin.`. For this reason, all punctuation is removed from the words before applying the Levenshtein labeling technique. Similarly, all words are ensured to be fully lowercase to ensure the casing does not affect the edit distance.

3.2.2 Data Selection

As described in the previous section, the method of using Levenshtein edit distance for labeling in disfluency removal is not perfect. Some sentences end up being completely deleted due to misalignment, while others are heavily edited with multiple replacements. Using heuristics, we can select optimal candidate sentences to reduce as much noise as possible. We define noise as words that either end up labeled as disfluent yet do not follow the definition described in Section 2.1 or words labeled as fluent but do follow that definition.

In this section, we will explain the different heuristics used to create various datasets. The novel nature of this application means experimentation is required to discover what heuristics create well-suited data for disfluency removal.

We implement a variety of parameters to create three initial datasets. These datasets are used for fine-tuning a pre-trained Large Language Model (see Section 3.2.5. The different heuristics we employ are the following:

- Ratio of replacements
- Ratio of insertions
- Ratio of deletions
- Sentence-final disfluency

- Class balance

The first decision is whether a sentence can have insertions or replacements and, if so, how many of each. Words with these labels cannot be labeled as disfluency, as shown in the previous section. However, sentences from ASR transcripts with many replacements or insertions are likely very different from the *clean-read* version. Any word labeled as *delete* in such sentences is not sure to be disfluent. Conversely, in sentences with solely correct and deleted words, the sentences are more similar, and words with the *delete* label are more likely to be disfluencies. Take the following examples:

(7) Examples of noisy Levenshtein tagging.

a. *Nee, de die rugzak*
replace insert^{x4}+replace

'No, the that backpack'

b. *En die goeie vriendin Jane Smith.*
replace replace replace replace replace replace

'And that good friend Jane Smith'

c. *Jep, dus ik ben begonnen.*
delete delete

'Yeah, so I have started'

Both sentences are part of a larger text. The labels given to the words do not follow disfluency definitions. If we correct example (7a) using only the sentence itself as context, all words would be correct except `de`, which is repaired with `die`. Instead, it's labeled as *replace*. Furthermore, the Levenshtein labeling process claims *rugzak* should be preceded by four other words and replaced. It is possible that the original and reference texts were misaligned or the ASR had recognition mistakes.

In example (7b), all words are labeled as *replace* even though no clear disfluency is present in the text. Thus it is important to select sentences carefully. When selecting sentences like example (7c) that only contain the labels *correct* and *delete*, we get a good approximation of disfluency, at the risk of not introducing enough noise.

The number of disfluent words in a sentence can vary a lot in a sentence. Similar to example (7b), sometimes all words in a sentence can be labeled as *delete*. Looking at context beyond the scope of the sentence, it might be clear that the entire sentence is disfluent. It could be an entire repetition or repair. Though in the scope of a single sentence, it is impossible to tell. If we use such sentences as training data, the model can learn to over-delete words regardless of whether they are disfluent in the scope of the single sentence. For insertions and replacements, and deletions, we can set a maximum ratio that is allowed to be in a sentence.

Sentence-final disfluency refers to sentence-final words labeled as disfluent by the Levenshtein labeling process. Following the definition of speech disfluencies, it is not possible for a disfluency to be sentence-final. Either through a repair or a repetition, disfluencies are always followed by fluent words. An exception might be filled pauses, which are easily removed in a rule-based manner. However, not having such sentences could limit the data available for training, while the impact might not be very big.

Severe class imbalance affects model performance since few examples are present for the model to learn the characteristics of underrepresented classes (Johnson and Khoshgoftaar, 2019). While the transformer architecture is less sensitive to class imbalance than other deep learning models, the effect could still be present if the imbalance is very large (Mustakim et al., 2022). On average, sentences only contain 20% disfluency. Balancing the number of fluent and disfluent tokens would not be feasible. However, if most of our data consists of completely fluent sentences, the model might not learn to mark disfluent words. To remedy this, we can balance the number of sentences with and without disfluency.

A final decision regards sentence length. Sentences shorter than three words are ignored. Such sentences are unlikely to contain disfluency, and it has little influence on readability if such short sentences are noisy. Since the source of the data is identical to future inference data, no upper limit is set across all datasets.

Several datasets are created using these heuristics. Table 3.4 gives an overview of the datasets and their heuristics.

Table 3.4: Heuristics overview of the Amberdata LS datasets

Heuristic	**LS1**	**LS2**	**LS3**	**LS4**	**LS5**
Insertion ratio	0	0	0	0	0
Replacement Ratio	0	0	< 0.20	< 0.40	< 0.40
Deletion Ratio	< 0.60	< 1.0	< 0.60	< 0.80	< 1.0
Sentence-final Disfluency	False	True	False	False	True
Class Balance	True	False	False	True	False

The first dataset in the table (*LS1*) was created to serve as the cleanest dataset we can make. The classes are balanced to ensure imbalance is not preventing the model from learning to remove disfluencies. No insertions or replacements are allowed to ensure alignment between the automatic and the *clean-read* transcripts. Furthermore, limiting insertions and replacements ensures no ASR errors are present in the automatic transcripts. The maximum ratio of disfluencies is limited to 0.60.

Datasets *LS2* and up all have a greater allowance of the various heuristics. Each is slightly more noisy than the last.

3.2.3 Dataset Overview

We investigate the performance of token classification models trained on six separate datasets. One dataset is the manually labeled CGN, and five are made using automatic labeling techniques (4 with Levenshtein and one with LARD). Table 3.5 shows some statistics on each dataset. One highlight from these statistics is that the LS2 dataset has a higher disfluency ratio than LS3 and LS4, which are supposedly more noisy. Remember that the term *noisy* here means the amount of possible incorrect labels in a sentence. These are increased when a sentence has many insertions or replacements as assigned by the Levenshtein labeling technique. This does not reflect the amount of disfluency present in the resulting datasets.

In addition to the LS, LARD, and CGN datasets, we combine all three into one large dataset. Under the assumption that each dataset is capable of capturing different patterns, combining the datasets might improve the performance. We combined the LARD and CGN datasets with the best-performing LS dataset on the validation set.

Table 3.5: Overview of all datasets used for token classification.

Dataset	**Samples**	**Sentence Length**	**Disfluency ratio**[1]
CGN	28k	9.60	0.107
Amberdata-LS1	278k	9.99	0.116
Amberdata-LS2	337k	9.62	0.212
Amberdata-LS3	560k	14.13	0.131
Amberdata-LS4	661k	14.24	0.144
Amberdata-LS5	796k	14.00	0.204
Amberdata-LARD	1.443k	26.44	0.163
Combo	1.808k	24.55	1.71

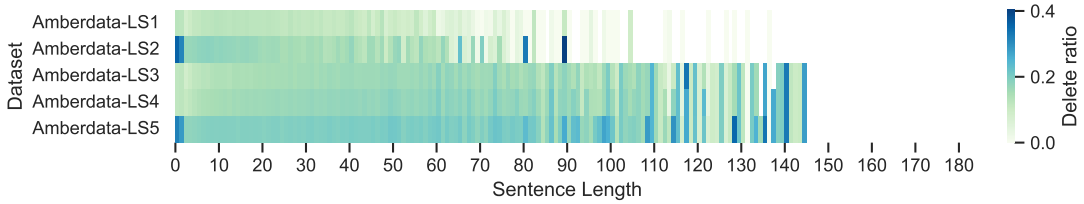

Figure 3.10: Disfluency ratio per sentence lengths in all Amber-LS datasets. To reduce space, the delete ratio is not displayed via bar height but via color changes

Figure 3.10 shows the distribution of disfluencies per sentence length. It is clear to see the effect of limiting deletions as LS1, LS3, and LS4 limit the number of deletions to 60% and 80%. The height of the bars in the short and long sentences is lower than in datasets LS2 and LS5, which do not limit the number of deletions. Allowing more replacements increases the total amount of data. This increase minimizes the variability between sentences of a given length and reduces the size of the error bars. For this reason, we see short error bars in all graphs in short sentences and taller error bars in long sentences.

In Figure 3.11, we can see that the average placement of disfluencies in a sentence follows the overall trend seen in the CGN and annotated data. However, we do see some differences within the various Amberdata-LS datasets.

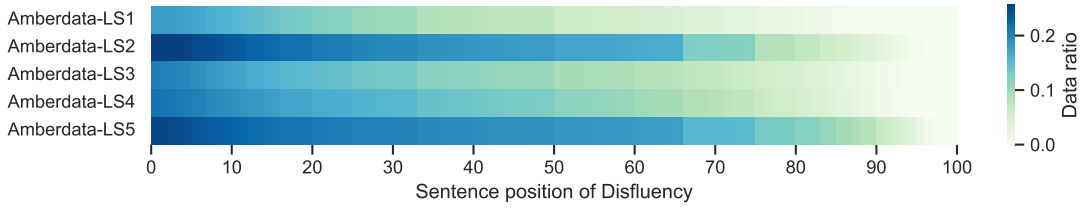

Figure 3.11: The normalized placement of disfluencies in a sentence in all Amber-LS datasets. To reduce space, the data ratio is not displayed via bar height but via color changes

Again, datasets LS1, LS3, and LS4 show a similar distribution, with an almost linear trend where the most disfluencies occur at the start of the sentence. However, datasets LS2 and LS5 have more sentences with sentence-initial disfluencies and more

sentences with middle-to-end disfluencies. The trend of these two datasets is closer to the CGN data, albeit with a higher overall disfluency ratio.

3.2.4 Rule-based model

We developed a rudimentary rule-based model alongside token classification models trained on the LS, LARD, and CGN datasets. While not the primary focus of the thesis, the rule-based model serves as a baseline for comparison with the other models. Given the novelty of disfluency removal in Dutch transcripts, there is no existing prior research to compare our results to.

The rule-based model simply removes all repetitions that span N words. It allows for M words to interrupt the repetition of the phrases, which will not be removed. Since some words in the Dutch language are allowed to repeat once, these words are ignored. We find the optimal N and M by validating the baseline on the validation set. The optimal values are found when we achieve the highest recall on the disfluent class while keeping the precision at 1.00. This resulted in N=3, M=1, and the ignored words `je` and `dat`. The rule-based model also removes filled pauses such as `uh` and `uhm` [2].

3.2.5 Model Design and Setup

Base model All created datasets are separately used as training data for fine-tuning the large language model RobBERT (Delobelle et al., 2020). RobBERT is a robust large language model based on RoBERTa. The authors report that the model outperforms other BERT models on small datasets and performs similarly on large datasets. The model was extensively tested on various classification tasks like sentiment analysis, NER, and POS tagging. The latter two tasks are relevant to the present task as they show the model is capable of word-level predictions on Dutch data.

Hyperparameter Tuning A hyperparameter tuning round was done to identify the optimal set of parameters for our models. It involves training multiple models with different combinations of hyperparameters and selecting the set that performs the best on the validation set. Only the CGN model and one of the Levenshtein datasets (LS2) are used for this. Fine-tuning RobBERT is an expensive task, especially on large datasets such as Amberdata. Doing hyperparameter tuning on all datasets was deemed too expensive and time-consuming. However, only doing hyperparameter tuning on either CGN or Amberdata could lead to unoptimized parameters as the datasets vary significantly in size and quality.

Fine-tuning Once we have identified the optimal hyperparameters, we fine-tune the remaining six models. The fine-tuning process involves training RobBERT on the target dataset.

3.3 Sequence-to-Sequence Text Generation

Seq2Seq models are a type of machine learning model used for generating sequences (Sutskever et al., 2014). A Seq2Seq model can be trained to generate well-written text based on a verbatim transcript. Token classification models can only identify disfluent

[2] A full overview of filled pauses can be found in Appendix A

words, and removing those words may result in ungrammatical text. On the other hand, the Seq2Seq model generates text based on its training data, so it's more likely to produce grammatically correct output if the training data is grammatical. Therefore, it would not only remove disfluent elements but also correct grammar.

3.3.1 Parallel Dataset Creation

As described in Section 3.1.1, the Amberdata consists of an automatic audio transcript and a *clean-read* edit of the same text. If the raw transcript is used as the input for the Seq2Seq model and the *clean-read* version as the target, the model can be trained to transform one to the other. The large size of many transcripts does not allow for such transformation. Thus the text needs to be split into smaller chunks.

The chunks of the raw and the *clean-read* transcripts must be semantically aligned. If not, the model might learn to replace words with semantically unrelated words or learn nothing at all, as the input is too varied. To ensure sentence alignment, the following process was followed.

Text alignment We use the Levenshtein labeling technique (see Section 3.2.1) to compare two texts and determine the minimal edits needed to change the raw version into a *clean-read* transcript. We then identify the first span of at least five words that should not be edited according to the Levenshtein process. The middle of this overlapping span serves as a *chunk-point* for both texts to ensure the start and end of each chunk are aligned. A minimum length of five was chosen since any shorter span could result in coincidentally overlapping phrases. Much longer overlapping spans could cause the resulting chunks to be too long.

Furthermore, if the *chunk-point* appears within the first ten words of the raw text, we ignore it and look for the next span. This is due to two reasons. First, since small texts are less likely to contain apparent disfluency, slightly longer texts can help the model learn the patterns. The second reason is to give ourselves some space, as the next step could shorten the text further. Our downstream Seq2Seq model can handle inputs of any length, but the computational cost increases quadratically with the input length (Raffel et al., 2020). To manage this, we've implemented a maximum length of 250 words for each input chunk. If a chunk exceeds this limit, we split the text at that length. This length allows one or more sentences to make up the chunk, each with disfluent elements.

The chunks produced by this process may not be ideal, as the overlapping spans can appear in the middle of a sentence. We considered moving the *chunk-point* to the nearest sentence boundary by searching for periods, question marks, or exclamation marks within a limited search window. This would improve the resulting chunks by having them start and finish normally. However, the ASR-generated transcript already has gone through a punctuation model that automatically predicts sentence boundaries. This model can be wrong, and some boundaries are not aligned with the *clean-read* version. This causes the alignment between the chunks to often deviate.

Data Selection Since the above-mentioned process is unsupervised, some pairs might coincidentally fit all constraints when chunking the full text but are semantically unrelated. Several heuristics are employed for selecting good training candidates. First, the cosine similarity between the raw and *clean-read* texts can give insights into the

semantic relatedness between the two (Reimers and Gurevych, 2019). The sentences are embedded using the `paraphrase-MiniLM-L12-v2` model (Reimers, 2022). Using the resulting vectors, the cosine similarity is calculated. Table 3.6 shows texts with various cosine-similarity scores.

Table 3.6: heuristics applied to the Seq2Seq training data.

Raw	**Clean**	**Cosine Sim.**
auto. Nee, nee, nee, klopt de kerk en achter. Dus om te weten: we gaan ons best doen, we gaan	auto. Nee, klopt. En we hebben een grote caravan erachter. Dus goed om te weten. We gaan ons best doen, we gaan	0.4
dit plan om goed te keuren. Dat wil eigenlijk onzichtbaar. Dank u wel. De heer van Janssen heeft een	dit plan ook goed te keuren. Dat was onze bijdrage. Dank u wel. Dank u wel. De heer Van Janssen heeft een	0.5
is geweest bij het hele ontwerp van een pak en daar hebben wij	is geweest bij het hele ontwerp van een park, en daar hebben wij	0.6
kan natuurlijk ook nog, hé voor wat lastig gespit want je zou denken	kan natuurlijk ook nog. Wat lastig is dit. Want je zou denken	0.7
auto, en dat heb ik dus gedaan en zij zeggen dat er geen Nederlandse Sensita is binnengekomen waarop gewerkt was. Ja, ik ben. Nee. Even kijken, nou, dan ga ik naar	auto, en dat heb ik dus gedaan en zij zeggen dat er geen Nederlandse Mercedes Vito is binnengekomen bij hun op de werkplaats. Ja. Even kijken. Dan ga ik naar	0.8

In Table 3.6, the sentences with a cosine similarity score of 0.4 or 0.5 semantically deviate. The edited sentences show that the ASR had significant recognition errors that could possibly encourage hallucination. However, higher-scoring texts still have some misalignment, although they become less problematic as the score increases. For example, the sentence-pair scoring 0.6 is identical except for a comma and the words `pak` and `park`. Supported by further manual assessment, a threshold of 0.65 is used, and text pairs with a score below this threshold are not used for training.

Even the example with a high cosine similarity score is not perfectly aligned. The word `Mercedes Vita` appears in the *clean-read* text with a 0.8 similarity score, but it's not included in the raw chunk, which shows a recognition error in the ASR transcript. Even pairs with high similarity can show multiple errors that are not disfluency errors. To avoid such instances, additional heuristics are added to remove most of these pairs. These heuristics limit the ratio of insertions, replacements, and deletions, similar to the Levenshtein-based heuristics applied in the token classification data. Table 3.7 shows the full heuristics for selecting training candidates. The resulting dataset has 613k text pairs ranging from 5 to 250 words.

Table 3.7: heuristics applied to the Seq2Seq training data.

Heuristic	**Thresholds**
Insertion ratio	< 0.40
Replacement Ratio	< 0.60
Deletion Ratio	< 0.60
Cosine Similarity	> 0.65

3.3.2 Model Design and Setup

Base Model The T5 model, specifically the *T5-base* version, is used for this task (Raffel et al., 2020). The T5 is a Seq2Seq model that reframes all NLP tasks into a unified text-to-text format. This framework allows the same model, loss function, and hyperparameters to be used on any NLP task, including machine translation, document summarization, question answering, and classification tasks. The T5 model is pre-trained on a large dataset called the Colossal Clean Crawled Corpus (C4) and can be fine-tuned on smaller labeled datasets for specific tasks. The T5 model has been trained solely on English data from the C4 dataset. On the other hand, the mT5 version has been trained on the entire dataset containing multiple languages, including Dutch Xue et al. (2021). However, due to limited computational capacity, we have decided to use the T5-base model as even the smallest mT5 model is larger. Although our target language is Dutch, we are assured that the large amount of data used for fine-tuning and the specificity of the task will yield satisfactory results. The purpose of the experiments described in this thesis is not to achieve a state-of-the-art disfluency removal system, but rather to demonstrate various approaches through proof of concepts.

Training Setup The training setup includes specific hyperparameters and the use of prefixes in training. When fine-tuning a T5 model, one can opt to use a prefix. When giving the model an input text, a prefix can be added that 'tells' the model what it must do. During its pretraining, the T5 model is prompted with several prefixes, like *summarize* or *translate from X to Y*. We set up a new prefix as no pre-existing prefix fits the present task. The prefix used in fine-tuning is *Clean transcription:* , which should guide the model to generate a clean version of the input text. Raffel et al. explain that while the prefix is essentially a hyperparameter, its influence when fine-tuning the model was limited.

3.4 Evaluation

The token classification models are evaluated with the precision, recall, and F1-score metrics. Additionally, we calculate the BLEU score so the models can be compared to the Seq2Seq model. The Seq2Seq model is only evaluated using the BLEU score.

Token classification It is standard practice to evaluate the performance of classification models with precision, recall, and F1-score.

- Precision is the proportion of true positive results among all positive results returned by the model. It measures the proportion of the items that the model identified as *disfluent* are actually *disfluent*.

- Recall is the proportion of true positive results among all actual positive results. It measures the proportion of the actual *disfluent* words the model was able to identify.

- F1-score is the harmonic mean of precision and recall. It provides a single metric that balances both precision and recall.

A deviation is made from the standard practice of relying on overall accuracy or averaged F1-scores during training. We focus solely on the scores of the disfluent class. This is due to the significant class imbalance present in our datasets, which results in averaged F1-scores providing a skewed representation of the model's performance. An overestimation of the model's performance can occur when relying on averaged F1-scores. Furthermore, the aim is for the resulting model to make as few false predictions that a word is disfluent. This will ensure all predicted disfluencies can be trusted. In addition, the goal of the model is to reduce the load of removing disfluencies by human transcribers. If the model removes too many words, the transcript can become unreadable, increasing the load for the transcribers instead. Thus, we focus specifically on the precision score of the disfluent class. Perfecting this will result in a sentence where no fluent items are missing.

BLEU The BLEU metric, BiLingual Evaluation Understudy (Papineni et al., 2002), is proposed for automated assessment of machine-translated texts. It offers quick, cheap, and language-independent evaluations that align well with human judgment. The concept of BLEU revolves around the idea of closeness. A machine translation is considered of superior quality if it mirrors the precision of a professional human translation. The BLEU score mainly determines the number of matching n-grams between a proposed translation and the reference translation without considering their positions.

More recent evaluation methods have emerged and are often preferred over traditional metrics like BLEU (Sai et al., 2022). Much like BLEU, they draw comparisons between the generated and reference texts but use syntactic structures or even word embeddings. Recent studies suggest that neural-based learned metrics outperform overlap metrics like BLEU, which do not have a strong correlation with human ratings (Freitag et al., 2022). However, due to the simplicity of implementing BLEU and the nature of the present thesis being a proof-of-concept for using Seq2Seq methods for disfluency removal, we choose to use standard BLEU.

For the purpose of this thesis, the *machine translation* expected by BLEU refers to the text that our model produces as *fluent* text. Additionally, a reference text is required for BLEU, for which we use the test data created for token classification. By eliminating all tokens marked as *disfluent* by the annotators, we can create a fluent sentence that serves as a reference. However, this method is not without drawbacks. Aside from disfluencies, these texts also contain grammatical errors and ASR errors. Therefore, it is not possible to view these references as a completely perfect or *gold* standard. However, to ensure some level of comparability between the token classification models and the Seq2Seq model, these references are used for the metric rather than a separate test set containing the edited transcripts (including grammatical and recognition corrections).

3.5 Summary

This chapter discussed the methods used for token classification in the context of disfluency removal. The chapter introduces the concept of token classification and describes the data used for training, which includes both labeled and unlabeled data. Automatic labeling methods are explored that potentially circumvent the need for labeled data. The approach from Passali et al. (2022) generates disfluencies by adding repetitions, false starts, and reparandums to sentences, while the Levenshtein labeling technique labels words based on the difference between the original and reference sentences. The heuristics used for data selection are explained, including the limitations and considerations in labeling disfluencies. The model design and setup are discussed, including rule-based baseline, the base model used (RobBERT), data formatting, hyperparameter tuning, fine-tuning process, and evaluation metrics.

The chapter also provides an in-depth explanation of the preparation of a parallel corpus for training a Seq2Seq machine learning model for text generation. The dataset is created using a raw ASR transcript and a *clean-read* version of the same text, which is cut into smaller, semantically aligned pieces. The cutting process involves using the Levenshtein labeling technique and finding overlapping phrases between the texts. Various heuristics are then employed to select good training candidates. The training setup makes use of a T5-base model, a Seq2Seq model pre-trained on the Colossal Clean Crawled Corpus (C4), which is then fine-tuned on the specific task of generating clean text from the raw transcript. The evaluation of the model's performance focuses on the use of the BLEU (BiLingual Evaluation Understudy) score, a widely accepted automatic evaluation tool for machine-translated texts.

Figure 3.12 gives an overview of all experimental components of the thesis.

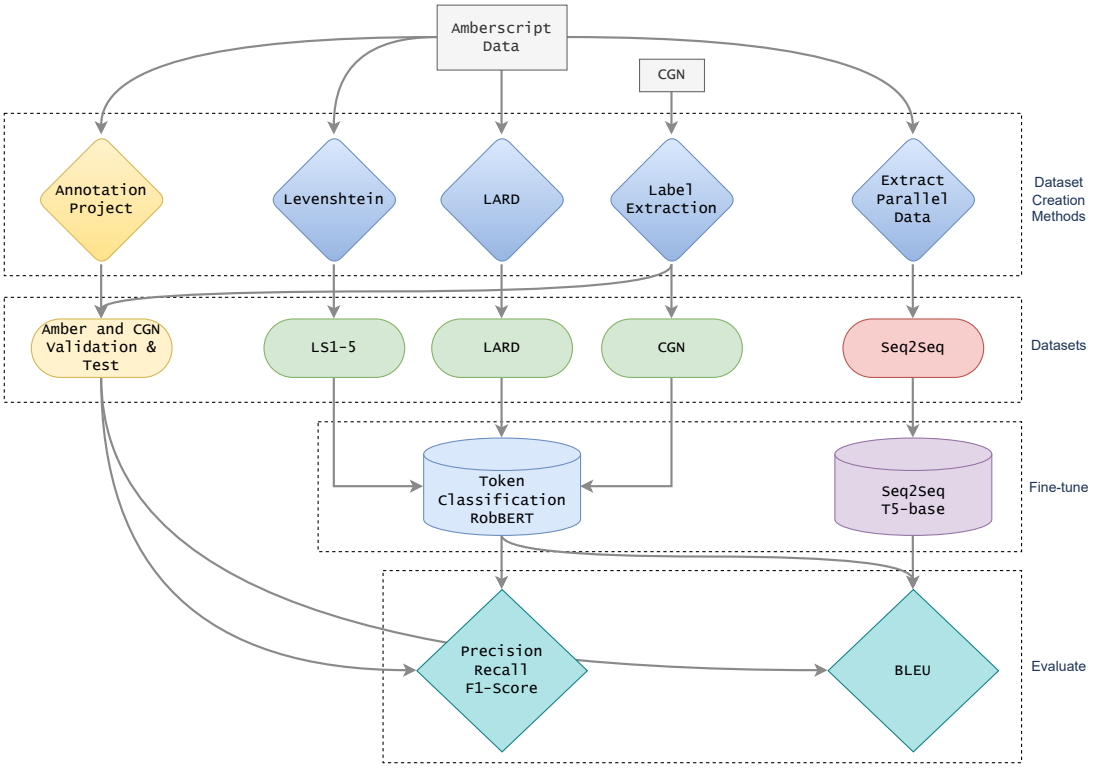

Figure 3.12: Overview of the experimental components of the thesis.

Chapter 4

Experimental Setup & Results

In this chapter, we first detail the setup of the experiments. We detail the specific, concrete steps we took to implement and evaluate these models. This includes the data preparation, the ranges of parameters used for hyperparameter tuning in the case of token classification, and the specifics of the training procedures for both token classification and Seq2Seq models. For each approach, we also discuss what metrics are used for the evaluation and on what test sets they are evaluated.

Then we give an overview of the results from a broad perspective. We show the results of the rule-based baseline, the eight token classification models, and the Seq2Seq model. We report on their performance using precision, recall, and F1-score for the token classification models and the BLEU score for all models. We briefly discuss visible trends and additional insights and compare the results to the rule-based baseline.

These scores in this chapter only give an initial insight into the models' performances. A deep dive into the performance is done in Chapter 5.

4.1 Experimental Setup

4.1.1 Dataset Overview

The data used in this study was partitioned into training, validation, and test sets (see Section 3.1.1 for the Amberdata and 3.1.3 for the CGN data). The training set was used to train the models, the validation set was used to fine-tune the hyperparameters, and the test set was used to evaluate the final model's performance. This partitioning was crucial to avoid overfitting and to ensure that the model's performance was evaluated on unseen data.

Token Classification We utilize seven distinct datasets for training token classification models. One of these datasets uses the CGN corpus and adds approximated disfluency labels through various existing annotations present in the corpus. The remaining six are the various *Amberdata* sets. These were automatically labeled using two labeling techniques. Five were labeled using the Levenshtein labeling technique (LS1-5), and one with the method proposed by Passali et al. (2022) (LARD).

In addition to these, we also created a combined dataset under the assumption that each dataset could capture different patterns, potentially improving the performance of the models. This combined dataset (Combo) included the LARD and CGN datasets along with the best-performing LS dataset based on the validation set.

Seq2Seq The single sequence-to-sequence dataset was made by cutting the raw transcript and *clean-read* version into semantically aligned pairs. This was done by again labeling the raw transaction using the Levenshtein labeling technique. The resulting dataset consists of 613k samples.

4.1.2 Data Pre-processing

Token Classification: Automatic labeling The novel approach using the Levenshtein edit distance uses a package of the same name (Bachmann, 2022). Specifically, we apply the `editops()` function using the raw transcript as the source and the clean transcript as the reference. Using the output, consisting of the name of the edit and the index in both the source and reference text, we label all tokens in the source text. Since a word in the source text can appear multiple times in the output of `editops()` when the reference text has multiple insertions at this index, we concatenate any multiple edits to create a layered label. Each part of layered labels counts toward the total of edits used in the heuristics.

Once we have the raw transcript with all the edits as labels for each word, we split the text into sentences using the `SpaCy` sentencizer (Honnibal and Montani, 2017). We then create a fluent sentence by removing all words that received the edit label *delete*. Using the two sentences, we relabel the disfluent sentence following the labeling process outlined in Figure 3.1. Finally, sentences are filtered to a minimum of 4 and a maximum of 250 tokens for all datasets.

The datasets used for the fine-tuning are stored as `json` files containing a column with the full sentence and a column containing the final labels.

Seq2Seq: Parallel Data Creation The preprocessing for the Seq2Seq data is limited. To cut the transcripts on overlapping spans, we apply the Levenshtein package's `editops()` function in the same way as for the token classification data preparation. However, in this case, after applying it, we find spans of indices where no edits are made. Once all pairs are extracted, we reapply the Levenshtein labeling technique and calculate the cosine similarity between the texts. We then use that information to apply heuristics, thus removing any severely misaligned pairs.

Formatting for fine-tuning The token classification datasets are labeled at the word level. However, the large language model used on these models, RobBERT, tokenizes the input into WordPieces, or *tokens*. This means that one word can be split into multiple tokens, and each token needs to be assigned a label. To align the word-level labels with the tokens, each token was assigned the label of its corresponding word.

The data preparation process, including tokenization and label alignment, was done using the `Hugging Face` package. This library was chosen due to its ease of use and its comprehensive set of tools for both data preparation and model training. In this setup, we use the tokenizer that is part of the *RobBERT* model hosted on Hugging Face to split each word into WordPieces. For the Seq2Seq data, we use the T5 tokenizer to split the text into tokens.

4.1.3 Model Configuration and Parameter Settings

Hyperparameter tuning In order to save on computation, we decided to do a limited hyperparameter tuning setup. No hyperparameter tuning is done for the Seq2Seq model as the size of this model is large, and training the model multiple times is too expensive. Due to the lighter architecture of token classification models, we decided to do hyperparameter tuning for these models. However, since the datasets are so varied in size, we decided to do hyperparameter tuning on two datasets: CGN and LS2. These datasets were chosen due to their size, CGN is a relatively small dataset, and LS2 is large. While Other LS datasets are larger, fine-tuning those would be even more expensive. LS2 will represent all automatically labeled datasets, and the best parameters found in its tuning will be applied to all other datasets. The ranges explored in the tuning are shown in Table 4.1, with the best-performing parameters highlighted in bold.

Table 4.1: Parameters used for Hyperparameter-tuning. Bolded values resulted in the best performance

Parameter	**Ranges**
Batch Size	**8**, 16, 32, 64
Learning Rate	3e-4, **3e-5**, 3e-6
Warm-up Steps	**500**, 100

Training Procedure The token classification models created in the hyperparameter tuning step are evaluated using precision, recall, and F1-score. We choose the best-performing model based on its precision of the disfluent class.

Once the best parameters are found, we start the fine-tuning process for each model[1]. During fine-tuning, the model's performance on the validation set is monitored with `MLFlow` (Databricks, 2018), and early stopping is used to prevent overfitting. We specifically set an early-stopping patience of *3*. This means that if the model does not improve its performance on the validation set for three evaluation steps, training will be interrupted, and the best model will be saved. Once again, we use the disfluent precision to judge a model's performance. We do this evaluation twice every epoch.

All token classification models are trained using four Nvidia V100 GPUs, each with 16GB of memory. On average, fine-tuning a model on a 300k sample dataset for a single epoch took around 4 hours.

The Seq2Seq model is trained on a single Nvidia T4 with 16GB of memory. All hyperparameters, except for the batch size, are kept to the default as used in the original T5 paper. The batch size had to be decreased from 128 to 16 to avoid memory issues.

The model's performance is also tracked on the validation set, and similar to the token classification model training, we also implement early-stopping patience of *3* to avoid overfitting the model. However, since the validation set contains only the raw texts and word-level labels, we cannot use it directly to evaluate a generative model. From the raw text, we strip all disfluent words resulting in a fluent version of the text. This fluent version is used as the target for the Seq2Seq model. We then calculate the BLEU score using Hugging Face's `evaluate()`. Specifically, we use the `sacreBLEU` metric.

[1]The results of the hyperparameter tuning can be found in Appendix D

4.1.4 Model Evaluation

To fully evaluate the token classification models, we use the precision, recall, and F1-score metrics. We use the `SKLearn` package's implementation `classification_report()` function (Pedregosa et al., 2011).

Since the task is novel (in the case of Dutch and its approach), no existing scores are available on public datasets. Using a *dummy-classifier* (one that always predicts the majority class) as a baseline will not be very helpful due to the highly imbalanced data. Instead, we use the rule-based model as a baseline (see Section 3.2.4.

In order to evaluate the Seq2Seq model, we rely on the BLEU score as the main metric for evaluation. To prepare the raw test data texts, we remove all disfluent words that were marked by the annotators and create a *gold* fluent version. As a secondary measure, we also calculate the BLEU score for the token classification models. To do this, we use the predictions of each token classification model to create fluent versions of the raw transcripts. We then calculate the BLEU scores between the predicted fluent texts and the *gold* target.

4.2 Results

4.2.1 Token Classification

When the precision of the disfluent class is high, the model rarely classifies a word as disfluent when it is not. If the model removes a word even though it is actually fluent, it can reduce the readability of the transcript rather than improve it.

Table 4.2: Full precision, recall, F1-score results on the human-annotated test set. The best performances on the disfluent class are highlighted in bold.

	Fluent			Disfluent		
Model	**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
Rule-based	0.89	1.00	0.94	**0.97**	0.16	0.27
LS1	0.94	0.97	0.95	0.74	0.55	0.63
LS2	0.93	0.98	0.96	0.79	0.51	0.62
LS3	0.94	0.97	0.96	0.76	0.58	**0.66**
LS4	0.94	0.98	0.96	0.77	0.54	0.64
LS5	0.94	0.97	0.96	0.74	0.58	0.65
CGN	0.92	0.97	0.94	0.66	0.41	0.51
LARD	0.89	0.99	0.94	0.69	0.17	0.28
Combo	0.94	0.96	0.95	0.69	**0.60**	0.64

Table 4.2 Shows the results of precision, recall, and F1-score for the disfluent class. The rule-based baseline was designed to ensure high precision by avoiding any risks, resulting in a low recall score and many missed disfluencies. The main goal for the other models is to increase the recall while limiting the loss in precision.

The LS datasets all show comparable performance, as shown by the similar F1-scores. Models with the highest precision score low on recall and vice versa. No LS model outperforms another LS model on both precision and recall.

We can observe that the models with higher precision (*LS3*, *LS4*, *LS2*) have lower recall, while models with lower precision (*LS1*, *LS5*) have a higher recall. This suggests

a trade-off between precision and recall: models that are stricter about labeling tokens as disfluent (higher precision) tend to miss more actual disfluencies (lower recall), while models that are more liberal about labeling tokens as disfluent (lower precision) manage to catch more of them (higher recall). The F1 score, which combines precision and recall, is highest for *LS3* and *LS4*. These models seem to strike the best balance between precision and recall.

The datasets *LS3*, *LS4*, and *LS5* allow a higher replacement ratio. This affects the total amount of samples available for training and the amount of noise in longer sentences (see Figure 3.10. The respective models show higher precision but lower recall for the disfluent class than *LS1* and *LS2*, suggesting that allowing some replacements can help improve precision but at the cost of a lower recall. This is likely due to the dataset containing more examples of disfluencies in noisy texts, causing the model to perform better on noisy inputs.

LS1, *LS3*, and *LS4* have a lower deletion ratio. They outperform *LS2* in recall and *LS5* in precision. However, they also show similar differences between each other. Suggesting that limiting the deletion ratio has little effect on the model's performance. Only *LS2* and *LS5* allow sentence-final disfluencies, the effect of which can also be seen in the average placement of the disfluencies in the sentence (see Figure 3.11). Again the performance difference is large between these two models and also between the models that do not allow sentence-final disfluencies, suggesting this heuristic may also have little effect. The same pattern is found when looking at the class balance heuristic. *LS1* and *LS4* balance the number of sentences with and without disfluencies. They have higher F1 scores than *LS2* and *LS5* (which don't balance the classes), suggesting that class balance can positively affect performance.

Overall, the replacement ratio heuristic seems to affect precision and recall in opposite directions, making it a trade-off that needs to be carefully considered. Beyond this heuristic, none of the differences in heuristics can be inspected in isolation, making it hard to truly predict the effect of the various heuristics. Each model has more than one heuristics variation, causing each difference in performance possibly caused by either difference.

The CGN model loses in terms of both precision and recall to all *LS* models. But significantly outperforms the rule-based baseline in terms of recall. The *LARD* model does score a comparable precision score to the *LS* models. However, its recall only barely outperforms the rule-based baseline. We evaluated the *LARD* model on a separate validation set created following the same disfluency generation method as the training data. The model scores an F1-score on the fluent and disfluent classes of 0.99 and 0.96, respectively. Indicating the problem does not lie in the training setup.

The potential of combining the different datasets is shown by the high recall of the disfluent class by the *combo* model. It outperforms all other models in terms of recall and does so without sacrificing much precision. The model was created with the hope it would be able to aggregate the patterns in the various datasets and increasing its ability to generalize.

Table 4.3 shows the results of all the models on the CGN test set. It is clear that the CGN model outperforms all others by a large margin. As opposed to the Amberdata transcripts, the CGN transcripts were cleaned by humans and therefore contain no ASR errors. Any errors in those transcripts are made by the speakers only. The difference between the two datasets can explain why the models trained on noisier data, the LS models, perform worse. The LARD model still performs badly in terms

Table 4.3: Full precision, recall, F1-score results on the CGN test set. The best performances on the disfluent class are highlighted in bold.

	Fluent			Disfluent		
Model	**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
Rule-based	0.91	1.00	0.96	**0.97**	0.35	0.51
LS1	0.95	0.95	0.95	0.63	0.61	0.62
LS2	0.93	0.98	0.96	0.79	0.51	0.62
LS3	0.94	0.96	0.95	0.67	0.57	0.61
LS4	0.93	0.97	0.95	0.69	0.51	0.59
LS5	0.95	0.94	0.95	0.61	0.63	0.62
CGN	0.96	0.99	0.98	0.93	**0.73**	**0.82**
LARD	0.90	0.99	0.94	0.70	0.22	0.34
Combo	0.96	0.98	0.97	0.86	0.69	0.77

of recall. However, its precision now outperforms the LS models. The LARD data also consists of clean data making it more similar to the CGN data.

The patterns between the different LS models are comparable. The difference in noise levels within the LS datasets has little influence on the performance of the resulting models. The LS4 model does show better precision at the cost of some recall. The LS5 model is the only model that scores a higher recall than precision. That model also scores the highest recall when validating on the human-annotated test set. The increased precision scores on LS3 and LS4 do not follow the prediction that the LS models perform less due to more noise in the training data. These models contain more noise than LS1 and LS2. However, the sentence-final disfluency heuristic can also be an influence, as the main difference between LS4 and LS5 is that LS5 does allow for sentence-final disfluency.

Due to the rule-based baseline model, we can see the difference between the CGN and the human-annotated data. The rule-based baseline only removed repeated spans of 1 or 2 words (excluding 'je' and 'dat'). Since the rule-based baseline has a higher recall when applying it to the CGN test set, it shows that the CGN data contains more 'simple' repetitions. However, since the other models do not perform better on the CGN data, it shows the annotations do not align very well.

Even more pronounced than when we evaluated on the *Amber-test* set, we can see the benefit of combining all data in the performance of the *combo* model. It outperforms all models except the *CGN* model. This shows it does not lose much knowledge specific to the CGN dataset while also learning the intricacies of the Amberdata texts.

4.2.2 Sequence-to-Sequence

In order to allow comparison between the token classification models and the Seq2Seq model, we also calculated the BLEU scores for the token classification models. We measure the BLEU score between the sentences without disfluencies by removing the disfluent words as labeled by either the human annotators or the model.

Table 4.4 shows the BLEU scores. The results show that the Seq2Seq model had lower BLEU scores compared to the token classification models. This could be due to the fact that the Seq2Seq model is capable of making more extensive corrections, including grammatical and stylistic changes, which may not be captured by the BLEU metric.

Table 4.4: The (sacre)BLEU scores for the S2S model and the token classification models. The first score shows the standard BLEU score where the text is tokenized by the tokenizer embedded in the sacreBLEU approach. The '-cleaned' score is where we stripped all punctuation and lower-cased the text before calculating the BLEU score. The scores are calculated with the human-annotated and CGN data.

	Human Annotated		CGN	
Model	**BLEU**	**BLEU** *-cleaned*	**BLEU**	**BLEU** *-cleaned*
Seq2Seq	66.69	74.43	66.42	72.87
LS1	86.56	85.94	90.08	90.04
LS2	86.48	85.86	89.84	89.65
LS3	87.13	86.46	88.91	89.94
LS4	86.67	86.04	88.63	89.55
LS5	87.17	86.37	90.79	90.19
CGN	84.85	84.08	95.01	94.75
LARD	80.06	80.21	83.15	83.32
Combo	87.17	86.35	93.64	93.38

When punctuation and capitalization were removed from the texts before calculating the BLEU score, the Seq2Seq model's score improved significantly, suggesting that the model may be making valuable corrections that are not reflected in the standard BLEU score.

However, even with the improved score, the Seq2Seq model still performed worse than the token classification models. This could be due to several reasons. Token classification models are designed to identify and classify individual tokens in a sequence. In the context of disfluency removal, these models can be trained to identify disfluent tokens and remove or replace them. This makes them inherently well-suited to the task of disfluency removal, as they can operate at the level of individual words or phrases, which is often where disfluencies occur. On the other hand, the Seq2Seq model is designed to generate new sequences based on the input sequence. While this allows it to make more extensive corrections, including grammatical and stylistic changes, it may also make it less precise in identifying and removing disfluencies. The Seq2Seq model's approach of generating new text might lead to overcorrection or unnecessary changes, which could negatively impact the fluency and coherence of the output.

Unlike the token classification models, the Seq2Seq model's hyperparameters were not tuned, which could have limited its performance. For example, adjusting the learning rate, batch size, or the number of layers in the model could potentially improve its ability to learn from the training data and generate a more accurate output. Additionally, the choice of a loss function and optimization algorithm could also impact the model's performance. Therefore, conducting a systematic exploration of the hyperparameter space could potentially lead to significant performance gains for the Seq2Seq model.

4.3 Summary

In this chapter, we describe the detailed steps taken in the data preparation, fine-tuning, and model evaluation phases and the packages used for these procedures. Then, we delve into the specifics of the fine-tuning process, highlighting the parameters used for each model.

We also show the results of the token classification models and the sequence-to-sequence model using the evaluation metrics explained in Chapter 4.1. The LS token classification models all perform similarly. They catch approximately 50% of all labeled disfluencies and are able to accurately label them approximately 75% of the time. The CGN model performs a little less well across the board. The LARD method scores the lowest in terms of recall but still achieves competitive precision. The best recall is achieved by the Combo model, which combines LS2, CGN, and LARD. Its precision scores competitively with the LS models.

The Seq2Seq model cannot be evaluated in the same way. Instead, we use the BLEU score to estimate the model's performance. Using the same test data as for the token classification models, we measure the BLEU score between the generated text and the target text (stripped of disfluencies). Since the Seq2Seq model is capable of adjusting more than just the disfluencies, the BLEU score might judge the performance too harshly. By stripping the punctuation and capitalization, we loosen the requirements for attaining higher BLEU scores. In both methods, the token classification models achieve a higher BLEU score than the Seq2Seq model.

However, the metrics given in this chapter only give a partial insight into the true models' performance. Especially the performance of the Seq2Seq model cannot be accurately estimated using the BLEU score. In the next chapter, we will analyze the performance of the various models in more depth to better understand the models' strengths and weaknesses.

Chapter 5

Error Analysis

In this chapter, we delve deeper into the performance of various models. We analyze the patterns evident in their predictions and outputs to identify their strengths and weaknesses. Additionally, we highlight the behavior of the top-performing token classification model and the Seq2Seq model by providing specific instances from the human-annotated test data where they either excelled or failed.

5.1 Token Classification

In the chapter discussing results, we show that the LS models outperform the other token classification models in terms of precision by correctly identifying disfluencies about 75% of the time. We also show that approximately 50% of the disfluent words were identified by the models. Thus, while the models do mistake some fluent words for disfluent, they mostly miss many disfluencies.

5.1.1 Word frequency in errors

Since the *LS2* model outperforms all other models on *Amber-test* when looking at the precision of the disfluent class, the analysis of the token classification models is thus focused on the *LS2* model.

Figure 5.1 shows the *False Negatives* and *False Positives* of the predictions of the *LS2* model on *Amber-test*. Many of the most frequently missed disfluencies are common Dutch words that can be involved in a variety of speech contexts, not necessarily just disfluencies, making them challenging to classify correctly. However, they are words that are often removed for sentence-initial positions. Starting sentences with `dus` (*so*) or `en` (*and*) is often avoided in written text.

(8) Examples of sentences starting with `ja`. The examples start with the original sentence where bolded words are those the human annotators deemed disfluent. The second line is the sentence without the disfluencies predicted by the model. the Ø-symbol refers to words that are deleted by the model. The last sentence is the translation of the original text.

a. *Ja,* ***dat zou ik,*** *dan zouden we moeten nakijken.*
Ø Ø Ø Ø dan zouden we moeten nakijken.

'yes, that I should, then we should check'

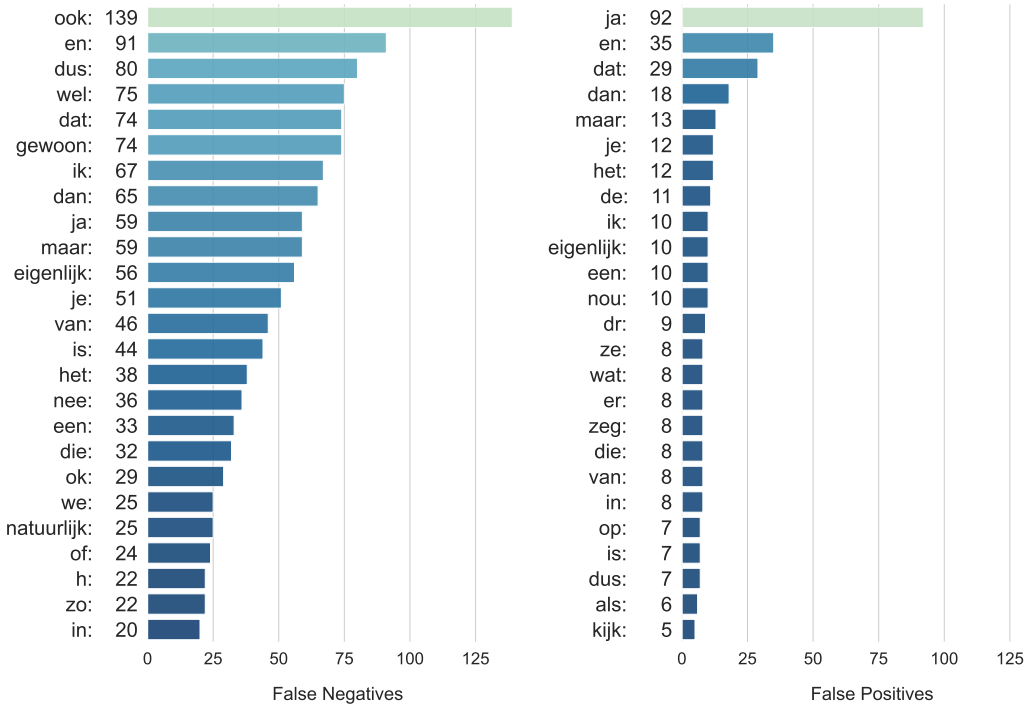

Figure 5.1: The 25 most frequent false negatives (left) and false positives (right) as predicted by the *LS2* model on *Amber-test*

b. ***Ja, oké,*** *en dat is ook een vriendin van je.*
Ja, oké, en dat is ook een vriendin van je.

'yes, okay, and that is also a friend of yours'

c. *Ja, maar dat betekent.*
Ja, maar dat betekent.

'yes, but that means.'

In all examples shown in (8), one can argue the need for `ja` is not possible to determine with certainty. However, in (8a), the annotator deemed it fluent, and in (8b), disfluent. The model's prediction is reversed, which shows that a lack of context makes it hard to label such examples consistently.

Some more typical disfluencies are also missed at times. Words such as `eigenlijk` (*actually*) and `natuurlijk` (*of course*) are often used as fillers and should be easier for the model to catch.

(9) Examples of sentences with `natuurlijk`.

a. *Wat natuurlijk ook gebeurt en wat ik zelf wel heel fijn vind*
Wat natuurlijk ook gebeurt en wat ik zelf wel heel fijn vind

'what happens as well of course and what I personally really like'

b. *En dan moet ik er **natuurlijk** een lopend verhaal van gaan maken*
En dan moet ik er natuurlijk een lopend verhaal van gaan maken

'And then I have to of course make a fluent story of it'

In example (9), we can see that the word is not consistently annotated. In the first example, `natuurlijk` is deemed fluent by the annotator, but it is removed in the second example. The added value of the word can be considered the same in each sentence, showing the subjectivity of the task. Inconsistencies in keeping or removing such words can affect the model's evaluation and training. Maintaining consistent behavior among the many different transcribers who worked on the *clean-read* versions can be challenging. The inconsistencies between the transcribers are likely carried forward into the training data.

Since this model scored a relatively high precision, it is expected that the number of false positives to be much lower than the number of false negatives. The most frequent false positives once again include words that are not distinct disfluencies. Here, the model appears to struggle with short, common words, possibly because of their frequency of usage in a wide array of contexts. It's likely that these words can often appear in disfluent sentences, but they also commonly appear in fluent sentences, hence causing the model to generalize and mark them as disfluent more often than it should.

Certain words appear as both frequent false negatives and false positives, such as `ja` (*yes*), `en` (*and*), and `dat` (*that*). This indicates that the model has difficulty classifying these words correctly in various contexts. The words often appear at the start of a sentence. The annotators did not have contextual information, so they could only judge whether the words were fluent on the basis of the sentence itself. This could lead to inconsistent labeling of sentence-initial disfluencies.

Another overlap is the word `eigenlijk` (*actually*). In total, the word occurs 78 times in the test data. It is marked correctly only 12 times. It's most often incorrectly marked as fluent.

(10) Examples of sentences with `eigenlijk`.

a. *want **dan** Dat is **eigenlijk** heel erg oneerlijk*
want Ø Ø is eigenlijk heel erg oneerlijk

'because that is actually very unfair'

b. *Eigenlijk niet, **zeg maar** de beursgenoteerde bedrijven in Europa.*
Eigenlijk niet, Ø Ø de beursgenoteerde bedrijven in Europa.

'Actually no, let's say the listed companies in Europe.'

In (10), the first example is quite short, and it is hard to be sure whether the word `eigenlijk` is needed semantically. The annotator deemed it unnecessary, while the model opted to keep it. In the second sentence is a little easier to see what pragmatic meaning the word adds, and both the model and the human annotator agree. A slightly longer sentence gives more context to which we can estimate whether a word is disfluent or not. It seems the model is very sensitive to that.

A deep semantic understanding or utilizing more contextual information is needed for the model to determine when these words are part of a disfluent phrase and when

they are not. Since the model's foundation is an LLM, it can access such contextual information. However, since the input sentences are often quite short, the model might lack the knowledge on a case-by-case level. Using longer sentences for the model to label during training and inference could increase the model's performance.

Another reason for the erred predictions could be the labeling method of the training data. The Levenshtein distance-based approach can potentially introduce noise into the labeling process because not all word deletions would correspond to a disfluency. As explained in Section 3.2.1, the method automatically labels each word in a source text based on its presence in the target text. As a result, a word labeled as *delete* might simply be a part of a paraphrased span in the target sentence. Even though the word does not appear in the target text, it is not necessarily disfluent in the source.

The LARD data does not have such errors. As such, we can expect the LARD model to perform better. However, it performed much worse, especially in terms of recall. This recall is improved when combining the three datasets in the *Combo* model.

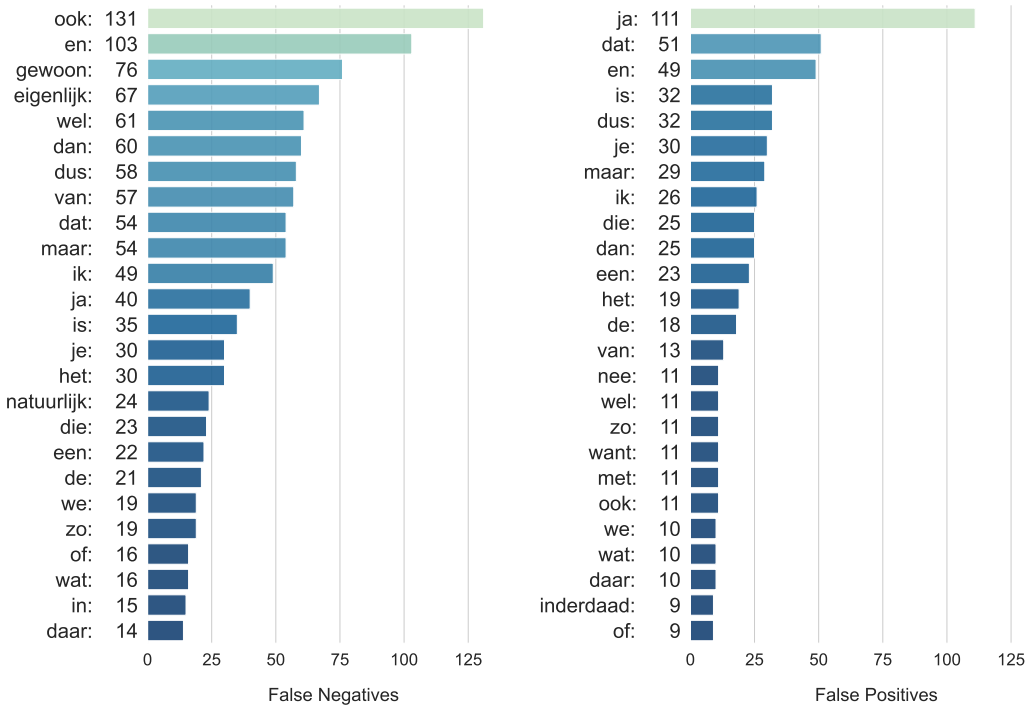

Figure 5.2: The 25 most frequent false negatives (left) and false positives (right) as predicted by the *Combo* model on *Amber-test*.

Figure 5.2 shows the false negatives and positives generated by the *Combo* model. Although the most common errors remain unchanged, the word `eigenlijk` (*actually*) is now correctly marked as disfluent in only three instances, whereas previously, it was marked incorrectly more often. On the other hand, the model is more likely to identify words like `dat` (*that*), `is` (*is*), and `dus` (*so*) as disfluent, as indicated by the lower false negatives and higher false positives compared to Figure 5.1. This could be because these words are more frequently used as disfluent elements in the LARD and CGN data, which the model learned to recognize.

5.1.2 Performance across input length

Training the model on longer inputs and using longer texts as input could improve the model's performance as it will increase access to contextual information. We tested this hypothesis by looking at the performance of the *LS2* model across two sentence-length groups. One group contains all sentences below the mean (16.57 words), and the other all sentences equal to and above the mean length.

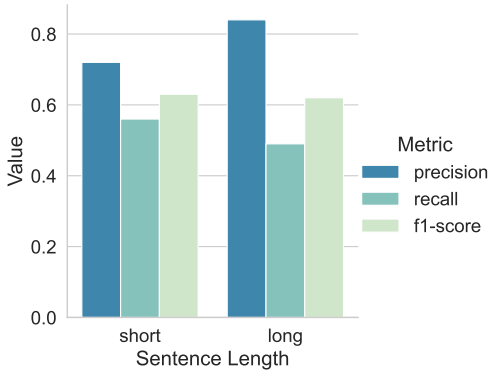

Figure 5.3: The precision, recall, and F1-score performance of the *LS2* model on the test set split in longer and shorter than the mean sentence lengths.

Figure 5.3 shows that the precision on longer sentences is much higher. Even though it comes at a cost in the recall, it is clear the model is able to more precisely judge words as disfluent. The increased length allows for more contextual information to be present on which the model can base its decision.

The recall might have decreased since longer sentences are more likely to contain repairs and false starts than span multiple words. Since the model is not trained on such lengths a lot, it might not be able to find such disfluencies consistently.

5.1.3 Performance across Noise Levels

In Section 3.1.1, we explain how the test data was collected in a stratified manner to represent the various levels of noise found in ASR transcripts. A very *noisy* text is heavily edited by the transcriber in order to clean it up.

Figure 5.4 shows the performance of the *LS2* model when separating the various noise levels. Aside from the *AA* level, the performance for all metrics decreases as we increase the amount of noise. Sentences with higher noise levels are expected to have more ASR errors and, thus, less clear context.

Even though the trend makes it seem like the performance of the model increases as the input improves, the performance on *AA* is not better than *A*. The sentences in *AA* are taken from already cleaned by annotators who had the task of doing a clean verbatim transcript. This means these texts contain no ASR errors but do contain disfluencies.

It is possible that there is little difference between the texts produced by the two noise levels or that the model has reached its maximum performance. Another potential reason is that the training data and test data may not align well. If the model was not trained on clean data, it might not be able to take advantage of the lack of noise. A similar misalignment can be observed with the CGN data. Tables 4.2 and 4.3 clearly

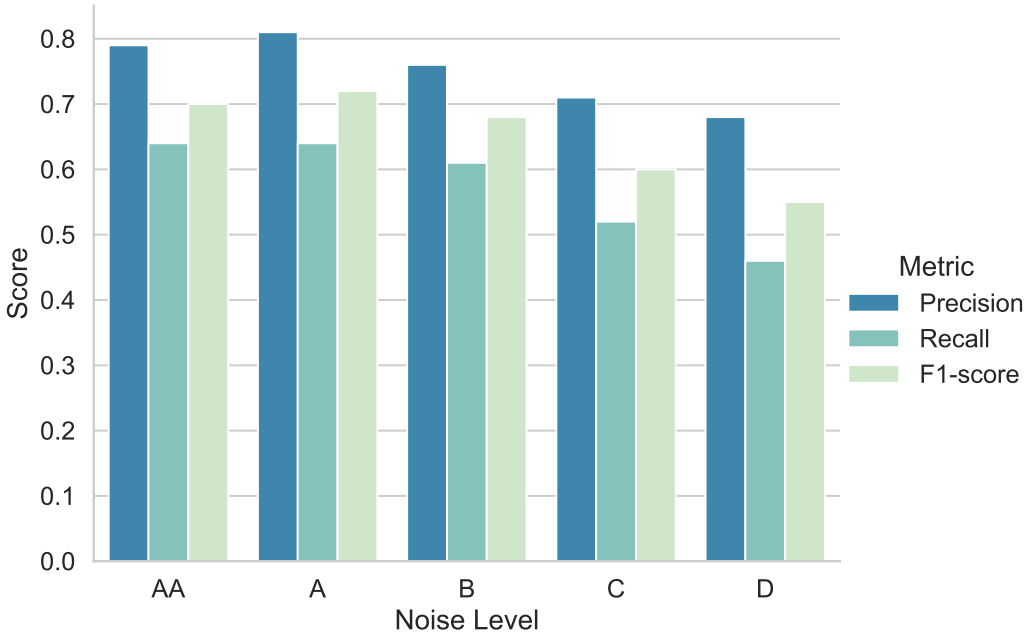

Figure 5.4: The precision, recall, and F1-score performance of the *LS2* model on the *Amber-test* split into the various Noise Levels. From *AA* to *D*, each level is progressively noisy.

demonstrate that models trained on one dataset do not perform as well when tested on another dataset. None of the LS models outperform the *CGN* model when tested on CGN data, and the *CGN* model does not outperform the LS models when tested on *Amber-test*.

5.1.4 Summary

Overall it is clear that the performance of token classification models is highly influenced by an alignment between training and inference data regarding the level of noise in the texts. Furthermore, the most common mistakes made by the models are filler words that can appear as both fluent and disfluent words. Due to inconsistent handling of sentence-initial disfluencies and some fillers such as `ook`, `eigenlijk`, and `gewoon`, it is hard to use the precision, recall, and F1-score metrics as well-defined estimates of model performance.

5.2 Sequence-to-Sequence

In this section, the Seq2Seq model's performance is analyzed. We discuss the representation of the BLEU scores briefly, after which we discuss specific strengths and weaknesses of the model. We analyze the hallucinatory tendencies and the influence of noise and input length on the novel generations made by the model.

5.2.1 General Performance Overview

In Section 4.2.2, the BLEU scores for the Seq2Seq model are shown. However, BLEU scores are limited and do not always give a good insight into model performance. Consider the following example:

(11) *Yesterday a man walked to the store and said I want some bread.*
Yesterday, a man walked to the store and said: "I want some bread."

BLEU: 64.87

In this example, the sentences are almost identical except for the differences in punctuation. The targets were created by removing words from the input text that were deemed disfluent by human annotators. Since the original text is a raw ASR transcript, it does not include punctuation except periods, question marks, and exclamation marks. If we were to remove punctuation and casing from both the target and generated text, the example would score a perfect 100. Furthermore, the annotators were instructed not to correct grammar. Therefore, if the Seq2Seq model does correct grammar, the generated text may deviate further from the target and lower its score.

An automatic score like BLEU can provide a rough estimate, but the best way to evaluate the model's performance is through manual evaluation. To start off, we highlight some of the model's strengths. We hypothesized that the model could correct grammar and remove disfluencies, and this has been observed in many of the model's generated outputs, as demonstrated by the following examples.

(12) Seq2Seq examples - Fixing grammar and style elements. The first line shows the original text. The line below that preceded by the '→' is the text generated by the *Seq2Seq* model. The final line shows the English translation of the generated text.

a. "we hebben alles weghalen omdat we natuurlijk iets niet meer werkt."
→we hebben alles weggehaald omdat er natuurlijk iets niet meer werkt.
"We removed everything because nothing was working of course."

b. "Dus zegt u, we hebben geen echt o, nee, we hebben wel zelf natuurlijk besproken hoe we nog zouden aanpakken, Hebben..."
→Dus zegt u: "We hebben wel zelf natuurlijk besproken hoe we nog zouden aanpakken." We hebben...
"So you say: 'we obviously discussed how we would still approach.' We have..."

It is important to note that ASR models are not always accurate and can sometimes misinterpret words. This can be caused by various factors such as low volume, background noise, accents, and code-switching. However, there are instances where the model can also correct mistakes made by the ASR. The Seq2Seq model has shown instances where it corrects mistakes made by the ASR.

(13) Seq2Seq examples - Correcting ASR errors.

a. "Levels, dus dat team leren en ouder of de-box denken."
→Levels, dus dat teams leren en out-of-the-box denken

"Levels, so that teams learn to think out-of-the-box."

b. "Maar ja, ik denk sowieso nu met courant na ja, weet je wel iets anders, "
→Maar ja, ik denk sowieso nu met corona, weet je wel, iets anders.
"But yeah, I think definitely now with corona, you know, something else"

Example (13a) demonstrates that the Dutch words `ouder of de-box` (*parent or the-box*) were correctly edited to `out-of-the-box`. This correction aligns with the context and indicates that the model can consider abstract information. In the next example, the model changed `courant na ja` (*newspaper after yeah*) to `corona`, referring to COVID-19. The remainder of the sentence (not included for readability) confirms that this is the correct interpretation. While this correction showcases the model's promising capability, it also implies that the model can anticipate words that are not present in the input. This raises the possibility of the model hallucinating.

For example (13b), it is clear that the model is not effectively removing disfluencies. It appears that the model's performance when it comes to removing disfluencies is inconsistent. The model does well on most repairs and false starts but struggles with interjections.

(14) Seq2Seq examples - Inconsistent disfluency removal. Bolded words are disfluencies correctly removed by the model. Underlined words are disfluencies that are not removed.

a. "**ja**, ik **denk dat ik** denk dat we wel wat processen hebben die wat stroperig verlopen."
→Ik denk dat we wel wat processen hebben die wat stroperig verlopen.
"I think we do have some processes that are moving a bit sluggishly."

b. "Maar nou, dat is gewoon niet de sfeer."
→Maar nou, dat is gewoon niet de sfeer.
"But well, that is just not the mood."

As we can see, the bold words in example (14a) are part of a repair and are correctly removed by the model. However, all underlined words are considered interjections and are not removed.

These interjections usually fall into the category of discourse markers or particles. Discourse particles can function as indicators of rhetorical relationships within a conversation or as markers of the connection between statements. They also serve to depict the speaker's attitude towards the dialogue or the person they are conversing with (Hogeweg and van Gerrevink, 2015). Since such words often bring nuance to a text, transcribers are unlikely to remove such words. This would result in the model having few such training examples and will not learn to remove them.

5.2.2 Analysis of Hallucinations

One of the biggest risks with generative models is that they are prone to hallucination. These models predict the next word in a sentence based on its likelihood using the context from previous words. In some cases, the generated word is possibly unrelated to the input the model was provided or even to the text the model is currently generating.

For this analysis, we focus on intrinsic hallucinations. We refer to all words in the generated content that is not present in the source text as *additions*. Note that not all additions are necessarily hallucinations. Approximately a fourth of the model's generations using *Amber-val* contain *additions*.

To investigate this risk, we highlight generated words that are not present in the input. Additionally, we inspect words that are present in the input yet are placed at a different position in the sentence. We create three categories for these *additions*:

- Function words: Words needed for grammatical or structural relationships
- Transformations: Words whose lemma is present in the input
- Novelties: Completely novel content words

As opposed to content words, function words do not carry semantic meaning directly. They form the grammatical and relational structure in a sentence and can specify the attitude or mood of the speaker (Klammer, 2007). *Additions* of this kind are less problematic than others. *Transformations* are words that appear in the input in a different form. Think about various verb conjugations and noun modifications like case, number, and gender. The final category is words that in no way are present in the input nor can be considered function words. These are the most detrimental for reliable and factual generations. If such words are generated often, it shows the model is generating information capable of changing the original meaning.

The reordered words are identified by applying the Levenshtein labeling method to the generated text using the input text as a reference. Any word that is present in the input and gets the label *replace* or *delete* can be seen as a reordered word. Note that reordered words also encompass duplications.

Figure 5.5 shows the 25 most common *additions* and reordered words in the generation of the Seq2Seq model when applied to the test set. Most frequent *additions* and almost all reordered words are function words. As such words do not affect the semantics in a major way, it is likely that the model's generations stay semantically close to the input. For reordered words to mainly consist of function words is logical as the reordering likely points to grammatical corrections. The most frequent content word found in the reordered words and the second most frequent hallucination is `we` (*we*). If the input text consists of multiple constituents, subsequent constituents seem to often miss the pronoun. The Seq2Seq model often inserts them, as shown in Example (15a). This insertion accounts for nearly all instances of `we` as reordered words.

(15) Seq2Seq examples - Inserting `we` in secondary constituents

a. "**We** gaan hiermee door, of **we** moeten ermee stoppen, of een handige kant op."
→**We** gaan hiermee door, of **we** moeten ermee stoppen, of **we** gaan de handige kant op.
"We either continue with this, or we have to stop, or we go the convenient way."

b. "Ja, maar nooit gebruikt ook, ken, maar **we** hebben *(dat)* nog nooit gedaaan."
→Ja, maar nooit gebruikt ook, ken ik. Maar dat hebben **we** nog nooit gedaan.
"Yes, but never used either, I know. But we've never done that."

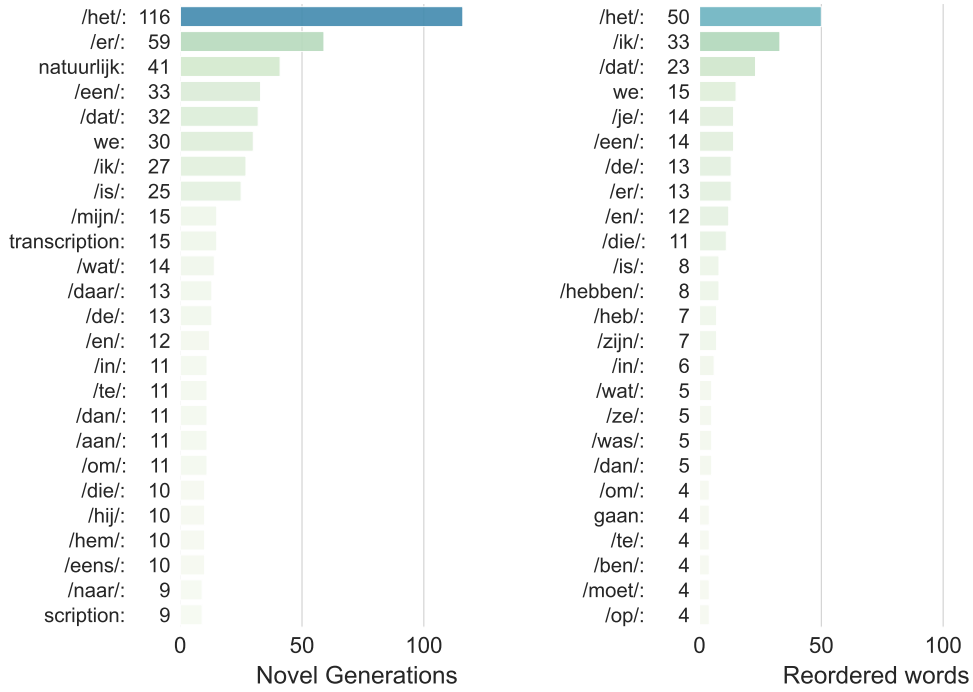

Figure 5.5: The most frequent words that are hallucinated or reordered. Words surrounded by `/` are Function words, and plain words are *novelties*

Example (15b) shows us a different reason. In Dutch, the placement of a pronoun is dependent on whether it is part of a main or subordinate clause. This *second-verb-order* (V2) phenomenon means you can start the sentence with a subject or an object, but regardless of which you start with, the finite verb will come next (Zwart, 2017). V2 is the norm in main clauses, while in subordinate clauses, the verb usually goes to the end. The example shows that the model adds a sentence boundary which changes the clause into a main clause and moves the subject accordingly.

Since function words can often be accepted as *additions*, we can temporarily ignore them when assessing the model's performance. Figure 5.6 shows the most frequent *additions* and reordered words that are not considered function words.

Aside from `we`, each individual frequently reordered word occurs a limited number of times, showing there is no structural change concerning specific words. However, the frequent *additions* show multiple content words that are frequently added. Starting with the most frequent word, `natuurlijk` (*of course*). One cause is that the short version of the word `tuurlijk` is consistently changed into the full word. However, this only constitutes three instances. One instance is where the word was attached to an adjacent word in the input text, which the model separates correctly. The remaining 37 instances show a troubling hallucination.

(16) Seq2Seq examples - Adding `natuurlijk`

a. "Dat is ook bent."
→Natuurlijk. Dat is ook het punt.
"that is also are" →Of course. That is also the point.

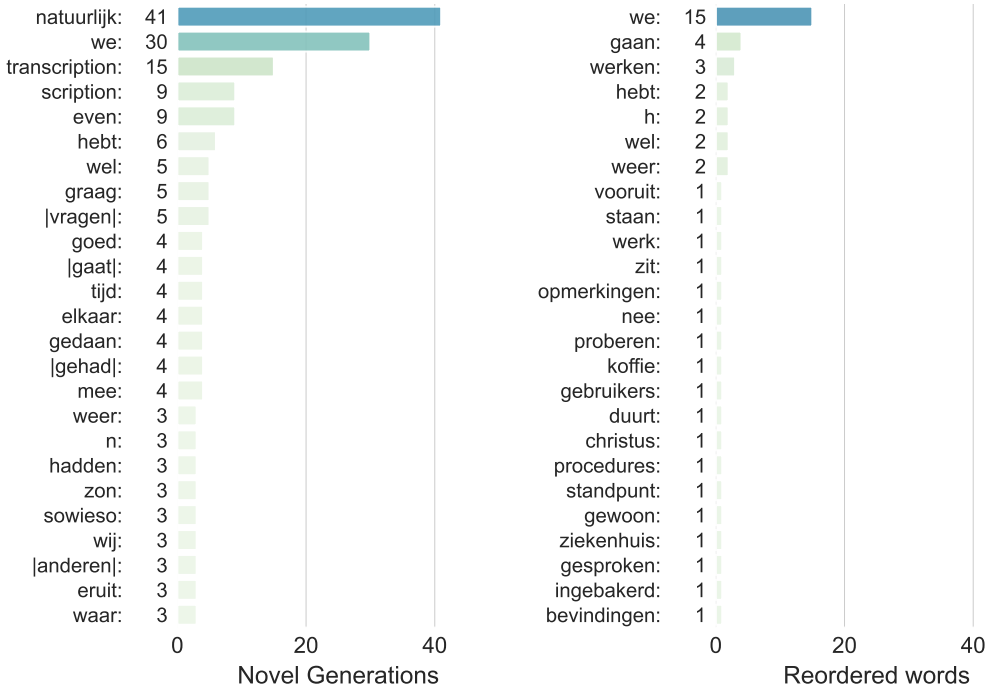

Figure 5.6: The most frequent content words that are hallucinated or reordered. Words surrounded by '|' are *transformations*, and plain words are *novelties*

b. "Maar allemaal kan."
→Natuurlijk. maar allemaal kan.
"But everything can" →Of course. But everything can

c. "Dat er inderdaad."
→Natuurlijk.
"You can do everything well" →Of course. You can do everything well–

In example (16a) the `natuurlijk` is inserted in a way that is semantically fitting. Even though the word is not in the input, it is believable that a speaker would have pronounced it. In example (16b), there is no way to judge whether the addition is acceptable. In the final example, the model replaces everything in the input for `naturlijk`. The frequency of these errors shows the model consistently hallucinates the word. This could be because the training data includes many examples where it is present only in the edited text, causing the model to learn to always include it.

Another frequently hallucinated word is `transcription`. The English word is part of the prefix *Clean Transcription:* , and the model is inconsistent in the removal of the prefix. Together with `scription`, the word always appears at the start of a newly generated text for a total of 21 times.

Both the addition of `natuurlijk` and the retaining of `transcription` (and `scription`) appear only in short texts. Since the model is not often trained on short texts, it is likely that it is not capable of handling such inputs very well.

Other slightly frequent errors are `even` (*briefly*) and `hebt` (*you have, singular*). The first is a consistent correction of `effe`, which is the spoken language rendition of `even`.

The second only occurs when the input misses the verb to form a correct sentence. Many of the other frequent *additions* follow similar patterns and usually fix grammar errors in the input.

5.2.3 Length and *Additions*

As seen before, the input length seems to affect the likelihood of the model's inclination to hallucinate. However, when ignoring the two specific *additions* mentioned above, we see that *additions* appear more in longer inputs than in shorter ones.

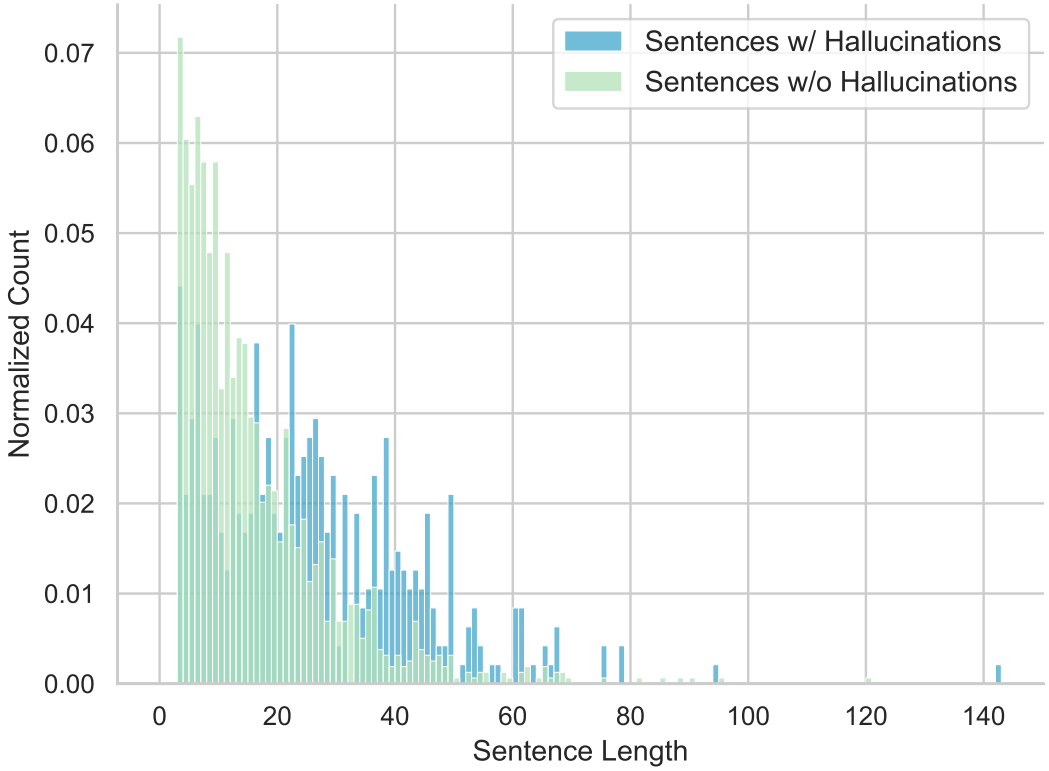

Figure 5.7: The normalized count of sentences per length for either with or without *novelties*.

While the model is able to access more contextual information in longer inputs, it hallucinates more as well. This could imply that the model has difficulties with maintaining contextual understanding over long sequences. Increasing a model's ability to extract information from larger context windows can increase its performance (Guo et al., 2021). However, this is unlikely the cause of the increased hallucination. Figure 5.7 shows that the largest difference between non-hallucinated and hallucinated sentences is between a length of 20 and 60 words, which is well within the context window of the T5-base model (Raffel et al., 2020). Instead, it is more probable that longer sequences simply allow for more instances where the model can hallucinate. Longer input texts tend to have a greater total number of grammatical errors or ASR errors. This gives the model more chances to correct them and, in doing so, hallucinate.

5.2.4 Problematic *Additions*

While the analysis so far has shown that most frequent *additions* are not structural errors, we cannot conclude the model does not hallucinate in a problematic manner. In total, the model hallucinates 361 different words we call *novelties*. If we exclude the structural errors mentioned before, we are still left with approximately 250 hallucinated words. The examples below show some of the problematic *additions* generated by the model.

(17) Seq2Seq examples - Strong *additions*

a. "En procedures en en die de levens van veel mensen negatief beheersen."
→En procedures die de **levensvaardigheden** van veel mensen negatief beheersen.
*"And procedures and which negatively influence the **lives** of many people."*
→life skills

b. "Ja, nee, ik had van kist gehoord."
→Ja, nee, ik had van Kessler gehoord.
*"Yeah, no, I've heard of **chest**" →**Kessler***

c. "Om iets wat daar net vorige."
→Om iets **te zeggen** wat **ik** daarnet voor **gehoord heb.**
"to something that there just last" →To say something that I heard just before

In Examples (17a) and (17b), we can see *additions* that deviate from the input significantly. Especially Example (17c) generates a lot of text not present in the original. What the examples do have in common is that the inputs are rather noisy, and not a lot of sense can be made from them, especially the last example. In these cases, it would be better if the model did not alter the input by much.

The performance of the model in terms of its inclination to hallucinate is affected by the amount of noise in the input. Figure 5.8 shows the proportion of the generated texts that contained fully novel *additions*, function word *additions*, or no *additions* at all. We see that in noise levels AAA and AA, the model hallucinates the least. The noisier the input, the more it hallucinates. However, the number of hallucinated texts is lower in level C than in B. One hypothesis is that at this level, the texts contain so much noise that the model cannot adequately generate improved versions, and instead, it doesn't alter the inputs at all.

Many of the *additions* expressed in this section are hard to identify. Since many automatic metrics do not take the semantic distance between texts into account, and the ones that do are computationally intensive, it is hard to detect them automatically. The present approach was to take a simple rule-based approach to identify novel generations and then manually go through them to find problematic cases. This is not sustainable in either an academic or production environment.

5.2.5 Disfluency Removal Analysis

Up until now, we have focused on the potential issues with the model's behavior, which may give the impression that it is not capable of producing good results. However, out of the 2000 texts that were corrected by the model in the test set, only 375 of them contained *novelties*, and 411 contained function word *additions*. This means that the remaining 1214 generations did not contain any *additions*, and even without assuming

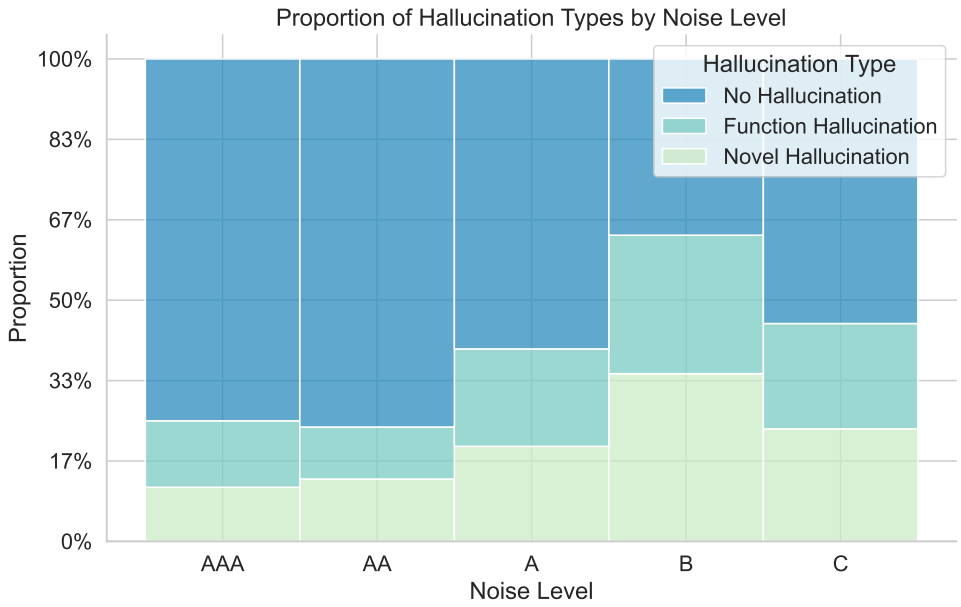

Figure 5.8: Proportion of generations containing different *additions* per noise level in the test data

that most function word *additions* are not harmful to the generated text, we can assume that most generations are valid.

As we mentioned earlier in this chapter, the model can sometimes add stylistic punctuation to the generated text. These changes can significantly enhance the text's readability and help guide the reader in understanding the speaker's intention.

(18) Seq2Seq examples - Changing punctuation.

a. "Als je echt zegt ik wil het niet dan slaan om over."
→Als je echt zegt: "Ik wil het niet", dan slaan we hem over.
"If you really say: 'I don't want it', then we will skip it."

b. "Ook nooit gezegd van: ja, je niet te lang blijven als dat je heb je wel."
→ook nooit gezegd: "Je moet niet te lang blijven." Dat heb je wel.
"Also never said: 'you should stay too long'. Yes you did."

c. "Ik heb hem twee drie weken geleden inderdaad gesproken en hij ja klopt ik heb zo'n ik heb dit dissel ook van hem gehoord."
→Ik heb hem twee, drie weken geleden gesproken en hij– Ja klopt, ik heb dit ook van hem gehoord.
"I spoke to him two, three weeks ago and he– Yeah true, I heard this from him too."

When predicting a phrase as a quote, the model includes colons and quotation marks. It also inserts sentence boundaries and potentially adds interruption marks (`--`) at possible cutoffs. These edits were included in the target texts used to train the model, which it appears to have incorporated into its learning. However, one should be wary about seeing punctuation changes as a definite benefit. Punctuation can change

the meaning of a text as well. As such, like function words, there is a chance that punctuation *additions* can be considered problematic.

(19) Seq2Seq examples - Problematic punctuation *additions*.

a. "Maar ja, ik denk sowieso nu met courant na ja, weet je wel iets anders, "
→Maar ja, ik denk sowieso nu met corona, weet je wel, iets anders.
"But yeah, I think definitely now with corona, you know, something else"

The model is also capable of extracting the core meaningful phrases within the larger text and correctly removing the disfluent elements.

(20) Seq2Seq examples - Removing disfluencies.

a. "**Maar zo belangrijk is het dus, sorry dat ik je heel even** zo belangrijk is**,** het dus **dat eh** dat degene bij wie jij komt vraagt."
→Maar zo belangrijk is het dus dat degene bij wie jij komt vraagt.
"But that's how important it is that the person you come to asks."

b. "**Zelfs voor mijn Maar datdat denk ik** zelfs voor mijn persoonlijke situatie."
→Zelfs voor mijn persoonlijke situatie.
"Even for my personal situation."

c. "**Nou,** ik denk dat ze **het merendeel eh, dat het merendeel** het merendeel van de les wel prima vond, natuurlijk ook omdat ze dan het is anders."
→Nou, ik denk dat ze het merendeel van de les wel prima vond, natuurlijk ook omdat ze dan... Het is anders.
"Well, I think she was fine for most of the lesson, of course because she then also... It's different."

These instances show how the model can eliminate disfluent phrases while preserving the primary message of the sentence. The first two examples display the model's ability to remove extended disfluencies. Both cases involve the repetition of numerous words, separated by an interjection that is also composed of multiple words. Detecting such lengthy interruptions can be challenging since it requires long-term contextual memory. However, since a significant portion of the phrase repeats, detecting it conceptually becomes easier. The third example demonstrates that even when there are multiple repetitions, the model removes the surplus and enhances the readability of other sections in the text.

One highlight is the model's reluctance to clean input-initial or final disfluencies. Some interjections that appeared at the start or end of the input were not removed. The same words were corrected when they appeared more central to the input.

(21) Seq2Seq examples - Removing input-initial/final disfluencies.

a. "**Nou ja,** ik ik zit toch niet zo heel hard in."
→Nou ja, ik zit er toch niet zo heel hard in.
"Well, I'm not in it very hard anyway."

b. "Voordat ik heb ik nog steeds niet, **nou ja**, die hou ik inmiddels."
→Voordat ik het heb, heb ik het nog steeds niet. Die hou ik inmiddels.
"Before I have it, I still don't have it. I'm keeping that one in the meantime."

This discrepancy could be caused by the method of cutting the full transcript into smaller chunks. We cut the texts in the center of any overlapping span. This causes all training data inputs to have no edits made at the start or end of the text. This could teach the model that such edits are not allowed.

5.3 Summary

In this chapter, the behavior of token classification models and Seq2Seq models was analyzed. The performance of the token classification models was assessed using precision, recall, F1-score, and manual analysis. It was found that the models performed well overall but had difficulty in accurately identifying specific interjections. The best-performing model, LS2, also showed that the amount of noise present in the input affected its performance, scoring better on cleaner inputs.

The Seq2Seq model was evaluated based on its ability to correct grammar and style elements, correct ASR errors, and remove disfluencies. The model showed promising results in some areas, such as correcting grammar and style elements and some ASR errors. However, it struggled with removing certain types of disfluencies, particularly interjections. The analysis also revealed that the model was prone to generating *additions*, where it generated words that were unrelated to the input. These *additions* were more likely to occur in longer sentences and noisier inputs. Overall, while the models showed strengths in certain areas, there were also limitations and areas for improvement.

Chapter 6

Discussion

In the preceding chapters, we delved into the challenges and possibilities inherent in the removal of speech disfluencies in automatic speech recognition (ASR) transcripts. With the rising demand for accurate and accessible transcripts driven by increased audio and video media consumption, our exploration sought to address a significant gap in the field. This chapter interprets, discusses, and analyzes the main findings of our research in the context of the research questions posed in the introduction chapter (see Chapter 1.2. First, we summarise the performance of the token classification models and the sequence-to-sequence model in disfluency removal, exploring their respective strengths and weaknesses. Next, we consider the wider implications of our study for the field of ASR transcript and for low-resource languages, underlining the contributions of our work. Finally, we examine the limitations of our study and recommend future directions for further research.

6.1 Token Classification

The token classification models were trained on datasets with two separate labeling setups. The Levenshtein-based models (LS1-5), which were labeled with the Levenshtein labeling setup using various thresholds for insertions, replacements, and deletions, proved the most effective. They correctly identify disfluencies about 75% of the time and can catch approximately 50% of all disfluent words. Despite their effectiveness, the models failed to detect all disfluencies and sometimes mislabel fluent words as disfluent. However, they still performed significantly better than the model trained with the LARD data created using the approach proposed by Passali et al. (Passali et al., 2022), which had good precision scores but struggled to detect many disfluencies. Finally, the model trained with CGN data struck a middle ground between the two approaches.

The differences within the LS models are attributed to varying thresholds and the control measures for noise in the datasets. For instance, in the LS1, LS3, and LS4 models, the disfluencies at the end of a sentence, which could have been due to noise, were not permitted. The thresholds were manipulated to increase the allowed replacement ratio, which increased the model's recall at the cost of some precision. This was apparent when comparing models LS2 and LS5. However, the influence of these heuristic parameters on the models' performance is unclear due to the large variety of values used across the datasets.

In the analysis, the models detected a range of disfluencies, some more accurately

than others. More common disfluencies, such as filler words like `eigenlijk` (*actually*) and `natuurlijk` (*of course*), were often correctly identified. Nonetheless, there were inconsistencies in the annotation, leading to some words being incorrectly labeled.

The models struggled with certain disfluencies due to a range of factors. For instance, they struggled with common words like `ja` (*yes*), `en` (*and*), `dat` (*that*), and `dan` (*then*) which often appear at the start of sentences. The labeling of these sentence-initial disfluencies was inconsistent due to the lack of contextual information. Additionally, some words were incorrectly marked as fluent or disfluent due to their high frequency of usage in various contexts, which caused the model to overgeneralize.

Another challenge came from the labeling method of the training data. The Levenshtein distance-based approach could introduce noise into the labeling process as not all word deletions correspond to a disfluency. This confusion could lead to false positives and false negatives in the model's predictions.

6.2 Sequence-to-Sequence (Seq2Seq)

The Seq2Seq model scored a BLEU score of 66.69 on the human-annotated test set. Cleaning the generated text and target text of capitalization and punctuation increases this score to 74.43. While this step is not common in other research, we deemed it valuable to highlight the lack of representation of the BLEU score for this task. The cleaning process shows how sensitive the score is to minute differences. Since the Seq2Seq model can affect word order and stylistic markers, it can change the text without affecting its meaning. Paraphrased sentences would be judged as worse even though they would potentially be better if we only view the goal of disfluency removal.

While the Seq2Seq model effectively corrects grammatical errors and rectifies errors introduced by the ASR model, it struggles with removing specific types of disfluencies, particularly interjections. This could be caused by lacking training data where such interjections are not consistently removed in the *clean-read* transcript. The edited versions in the data are a collection of edits by a large team of professional transcribers. While the transcribers follow protocols for their work, they are viewed more like guidelines and offer some freedom. Thus, it comes down to each transcriber to judge the need for the interjections. This could lead to inconsistent removal causing the model to be unable to learn the pattern.

One key issue with Seq2Seq models is their susceptibility to *additions* - the generation of words or phrases that do not appear in the input. These *additions* have been categorized into function words, transformations, and novelties. Function words and transformations do not generally impact the semantics of the output significantly, but novelties introduce completely new content words that can alter the meaning of the text. The model showed a high level of *additions* but a much lower generation of novelties. This suggests the semantics of the generated text remain close to the original.

The frequency of *additions* in the Seq2Seq model tends to increase with the length of the input. Rather than suggesting that the model struggles with maintaining contextual understanding over longer sequences, it likely reflects the higher probability of the model hallucinating, given the increased opportunity with more words.

In addition to sequence length, the level of noise in the input text significantly influences the model's tendency to hallucinate, with noisier input leading to more *additions*. This effect disappears at high noise levels, where the hallucination reduces relative to a lower level of noise. Possibly, the model cannot generate improved versions of the

text since the inputs are somewhat nonsensical, causing the model to leave the inputs unaltered.

Despite these challenges, the Seq2Seq model exhibits noteworthy strengths. It shows proficiency in correcting grammar and improving stylistic elements, which enhance readability and clarity. This model's capability to extract core phrases from larger texts and eliminate disfluent elements further underscores its utility in real-world applications.

6.2.1 Implications of the findings

This research offers a novel approach to using automatically labeled and unlabeled data for the removal of disfluencies in ASR transcripts. The token classification approach and the Seq2Seq models investigated could be adopted and further improved by other researchers and technology developers to increase the readability and comprehension of ASR outputs. The approaches are especially useful in low-resource languages where there is little to no labeled data. However, one would still require raw and *clean-read* transcripts in order to fulfill the automatic labeling using the Levenshtein approach.

Both approaches to the task of removing disfluency were capable of generating text with improved readability. The investigation into automatic data labeling techniques provides a way to bypass the expensive manual annotation process. The unsupervised methods studied can be used to create datasets for training token classification models. It demonstrates the viability of automatically labeled data for effective NLP model training. However, we also show not all automatic labeling techniques are equally viable. The Levenshtein labeling approach profited from the similarity between the training data and the inference data. However, the required data (text pairs of raw and cleaned transcripts) is rare and hard to acquire. Even though the LARD method created by Pasalli et al. (Passali et al., 2022), performed subpar on the human-annotated test set, the method has the benefit of easily creating data from widely available cleanly written language.

It is also important to consider that disfluencies in speech can carry significant meaning and nuance. Their removal during the transcript process can potentially alter the tone and intent of the original speech. This can result in a cleaned-up text that is harsher or less nuanced than the original spoken words.

The bias in this context stems from the transcript process when annotators remove these disfluencies. The perception and interpretation of these disfluencies can vary greatly among individual transcribers. Some might not catch the nuance intended by the speaker or may not prioritize maintaining it, while others might. Having a large number of transcribers could potentially average out these individual differences. However, the transcription guidelines that the transcribers are required to follow can contain their own bias, causing potential overlap in individual transcribers.

This work highlights the need for appropriate evaluation metrics for Seq2Seq models. The identified discrepancy between the BLEU score and the real-world applicability of the models implies the need for further research on suitable evaluation methodologies for Seq2Seq tasks. The insights generated regarding the tendency of Seq2Seq models to generate *additions* could guide further research to develop robust metrics to evaluate hallucination. Specifically for disfluency removal, but for other generative tasks as well, hallucinations are the main risk preventing the models from being securely used in practice.

6.3 Future Work

Automatic labeling techniques The LARD method can be improved a lot which would very likely improve its applicability and performance. One such improvement can be found in the use of LLMs. Large Language models have been used in lexical simplification to add a synonym into texts to replace a more complex word (Qiang et al., 2020; Arefyev et al., 2022; Seneviratne et al., 2022). A similar approach could be used to create repairs and no longer rely on lexical databases that might lack the needed coverage of low-resource languages. Since this method does not require both raw and *clean-read* transcripts, only clean language data, it is easy to create datasets in low-resource languages. This makes it very promising, and future research should focus on its expansion.

All disfluencies generated in the LARD approach can also be changed to fit linguistic patterns of disfluency more closely. Not all words in a sentence are equally likely to be repaired or repeated. Neither do interjections appear at random points in a sentence. Constraining the candidates for the application of disfluency to certain part-of-speech (POS) tags can increase the authenticity of the resulting disfluent text.

Furthermore, Passali et al. (2022) aimed to balance the number of samples for each type of disfluency. In doing so, they limit each text to contain only a single disfluent element. Many of the transcripts found in our data often contained a blend of disfluencies in a single sentence. One sentence can contain a repetition, followed by a repair which might all be part of a false start. The layered nature of disfluencies is not fully captured in the LARD method. Expanding the generation could improve the authenticity and applicability of datasets created with the method.

The other approach to automatically creating training data using the Levenshtein labeling method could also be improved. First, to extract the labels, we applied the labeling method to the entire transcript at once. We then split the text into sentences. The resulting labels did not always make sense, as the surrounding text was no longer present. Meaning that sentences could contain labels that no longer fit the text. To solve this issue, we applied heuristics to throw out instances where the text and labels were likely no longer aligned. Instead, we could have used the semantically aligned text-pair extraction used in the Seq2Seq dataset creation. Once the aligned texts were extracted, we could apply the Levenshtein labeling method. This would allow for less data being discarded and possibly less noisy labels.

Secondly, the Levenshtein labeling method is based on the edit distance of the same name. In the distance calculation, each edit (replacement, insertion, and deletion) is viewed as equal. However, for the task of disfluency removal, we might want to place emphasis on the deletion of words rather than replacing or inserting them. Altering the algorithm to favor deletions over other edits could nudge it to better label disfluent elements.

Monolingual Parallel Data The approach to cutting transcripts into smaller chunks resulted in texts that never had an altered word at the beginning or end. As observed in the error analysis section (Section 5), the model is unlikely to change the start or end of the input. Future applications should address this by either randomizing the cutting point within the overlapping sets or always cutting at the end of the set to allow for changes in input-initial words in the *clean-read* transcript.

Contextual Information Models seem to struggle with sentence-initial disfluencies due to a lack of contextual information. The training and testing data was mainly made up of short texts, sometimes only containing a few words. Such texts are very hard to process, even for a human, as was visible in the annotation study. The annotators did not agree often. They reported that the lack of contextual understanding made it difficult for them to accurately decide if a word was disfluent. It can be expected that the same reason is causing the model to have difficulty with the detection as well. Increasing the length of the texts for training and testing data is likely to improve the accuracy of a model's predictions. This can be done either by following the approach used in the Seq2Seq dataset creation or by joining multiple sentences from the token classification dataset.

Improved Base Models The model underlying the Seq2Seq model can be improved upon. The T5-base used to fine-tune the Seq2Seq model is only pre-trained on English data. Subsequent research should explore the effect of using the multilingual version on both performance and minimum required data for fine-tuning. While a large fine-tuning dataset was used to compensate for the lack of Dutch knowledge present in the T5-base, this causes computational overhead as it increases training time.

Furthermore, the experiments using GPT4 show that newer LLM models have a high capacity for zero-shot and few-shot learning of a novel task. Future research could explore the fine-tuning of similar open-source models, such as openLLama-7b and Falcon Geng and Liu (2023); Almazrouei et al..

Even Larger LLMs Recently many models have become available that make previous LLMs such as BERT, RoBERTa, and T5 less worthy of their 'Large' moniker. With the introduction of these multi-billion parameter models (as opposed to the millions present in the older LLMs), several benefits can be found. The models boast much larger context sizes and high zero-shot performance. In such zero-shot settings, the content of a prompt significantly influences the performance of the models Wang et al. (2023). Future research could explore the potential of a zero-shot approach or even a few-shot approach to disfluency removal by experimenting with various prompts. While computationally extremely heavy, some of the models can be fine-tuned, potentially increasing the performance even further.

Lack of Gold-standard Evaluation The experiments in this thesis hinge significantly on the proprietary in-house data provided by Amberscript, which in turn reduces the potential for reproduction and validation of the results in other settings or with different datasets. This reliance creates a significant limitation, restricting the comparability of our findings to other approaches.

As a temporary solution, we implemented the CGN test data to allow for an openly accessible evaluation set. However, this data was not labeled specifically for speech disfluency, and the resulting labels are but an approximation. Furthermore, the large difference in model performance between the two datasets shows the dissimilarity between the disfluency labels. To enhance the comparability of research in this field, future studies could focus on the creation of standardized datasets. These datasets could serve as a *gold standard* for evaluating different methods of disfluency removal. These sets would not only enable consistency across various research efforts but also encourage a more thorough evaluation of different approaches. It is preferable for these

datasets to have distinct labels for each type of disfluency, akin to the Switchboard corpus (Godfrey et al., 1992).

It's noteworthy that our work and the performance of our model are closely tied to the quality of the input data. A better ASR, one that makes fewer recognition errors and is capable of achieving a better verbatim transcript, will provide cleaner input data, simplifying the downstream task of disfluency removal. Any dataset created with the goal of being a central point for comparisons could quickly become redundant as its transcripts are not representative of the present performance of ASR systems.

Furthermore, it's important to acknowledge that not all ASR systems produce verbatim data. Some are trained on audio paired with clean transcripts, which might lead them to eliminate disfluencies during the transcript process. This variation introduces another layer of complexity in evaluation downstream models as the input transcripts are too variable.

Evaluation Metrics We mentioned that the BLEU score does not give a fully accurate measure of the model performance. Future research could employ alternative metrics to reinforce the validity of the quantitative evaluation. Recall-oriented metrics such as ROUGE (Lin, 2004) are more often used in summarization tasks as they are sensitive to deviations from the input. Therefore they possibly better represent the faithfulness of the generation to the input. Furthermore, future research would benefit from moving away from overlap metrics and rather using neural-based learned metrics. Freitag et al. (2022) Show that such metrics better correlate with human judgment across multiple tasks.

6.4 Summary

This chapter discusses the findings of the token classification and Seq2Seq models. The token classification models demonstrated a significant ability to identify disfluencies. Despite their effectiveness, they were not flawless, sometimes failing to detect all disfluencies and occasionally mislabeling fluent words as disfluent. The Seq2Seq model was effective in correcting grammatical errors and rectifying errors introduced by the ASR model. It had difficulty recognizing interjections and exhibited a high level of *additions*. However, we hypothesize the *additions* do not affect the semantic value of the generated text.

The findings of this research offer a novel approach to using automatically labeled and unlabeled data for the removal of disfluencies in ASR transcripts. These approaches are especially beneficial in low-resource languages where there is little to no labeled data. However, the research also highlights the potential risk of altering the tone and intent of the original speech during the transcript process due to the removal of disfluencies.

The chapter suggests several areas for future work. These include improving automatic labeling techniques and improving base models. The chapter also underscores the need for the creation of standardized datasets to enhance the comparability of research in this field. It also acknowledges the limitations of relying on proprietary in-house data, which reduces the potential for reproduction and validation of the results in other settings or with different datasets.

In conclusion, this chapter provides valuable insights into the challenges and possibilities inherent in the removal of speech disfluencies in ASR transcripts and sets the stage for future research in this area.

Chapter 7

Conclusion

This thesis has presented a deep dive into the exploration of data labeling techniques and their applicability in disfluency removal within automatic speech recognition (ASR) pipelines. It delved into the strengths and limitations of token classification and sequence-to-sequence (Seq2Seq) models, marking a noteworthy contribution to disfluency removal in downstream ASR applications.

Addressing the first research question, "Can automatic labeling techniques be reliably used to create datasets to train effective transformer-based token classification models?", the results yield a promising conclusion. The study confirmed the applicability of automatic labeling techniques for training transformer-based token classification models. The models trained with the automatically labeled data were capable of accurately identifying disfluencies. Particularly using the Levenshtein-based models, they are able to recognize disfluencies accurately around 75% of the time, with a recall of approximately 50%. However, the method proposed by Passali et al. (2022), although showcasing reasonable precision, proved less effective in recall. Despite the complexities of the labeling process and the challenges in detecting certain disfluencies, the robustness of the automatic labeling techniques solidifies their potential for future work in this area.

The second part of the first research question focused on the effectiveness of artificially generated disfluencies and automatically labeled data using the Levenshtein approach as training datasets for token classification models. The lower performance found in LARD-based token classification models aligns with the initial hypothesis and is likely due to the realistic representation of conversational language in the Levenshtein labeled data.

The second research question asked, "Can sequence-to-sequence models be reliably trained using automatically aligned fluent-disfluent data to create effective disfluency correction models?". The Seq2Seq model demonstrated its potential, despite some challenges. Although it did not outperform the token classification models when assessed via the BLEU score, it illustrated key strengths in correcting grammatical errors, enhancing stylistic elements, and enhancing readability and clarity. It also showed promise in removing single and multi-word repetitions and repairs. However, it struggled to remove interjections. Furthermore, the Seq2Seq model's tendency towards hallucinations, especially with increasing input length and noise, requires further exploration. Thus, while the hypothesis that the Seq2Seq model would effectively remove disfluencies was partially met, its performance in terms of hallucination and interjections diverged from expectations.

In conclusion, this thesis emphasizes the viability of both automatic labeling techniques for token classification models and the use of Seq2Seq models in disfluency removal. Future work can focus on refining the labeling methods, providing more contextual information for training, and researching the hallucination issue in Seq2Seq models. The findings of this study, while illuminating the potential of these methods, also emphasize the complexity of disfluency removal tasks, inviting further research and development in this promising field.

Appendix A

Rule-based Baseline Fillers

The list below shows all filled pauses removed by the rule-based baseline. This list contains all words that are often used by transcribers to denote filled pauses in verbatim transcriptions. This list might be excessive. The Rule-based model is normally only applied to raw transcriptions. The ASR transcribing the audio usually does not have this many variations to denote filled pauses.

- EH
- Eh
- eh
- uh
- Ee
- Eh,
- Eh...
- Ehh
- Ehm
- Ej
- ehm
- Euhm
- Hm
- Hmhm
- Hmm
- Mhmh
- Mm
- Mm-hmm
- Uh
- Uhm
- Um
- Uuh
- ah
- eeeh
- eeh
- eg
- eh
- eh-eh
- ehj
- ehm
- euh
- euhm
- hm
- hm-hm
- hm-mm
- hmhm
- hmm
- hmm-hm
- mhm
- mhmh
- mm
- mmm
- mh
- oeh
- oh
- u
- uch
- uh-
- uh-uh
- uhh
- um
- ùh
- ûh

Appendix B

Validation and Test set - Noise Levels

Table B.1: We calculated the Levenshtein labels following the process in Section 3.2.1. We set thresholds for the labels: insertion, deletion, and replacements to create the various noise levels. The level 'AA' is created by only using sentences that were cleaned into 'verbatim' by transcribers. These cleaned-up texts are corrected for all ASR errors but not for disfluencies. Levels A to D follow the heuristics using the Levenshtein labels.

Noise Level	**Insertions**	**Replacements**	**Count**
AA	-	-	200
A	0.00-0.01	0.00-0.01	600
B	0.01-0.10	0.01-0.10	400
C	0.10-0.30	0.10-0.30	400
D	0.30-1.00	-	400

Appendix C

Annotation Guidelines

C.1 Annotation Guidelines - EN

[*Note that the guideline's original language is Dutch. The examples are kept in Dutch as their translation will not carry over the patterns found in Dutch disfluency, making the guidelines rather nonsensical*]

The aim of this task is to improve the transcription of our Dutch Automatic Speech Recognition (ASR) system by removing words from the text that do not add meaning. Removing these "disfluencies" is a time-consuming task for transcribers, and automating this process should benefit freelancers in the future. To evaluate any changes made to the system, we need examples that an expert has cleaned up well. This is where this task comes in.

The Excel (or Sheets) file contains two columns with identical sentences in each column. There is no audio available, so all judgments must be based on the text alone. Also, only a single sentence is available, so you cannot use contextual information to make a judgment.

Example of the Excel/Sheets file:

Original Sentence	**Sentence to be edited**
Een heel andere vertrekpunt die nou ja denk ik.	Een heel andere vertrekpunt die nou ja denk ik.

Your goal is to remove words from the sentence in the second column to make it as readable as possible. The resulting sentence does not necessarily have to be grammatically correct. Some sentences may already be incorrect or incomprehensible in the original. If no improvements are possible, it is allowed to leave the sentence as it is without any changes.

Example of a cleaned-up sentence:

Original Sentence	**Sentence to be edited**
Een heel andere vertrekpunt die nou ja denk ik.	Een heel andere vertrekpunt denk ik.

The following page contains examples of errors that you may encounter and how to address them. Try to follow them as closely as possible. The examples are not exhaustive, so use your own judgement. But first, here are some general do's and don'ts:

Do:

- Delete words that seem unnecessary (even if the result is ungrammatical)
- Keep sentences as they are in the original if they appear correct
- Keep sentences as they are in the original if they are incomprehensible
- Make as few changes as possible (small changes are preferred over large ones)

Don't:

- **Replace or insert words**
- Rearrange words in the sentence
- Add or remove punctuation marks
- Change the spelling of the words, i.e., lowercase to uppercase or vice versa, or correct spelling

All errors you encounter can be caused either by the speaker or by the ASR system. Below, we categorize different errors based on these causes. Be aware that in practice, they can overlap, making it difficult to identify and/or correct them.

Human errors

The entire task revolves around these human errors. The following sections provide examples of four types of disfluencies.

Filler words

When we speak, we tend to include unnecessary words in our sentences, such as 'uh' or 'um.' These words serve as pauses to organize our thoughts or can add subtle nuances to our expressions. However, in written language, these words can disrupt the flow of the text and are often better left out.

Examples

The words 'zeg maar' (*like*) in example A are filler words and can be removed. Note that the comma after 'maar,'(*but*) is considered the same disfluency. Beware that example B contains multiple words that can be considered disfluent. They can all be removed to make the resulting sentence concise and understandable. Be careful! Sometimes a word can indicate nuance and be important for the sentence. The word 'gewoon'(*just*) in this sentence could also be important.

Example C shows a sentence that does not grammatically end correctly. This is acceptable. You could also remove all the words from that sentence and end with just

	Original sentence	Improved sentence
A	*"Ik ging naar de winkel* ***zeg maar,*** *om appels te kopen"*	*"Ik ging naar de winkel om appels te kopen"*
B.	***"Dus*** *ik zei* ***zo van*** *ga gewoon je kamer opruimen"*	*"Ik zei ga gewoon je kamer opruimen"*
C.	*"Als je nou eens eventjes* ***zeg maar*** *opschieten""*	*"Als je nou eens eventjes opschieten"*
D.	*"Ik dacht* ***hmm*** *dat kan best"*	*"Ik dacht dat kan best"*
E.	*"Ik ga dan* ***ook*** *even naar de winkel"*	*"Ik ga dan ook even naar de winkel"*

'Opschieten' (*hurry up*), but that deviates too much from the original. Again, note the nuance in the word 'eventjes' (*just*).

In example D, we see a slightly different word. 'Hmm' is often used to verbalize our thoughts; if the verbalization is unnecessary, we can see it as disfluent and remove it. Be careful again when the word adds nuance to the sentence.

For all examples, feel free to use your instincts to make these kinds of decisions.

In example E, 'ook' (*also*) is a possible filler word, but we cannot remove it. It is difficult to assess whether 'ook' is a filler word in this sentence. Only remove it if you are very sure. For example, if the sentence also included the word 'gerust' to become: 'Ik ga dan ook gerust even naar de winkel'. In this case, it is more apparent that 'ook' is a filler word.

Repetitions

Another way to give ourselves time to think or formulate our message is by repeating words. This is very common in speech.

Examples

	Original sentence	Improved sentence
A	*"Ik* ***ik ik,*** *wil graag een appel"*	*"Ik wil graag een appel"*
B.	*"Dat denk ik ook en* ***en*** *misschien wel meer"*	*"Dat denk ik ook en misschien wel meer"*

Each word can be repeated one or more times. Make sure to remove all repeated words except for one.

Repairs

Sometimes, we use the wrong sentence or word and realize the mistake. We can correct it by saying what we meant. Both the initial mistake and the additions before the repair are considered disfluent. We always consider the last element as the repair and the first element as the disfluency.

Examples

	Original sentence	Improved sentence
A	*"Ik ging naar de **winkel ik bedoel** supermarkt om appels te kopen"*	*"Ik ging naar de supermarkt om appels te kopen"*
B.	*"Welke type **mensen** professionals hebben we nodig?"*	*"Welke type professionals hebben we nodig?"*
C.	*"Als je opgroeide in een **klein plekje zeg maar** dorpje"*	*"Als je opgroeide in een dorpje"*
D.	*"dat kunnen we gebruiken voor **energiebesparing** besparende maatregelen"*	*"dat kunnen we gebruiken voor besparende maatregelen"*

The most important thing in the above examples is that repairs can be preceded by a filler word or a vocalization of recognizing the mistake. In Example A, the speaker adds 'ik bedoel' (*I mean*) to recognize that they made a mistake and initiate the repair. In Example B, this filler word is not pronounced. Example C shows another possible filler word.

Example D shows that it is common for only the last part of compound words to be modified. We stick to the rule of repairs: "the last element is always the repair, and the first element is the disfluency."

Restart

A restart is essentially similar to a repair, but it can sometimes be harder to recognize. These disfluencies occur when a speaker restarts their sentence because they have changed their mind or want to restructure their message.

Examples

	Original sentence	Improved sentence
A	*"**Ik wilde gister** we gingen naar de dierentuin"*	*"we gingen naar de dierentuin"*
B.	*"**Als je opgroeide in** als je woont in de buurt van een stad"*	*"als je woont in de buurt van een stad"*

Note that each of the above-mentioned human errors can occur simultaneously. For example, a part of a sentence can be repaired, and in that repair, there may be fillers and repetitions. This makes the task rather complex and sometimes messy. Try to do as much as possible to improve the text, of course, by only removing words.

ASR-Specific Errors

Unfortunately, our ASR system cannot recognize every word correctly. In some cases, the system may transcribe a different word instead of the correct one. This means that most sentences in the data contain one or more ASR errors. If these errors fall within the patterns of human disfluencies, then they should be removed. However, do not replace incorrect words with correct ones, even if you know what they should be.

Examples In example A, it is likely said: "Ja, maar wat goed van je!" (*Yes, but how*

	Original sentence	Improved sentence
A	*"Ja, maar goed* wat *je!"*	*"Ja, maar goed wat je!"*
B.	*"Te liggen,* ***ja,*** *wat mij opvalt is dat zij op het vlak naast elkaar hé focusgroepen behandeling nodig."*	*"Te liggen****,*** *wat mij opvalt is dat zij op het vlak naast elkaar hé focusgroepen behandeling nodig."*
C.	*"zeker voor die jongens, dr zijn er ook een aantal* ***design****, heel snel doorgestroomd"*	*"zeker voor die jongens, dr zijn er ook een aantal heel snel doorgestroomd"*
D.	*"Hij verbood op een gegeven moment de verkoop van losse melk zo uit* ***uit*** *elkaar."*	*"Hij verbood op een gegeven moment de verkoop van losse melk zo uit elkaar."*

good of you!). To solve this, we would need to replace the word 'wat' (*what*) with 'van'(*of*), but that is not allowed.

Example B shows a slightly more complex sentence. The beginning is not clear what it means or refers to. It also doesn't seem to add much to the rest of the sentence. However, 'te liggen' (*to lay*) cannot be removed. The part that says 'hé focusgroepen' (*Hey focusgroups!*) could also be an ASR error. But since it's not clear how this sentence is intended, we have to leave it in. However, the word 'ja'(*yes*) seems to be a filler word and can be removed.

Example C shows that, with some phonetic analysis, it is sometimes possible to deduce what may have been said. Instead of the predicted word 'design' (*design*), the intended words might have been: 'die zijn' (*those are*). This fits well in the sentence. However, in that case, it could also be a disfluency, namely a reformulation of 'dr zijn' (*there are*). Be careful with such deletions, be very sure that it is a disfluency. When in doubt, leave the sentence as it is.

In example D, the sentence ends in a way that does not fit with the rest. Despite being nonsensical, we **cannot** remove it! None of the human disfluencies provide a reason to remove the ending. However, in the nonsensical part, a disfluency can be recognized (a repetition), don't forget to check for this.

In conclusion

A final note on the data. We have compiled the data to reflect various real-life automatic transcriptions. This means that there are also rather nonsensical sentences. They may be abruptly cut off or appear to be part of another sentence that is not visible in the text. This was done on purpose, and it is up to your own judgment to see if there is something sensible to extract from it.

Some speakers make a lot of speech errors and repeatedly correct themselves. Checking the text a second time after removing a disfluency can help find more disfluencies.

C.2 Annotation Guidelines - NL

Het doel van deze taak is de transcriptie van ons Nederlandse automatische spraakherkenningssysteem (ASR) te verbeteren door woorden uit de tekst te verwijderen die geen betekenis toevoegen. Het verwijderen van deze "disfluenties" is een tijdrovende taak voor transcribisten, en het automatiseren van dit proces zou freelancers in de toekomst ten goede moeten komen. Om eventuele wijzigingen in het systeem te kunnen beoordelen, hebben we voorbeelden nodig die een expert goed heeft opgeschoond. Hier komt deze taak om de hoek kijken.

Het Excel (of Sheets) bestand bevat twee kolommen met identieke zinnen in elke kolom. Er is geen audio beschikbaar, dus alle oordelen moeten alleen gebaseerd zijn op de tekst. Ook is alleen een enkele zin beschikbaar, dus je kunt geen contextuele informatie gebruiken om een oordeel te vellen.

Voorbeeld van het Excel/Sheets bestand:

Oorspronkelijke zin	**Zin om te verbeteren**
Een heel andere vertrekpunt die nou ja denk ik.	Een heel andere vertrekpunt die nou ja denk ik.

Jouw doel is om woorden uit de zin van de tweede kolom te verwijderen om de zin zo leesbaar mogelijk te maken. De resulterende zin hoeft niet noodzakelijk grammaticaal correct te zijn. Sommige zinnen kunnen in het origineel al onjuist of onbegrijpelijk zijn. Als er geen verbeteringen mogelijk zijn is het toegestaan de zin te laten zoals die is, zonder wijzigingen.

Voorbeeld van een opgeschoonde zin:

Oorspronkelijke zin	**Zin om te verbeteren**
Een heel andere vertrekpunt die nou ja denk ik.	Een heel andere vertrekpunt denk ik.

De volgende pagina bevat voorbeelden van fouten die u kunt tegenkomen en hoe u die kunt aanpakken. Probeer deze zo goed mogelijk te volgen. De voorbeelden zijn niet allesomvattend, dus gebruikt uw eigen inzicht. Maar eerst zijn er enkele algemene do's en don'ts:

Do:

- Woorden schrappen die overbodig lijken (zelfs als het resultaat ongrammaticaal is)
- Zinnen als origineel behouden als ze correct lijken
- Zinnen als origineel behouden als ze onbegrijpelijk zijn
- Wijzig zo weinig mogelijk (kleine wijzigingen hebben de voorkeur boven grote)

Don't:

- **Woorden vervangen of invoegen**
- Woorden in de zin herschikken
- Leestekens toevoegen of verwijderen
- De schrijfwijze van de woorden veranderen, d.w.z. kleine letters in hoofdletters en omgekeerd, of spelling verbeteren

Alle fouten die u tegenkomt kunnen ofwel door de spreker worden veroorzaakt, ofwel door het ASR-systeem. Hieronder categoriseren we verschillende fouten aan de hand van deze oorzaken. Wees u ervan bewust dat ze in de praktijk door elkaar kunnen lopen, en dat het daarom moeilijk kan zijn ze te identificeren en/of te verbeteren.

Menselijke fouten

De hele taak draait om deze menselijke fouten. De volgende secties geven voorbeelden van vier type disfluenties

Vulwoorden

Als we spreken, hebben we de neiging onnodige woorden in onze zinnen op te nemen, zoals 'eh' of 'ehm'. Deze woorden dienen als pauzes om onze gedachten te ordenen of kunnen subtiele nuances aanbrengen in onze uitdrukkingen. In geschreven taal kunnen deze woorden echter de doorstroming van de tekst verstoren en kunnen ze vaak beter worden weggelaten.

Voorbeelden

	Oorspronkelijke zin	Verbeterde zin
A	*"Ik ging naar de winkel* ***zeg maar,*** *om appels te kopen"*	*"Ik ging naar de winkel om appels te kopen"*
B.	***"Dus*** *ik zei* ***zo van*** *ga gewoon je kamer opruimen"*	*"Ik zei ga gewoon je kamer opruimen"*
C.	*"Als je nou eens eventjes* ***zeg maar*** *opschieten""*	*"Als je nou eens eventjes opschieten"*
D.	*"Ik dacht* ***hmm*** *dat kan best"*	*"Ik dacht dat kan best"*
E.	*"Ik ga dan* ***ook****even naar de winkel"*	*"Ik ga dan ook even naar de winkel"*

De woorden 'zeg maar' in voorbeeld A zijn vulwoorden en kunnen worden verwijderd. Let op dat de komma achter 'maar,' wordt gezien als dezelfde disfluentie. Merk op dat voorbeeld B meerdere woorden bevat die als disfluent kunnen worden beschouwd. Ze kunnen allemaal worden verwijderd om de resulterende zin beknopt en begrijpelijk te maken. Pas wel op! Soms kan een woord juist nuance aangeven en mogelijk belangrijk voor de zin. Het woord 'gewoon' in deze zin zou ook belangrijk kunnen zijn.

Voorbeeld C toont een zin die niet grammaticaal correct eindigt. Dit wordt geaccepteerd. Je zou ook alle woorden uit die zin kunnen schrappen en alleen eindigen met: 'Opschieten', maar dat wijkt te veel af van het origineel. Zie ook weer de nuance in het woord 'eventjes'.

In voorbeeld D zien we een iets ander woord. 'Hmm' kan vaak gebruikt worden om onze gedachten te verbaliseren; als de verbalisatie overbodig is, kunnen we het als disfluent zien en verwijderen. Pas weer op voor wanneer het woord nuance toevoegt aan de zin. Voor alle voorbeelden geldt: Gebruik gerust wat instinct om dit soort besluiten te maken.

In voorbeeld E is 'ook' een mogelijk vulwoord maar kunnen we niet verwijderen. Met 'ook' is het lastig is om in te schatten of dit een vulwoord is. Verwijder het alleen als je echt heel zeker bent. Door bijvoorbeeld 'gerust' aan de zin toe te voegen: 'Ik ga dan ook gerust even naar de winkel'. In dit geval is het duidelijker dat 'ook' een vulwoord is.

Herhalingen

Een andere manier om onszelf tijd te geven om na te denken of onze boodschap te formuleren is door woorden te herhalen. Dit is heel gebruikelijk in spraak.

Voorbeelden

	Oorspronkelijke zin	Verbeterde zin
A	*"Ik **ik ik,** wil graag een appel"*	*"Ik wil graag een appel"*
B.	*"Dat denk ik ook en **en** misschien wel meer"*	*"Dat denk ik ook en misschien wel meer"*

Elk woord kan één of meerdere keren herhaald worden. Zorg ervoor dat je alle herhaalde woorden verwijdert, behalve één.

Reparatie

Soms gebruiken we een verkeerde zin of een verkeerd woord en beseffen we de fout. We kunnen het corrigeren door te zeggen wat we bedoeld hadden. Zowel de aanvankelijke fout als de toevoegingen vóór de reparatie worden als disfluent beschouwd. We beschouwen het laatste element altijd als de reparatie, en het eerste element als de disfluentie.

Voorbeelden

	Oorspronkelijke zin	Verbeterde zin
A	*"Ik ging naar de **winkel ik bedoel** supermarkt om appels te kopen"*	*"Ik ging naar de supermarkt om appels te kopen"*
B.	*"Welke type **mensen** professionals hebben we nodig?"*	*"Welke type professionals hebben we nodig?"*
C.	*"Als je opgroeide in een **klein plekje zeg maar** dorpje"*	*"Als je opgroeide in een dorpje"*
D.	*"dat kunnen we gebruiken voor **energiebesparing** besparende maatregelen"*	*"dat kunnen we gebruiken voor besparende maatregelen"*

Het belangrijkste in bovenstaande voorbeelden is dat de reparaties kunnen worden voorafgegaan door een vulwoord of een vocalisatie van het herkennen van de fout. In Voorbeeld A voegt de spreker 'ik bedoel' toe om te herkennen dat ze een fout hebben gemaakt en de reparatie in te leiden. In Voorbeeld B is dit vulwoord niet uitgesproken. Voorbeeld C toont een ander mogelijk vulwoord.

Voorbeeld D laat zien dat het voorkomt dan in samengestelde woorden alleen het laatste deel wordt aangepast. We houden vast aan de vuistregel omtrent reparaties: 'het laatste element is altijd de reparatie, en het eerste element de disfluentie'

Herstart

Een herstart lijkt in wezen op een reparatie, maar is soms moeilijker te herkennen. Deze disfluencies treden op wanneer een spreker opnieuw begint met zijn zin omdat zij van gedachten is veranderd of de boodschap wil herstructureren.

Voorbeelden

	Oorspronkelijke zin	Verbeterde zin
A	*"**Ik wilde gister** we gingen naar de dierentuin"*	*"we gingen naar de dierentuin"*
B.	*"**Als je opgroeide in** als je woont in de buurt van een stad"*	*"als je woont in de buurt van een stad"*

Merk op dat elk van de bovengenoemde menselijke fouten tegelijkertijd kan voorkomen. Zo kan een deel van een zin gerepareerd worden, en in die reparatie zitten vervolgens vulwoorden en herhalingen. Dit maakt de taak nogal complex en soms rommelig. Probeer zoveel mogelijk te doen om de tekst te verbeteren, uiteraard door alleen woorden te verwijderen.

ASR-Specifieke fouten

Helaas kan ons ASR-systeem niet elk woord correct herkennen. In sommige gevallen kan het systeem een ander woord transcriberen in plaats van het juiste woord. Dit betekent dat de meeste zinnen in de data één of meer ASR-fouten bevatten. Als deze fouten binnen de patronen van menselijke disfluenties vallen, dan moeten ze verwijdert worden. Vervang foutieve woorden echter niet door correcte, ook al weet u wat ze zouden moeten zijn.

Voorbeelden

In voorbeeld A wordt waarschijnlijk gezegd: "Ja, maar wat goed van je!". Om dit op te lossen zouden we het woord "wat" moeten vervangen door 'van', en dat mag niet.

Voorbeeld B toont een iets complexere zin. Het begin is niet duidelijk wat het moet betekenen of waarnaar het verwijst. Het lijkt ook niet veel toe te voegen aan de rest van de zin. Toch mag je "te liggen" niet verwijderen. Het gedeelte waar staat 'hé focusgroepen' zou ook een ASR-fout kunnen zijn. Maar omdat het niet duidelijk is hoe deze zin bedoeld is, moeten we het er in laten. Het woord 'ja' lijkt echter een vul woord en mag verwijderd worden.

	Oorspronkelijke zin	Verbeterde zin
A	*"Ja, maar goed* wat *je!"*	*"Ja, maar goed wat je!"*
B.	*"Te liggen,* ***ja,*** *wat mij opvalt is dat zij op het vlak naast elkaar hé focusgroepen behandeling nodig."*	*"Te liggen****,*** *wat mij opvalt is dat zij op het vlak naast elkaar hé focusgroepen behandeling nodig."*
C.	*"zeker voor die jongens, dr zijn er ook een aantal* ***design****, heel snel doorgestroomd"*	*"zeker voor die jongens, dr zijn er ook een aantal heel snel doorgestroomd"*
D.	*"Hij verbood op een gegeven moment de verkoop van losse melk zo uit* ***uit*** *elkaar."*	*"Hij verbood op een gegeven moment de verkoop van losse melk zo uit elkaar."*

Voorbeeld C laat zien dat, met wat fonetisch analyseren, het soms best te herleiden valt wat er mogelijk gezegd is. In plaats van het voorspelde woord 'design' kan heel goed 'die zijn' uitgesproken zijn. Dit past goed in de zin. Echter, in dat geval kan ook nog eens een disfluentie zijn, namelijk een herformulering van 'dr zijn'. Pas wel op met dit soort verwijderingen, wees heel zeker dat het disfluentie is. Bij twijfel, de zin laten zoals het is.

In voorbeeld D eindigt de zin op een manier dat niet past bij de rest. Ondanks dat het onzinnig is, kunnen we het **niet** verwijderen! Geen van de menselijke disfluenties geeft reden om dit te verwijderen. Echter in het onzinnige deel, is wel een disfluentie te herkennen (een herhaling), vergeet dit niet te controleren.

Tot slot

Een laatste opmerking over de data. We hebben de data zo samengesteld dat ze verschillende automatische transcripties uit de praktijk weerspiegelen. Dat betekent dat er ook zinnen zijn die nogal onzinnig zijn. Ze kunnen plotseling worden afgekapt of deel lijken uit te maken van een andere zin die niet zichtbaar is in de tekst. Dit is met opzet gedaan en het is aan uw eigen oordeel om te zien of er iets zinnigs uit te halen valt.

Sommige sprekers maken veel spreek fouten en herstellen keer op keer. De tekst een tweede keer controleren na een disfluencie te hebben verwijderd, kan helpen om meer disfluenties te vinden.

Appendix D

Hyperparameter tuning results

Table D.1: The full results of the hyperparameter tuning of LS2 tested on *Amber-val*.

Learning Rate	Batch Size	Warm-up steps	Fluent			Disfluent		
			Precision	Recall	F1-score	Precision	Recall	F1-score
3e-6	8	500	0.93	0.97	0.95	0.76	0.53	0.62
3e-6	8	1000	0.92	0.97	0.95	0.74	0.49	0.59
3e-6	16	500	0.93	0.97	0.95	0.74	0.53	0.62
3e-6	16	1000	0.93	0.97	0.95	0.75	0.54	0.63
3e-6	32	500	0.93	0.97	0.95	0.73	0.54	0.62
3e-6	32	1000	0.93	0.97	0.95	0.74	0.53	0.62
3e-6	64	500	0.93	0.97	0.95	0.72	**0.55**	0.62
3e-6	64	1000	0.93	0.97	0.95	0.72	**0.55**	0.62
3e-5	8	500	0.93	**0.99**	**0.96**	**0.88**	0.53	**0.66**
3e-5	8	1000	0.92	0.98	0.95	0.82	0.47	0.60
3e-5	16	500	0.92	0.99	0.95	0.84	0.49	0.62
3e-5	16	1000	0.92	0.99	0.95	0.85	0.48	0.62
3e-5	32	500	0.93	0.98	0.95	0.79	0.53	0.64
3e-5	32	1000	0.93	0.98	0.95	0.79	0.52	0.63
3e-5	64	500	0.93	0.97	0.95	0.75	0.54	0.63
3e-5	64	1000	0.93	0.98	0.95	0.77	0.52	0.62

Table D.2: The full results of the hyperparameter tuning of CGN tested on the CGN validation data. Note that no warm-up step tuning was done, as we saw little to no effect on the LS2 tuning and to save computation

Learning Rate	Batch Size	Fluent			Disfluent		
		Precision	Recall	F1-score	Precision	Recall	F1-score
3e-6	8	0.96	0.99	0.98	0.94	0.77	0.85
3e-6	16	0.96	0.99	0.98	0.94	0.76	0.84
3e-6	32	0.96	0.99	0.98	0.93	0.75	0.83
3e-6	64	0.96	0.99	0.97	0.94	0.73	0.82
3e-5	8	0.97	0.99	0.98	**0.95**	0.79	**0.86**
3e-5	16	0.97	0.99	0.98	0.93	0.80	**0.86**
3e-5	32	0.97	0.99	0.98	0.91	**0.82**	**0.86**
3e-5	64	0.97	0.99	0.98	0.93	0.80	**0.86**

Bibliography

E. Almazrouei, A. Cappelli, R. Cojocaru, M. Debbah, E. Goffinet, D. Heslow, J. Launay, Q. Malartic, B. Noune, B. Pannier, and G. Penedo. Falcon-40b: an open large language model with state-of-the-art performance.

N. Arefyev, B. Sheludko, A. Podolskiy, and A. Panchenko. Always keep your target in mind: Studying semantics and improving performance of neural lexical substitution. *arXiv preprint arXiv:2206.11815*, 2022.

J. E. Arnold, C. L. H. Kam, and M. K. Tanenhaus. If you say thee uh you are describing something hard: the on-line attribution of disfluency during reference comprehension. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 33(5):914, 2007.

N. Bach and F. Huang. Noisy bilstm-based models for disfluency detection. In *INTERSPEECH*, pages 4230–4234, 2019.

M. Bachmann. Levenshtein. `https://github.com/maxbachmann/Levenshtein`, 2022.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

G. W. Beattie and B. L. Butterworth. Contextual probability and word frequency as determinants of pauses and errors in spontaneous speech. *Language and speech*, 22 (3):201–211, 1979.

A. Bell, D. Jurafsky, E. Fosler-Lussier, C. Girand, M. Gregory, and D. Gildea. Effects of disfluencies, predictability, and utterance position on word form variation in english conversation. *The Journal of the acoustical society of America*, 113(2):1001–1024, 2003.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

S. Bird, E. Klein, and E. Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

O. Bloodstein, N. B. Ratner, and S. B. Brundage. *A handbook on stuttering.* Plural Publishing, 2021.

P. Boersma and T. Weenink. Praat, 1992. URL `https://praat.org`.

H. Bortfeld, S. D. Leon, J. E. Bloom, M. F. Schober, and S. E. Brennan. Disfluency rates in conversation: Effects of age, relationship, topic, role, and gender. *Language and speech*, 44(2):123–147, 2001.

S. E. Brennan and M. Williams. The feeling of another s knowing: Prosody and filled pauses as cues to listeners about the metacognitive states of speakers. *Journal of memory and language*, 34(3):383–398, 1995.

M. Chen. How long does it actually take to transcribe one hour of audio? 2022. URL `https://www.notta.ai/en/blog/how-long-does-it-take-to-transcribe-1-hour-of-audio`.

V. Cherkassky and F. M. Mulier. *Learning from data: concepts, theory, and methods*. John Wiley & Sons, 2007.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

N. Chomsky. Aspects of the theory of syntax cambridge. *Multilingual Matters: MIT Press*, pages 1–15, 1965.

H. H. Clark and J. E. F. Tree. Using uh and um in spontaneous speaking. *Cognition*, 84(1):73–111, 2002.

H. H. Clark and T. Wasow. Repeating words in spontaneous speech. *Cognitive psychology*, 37(3):201–242, 1998.

Databricks. Mlflow: An open source platform for the machine learning lifecycle. `https://github.com/mlflow/mlflow`, 2018.

J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, et al. Large scale distributed deep networks. *Advances in neural information processing systems*, 25, 2012.

P. Delobelle, T. Winters, and B. Berendt. Robbert: a dutch roberta-based language model. *arXiv preprint arXiv:2001.06286*, 2020.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

European Parliament and Council of the European Union. Directive (EU) 2019/882 of the European Parliament and of the Council of 17 April 2019 on the accessibility requirements for products and services (Text with EEA relevance), 2019. URL `https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:32019L0882`. Accessed: June 15, 2023.

C. Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998. URL `https://mitpress.mit.edu/9780262561167/`.

F. Ferreira and K. G. Bailey. Disfluencies and human language comprehension. *Trends in cognitive sciences*, 8(5):231–237, 2004.

V. S. Ferreira and H. Pashler. Central bottleneck influences on the processing stages of word production. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 28(6):1187, 2002.

K. Filippova. Controlled hallucinations: Learning to generate faithfully from noisy data. *arXiv preprint arXiv:2010.05873*, 2020.

M. Freitag, R. Rei, N. Mathur, C.-k. Lo, C. Stewart, E. Avramidis, T. Kocmi, G. Foster, A. Lavie, and A. F. Martins. Results of wmt22 metrics shared task: Stop using bleu–neural metrics are better and more robust. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 46–68, 2022.

J. Fruehwald. Filled pause choice as a sociolinguistic variable. *University of Pennsylvania Working Papers in Linguistics*, 22(2):6, 2016.

W. A. Gale and K. W. Church. A program for aligning sentences in bilingual corpora. *Comput. Linguist.*, 19(1):75–102, mar 1993. ISSN 0891-2017.

M. W. Gardner and S. Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

X. Geng and H. Liu. Openllama: An open reproduction of llama, May 2023. URL `https://github.com/openlm-research/open_llama`.

J. J. Godfrey, E. C. Holliman, and J. McDaniel. Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, volume 1, pages 517–520. IEEE Computer Society, 1992.

F. Goldman-Eisler. Speech production and the predictability of words in context. *Quarterly Journal of Experimental Psychology*, 10(2):96–106, 1958.

C. Goodwin. Between and within: Alternative sequential treatments of continuers and assessments. *Human studies*, 9(2-3):205–217, 1986.

M. Guo, J. Ainslie, D. Uthus, S. Ontanon, J. Ni, Y.-H. Sung, and Y. Yang. Longt5: Efficient text-to-text transformer for long sequences. *arXiv preprint arXiv:2112.07916*, 2021.

J. He, X. Wang, G. Neubig, and T. Berg-Kirkpatrick. A probabilistic formulation of unsupervised text style transfer. *CoRR*, abs/2002.03912, 2020. URL `https://arxiv.org/abs/2002.03912`.

A. E. Hieke. A content-processing view of hesitation phenomena. *Language and Speech*, 24(2):147–160, 1981.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

H. Hoekstra, M. Moortgat, B. Renmans, M. Schouppe, I. Schuurman, and T. van der Wouden. Cgn syntactische annotatie, 2003.

L. Hogeweg and R. van Gerrevink. The acquisition of the dutch discourse particle wel. *Nederlandse taalkunde*, 20(2):187–213, 2015.

V. M. Holmes. Hesitations and sentence planning. *Language and cognitive processes*, 3 (4):323–361, 1988.

M. Honal and T. Schultz. Correction of disfluencies in spontaneous speech using a noisy-channel approach. In *Interspeech*, 2003.

M. Honnibal and I. Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. Bang, A. Madotto, and P. Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 2022.

J. M. Johnson and T. M. Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):1–54, 2019.

W. Johnson. Measurements of oral reading and speaking rate and disfluency of adult male and female stutterers and nonstutterers. *Journal of Speech & Hearing Disorders. Monograph Supplement*, 1961.

T. P. Klammer. *Analyzing English Grammar, 6/e*. Pearson Education India, 2007.

K. Krippendorff. Estimating the reliability, systematic error and random error of interval data. *Educational and psychological measurement*, 30(1):61–70, 1970.

K. Krippendorff. Content analysis: An introduction to its methodology (2 nd thousand oaks, 2004.

A. Kutuzov. Improving english-russian sentence alignment through pos tagging and damerau-levenshtein distance. In *Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing*, pages 63–68, 2013.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

P. Lendvai, A. v. d. Bosch, and E. Krahmer. Memory-based disfluency chunking. In *ISCA Tutorial and Research Workshop on Disfluency in Spontaneous Speech*, 2003.

W. J. Levelt. Monitoring and self-repair in speech. *Cognition*, 14(1):41–104, 1983.

W. J. Levelt, A. Roelofs, and A. S. Meyer. A theory of lexical access in speech production. *Behavioral and brain sciences*, 22(1):1–38, 1999.

R. J. Lickley. Fluency and disfluency. *The handbook of speech production*, pages 445–474, 2015.

C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

H. Liu, A. Gegov, and M. Cocea. *Rule based systems for big data: a machine learning approach*, volume 13. Springer, 2015.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

G. F. Mahl. Disturbances and silences in the patient's speech in psychotherapy. *The Journal of Abnormal and Social Psychology*, 53(1):1, 1956.

V. Mendelev, T. Raissi, G. Camporese, and M. Giollo. Improved robustness to disfluencies in rnn-transducer based speech recognition. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6878–6882. IEEE, 2021.

H. Moniz, F. Batista, A. I. Mata, and I. Trancoso. Speaking style effects in the production of disfluencies. *Speech communication*, 65:20–35, 2014.

R. C. Moore. Fast and accurate sentence alignment of bilingual corpora. In S. D. Richardson, editor, *Machine Translation: From Research to Real Users*, pages 135–144, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-45820-3.

N. Mustakim, R. Rabu, G. M. Mursalin, E. Hossain, O. Sharif, and M. M. Hoque. Cuet-nlp@ tamilnlp-acl2022: Multi-class textual emotion detection from social media using transformer. In *Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages*, pages 199–206, 2022.

C. H. Nakatani and J. Hirschberg. A corpus-based study of repair cues in spontaneous speech. *The Journal of the Acoustical Society of America*, 95(3):1603–1616, 1994.

N. Oostdijk, W. Goedertier, F. v. Eynde, L. Boves, J.-P. Martens, M. Moortgat, and R. H. Baayen. Experiences from the spoken dutch corpus project. 2002.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

T. Passali, T. Mavropoulos, G. Tsoumakas, G. Meditskos, and S. Vrochidis. Lard: Large-scale artificial disfluency generation. *arXiv preprint arXiv:2201.05041*, 2022.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

M. Postma, E. van Miltenburg, R. Segers, A. Schoen, and P. Vossen. Open Dutch WordNet. In *Proceedings of the 8th Global WordNet Conference (GWC)*, pages 302–310, Bucharest, Romania, 27–30 Jan. 2016. Global Wordnet Association. URL `https://aclanthology.org/2016.gwc-1.43`.

J. Qiang, Y. Li, Y. Zhu, Y. Yuan, and X. Wu. Lexical simplification with pretrained encoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8649–8656, 2020.

J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng. A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, 2016:1–16, 2016.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

N. Reimers. Pretrained models, 2022. URL `https://www.sbert.net/docs/pretrained_models.html`.

N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

S. R. Rochester. The significance of pauses in spontaneous speech. *Journal of Psycholinguistic Research*, 2:51–81, 1973.

J. C. Rocholl, V. Zayats, D. D. Walker, N. B. Murad, A. Schneider, and D. J. Liebling. Disfluency detection with unlabeled data and small bert models. *arXiv preprint arXiv:2104.10769*, 2021.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

A. B. Sai, A. K. Mohankumar, and M. M. Khapra. A survey of evaluation metrics used for nlg systems. *ACM Computing Surveys (CSUR)*, 55(2):1–39, 2022.

N. Saini, D. Trivedi, S. Khare, T. Dhamecha, P. Jyothi, S. Bharadwaj, and P. Bhattacharyya. Disfluency correction using unsupervised and semi-supervised learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3421–3427, Online, Apr. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.299. URL `https://aclanthology.org/2021.eacl-main.299`.

S. Schachter, N. Christenfeld, B. Ravina, and F. Bilous. Speech disfluency and the structure of knowledge. *Journal of personality and social psychology*, 60(3):362, 1991.

I. Schuurman, M. Schouppe, H. Hoekstra, and T. van der Wouden. CGN, an annotated corpus of spoken Dutch. In *Proceedings of 4th International Workshop on Linguistically Interpreted Corpora (LINC-03) at EACL 2003*, 2003. URL `https://aclanthology.org/W03-2414`.

S. Seneviratne, E. Daskalaki, and H. Suominen. Cils at tsar-2022 shared task: Investigating the applicability of lexical substitution methods for lexical simplification. In *Proceedings of the Workshop on Text Simplification, Accessibility, and Readability (TSAR-2022)*, pages 207–212, 2022.

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

P. P. Shinde and S. Shah. A review of machine learning and deep learning applications. In *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, pages 1–6. IEEE, 2018.

E. Shriberg and A. Stolcke. Word predictability after hesitations: A corpus-based study. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, volume 3, pages 1868–1871. IEEE, 1996.

E. E. Shriberg. *Preliminaries to a theory of speech disfluencies*. PhD thesis, Citeseer, 1994.

V. L. Smith and H. H. Clark. On the course of answering questions. *Journal of memory and language*, 32(1):25–38, 1993.

F. Stouten and J.-P. Martens. Benefits of disfluency detection in spontaneous speech recognition. In *COST278 and ISCA Tutorial and Research Workshop (ITRW) on Robustness Issues in Conversational Interaction*, 2004.

P. J. O. Suárez, B. Sagot, and L. Romary. Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures. In *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*. Leibniz-Institut für Deutsche Sprache, 2019.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

P. H. Tannenbaum, F. Williams, and C. S. Hillier. Word predictability in the environments of hesitations. *Journal of verbal learning and verbal behavior*, 4(2):134–140, 1965.

H. Tian, C. Gao, X. Xiao, H. Liu, B. He, H. Wu, H. Wang, and F. Wu. Skep: Sentiment knowledge enhanced pre-training for sentiment analysis. *arXiv preprint arXiv:2005.05635*, 2020.

G. Tottie. Uh and um as sociolinguistic markers in british english. *International Journal of Corpus Linguistics*, 16(2):173–197, 2011.

G. Tottie. On the use of uh and um in american english. *Functions of Language*, 21 (1):6–29, 2014.

J. E. F. Tree. The effects of false starts and repetitions on the processing of subsequent words in spontaneous speech. *Journal of memory and language*, 34(6):709–738, 1995.

F. Van Eynde. Part of speech tagging en lemmatisering van het corpus gesproken nederlands. *KU Leuven*, 2004.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

L. Vladimir. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

G. Walford. *Doing qualitative educational research*. Bloomsbury Publishing, 2001.

J. Wang, E. Shi, S. Yu, Z. Wu, C. Ma, H. Dai, Q. Yang, Y. Kang, J. Wu, H. Hu, et al. Prompt engineering for healthcare: Methodologies and applications. *arXiv preprint arXiv:2304.14670*, 2023.

S. Wang, W. Che, Q. Liu, P. Qin, T. Liu, and W. Y. Wang. Multi-task self-supervised learning for disfluency detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9193–9200, 2020a.

S. Wang, Z. Wang, W. Che, and T. Liu. Combining self-training and self-supervised learning for unsupervised disfluency detection. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1813–1822, Online, Nov. 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.142. URL `https://aclanthology.org/2020.emnlp-main.142`.

S. Wang, Z. Wang, W. Che, and T. Liu. Combining self-training and self-supervised learning for unsupervised disfluency detection. *arXiv preprint arXiv:2010.15360*, 2020c.

Y. Wang, Y. Sun, Z. Ma, L. Gao, Y. Xu, and T. Sun. Application of pre-training models in named entity recognition. In *2020 12th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 1, pages 23–26. IEEE, 2020d.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

S. Wubben, A. Van Den Bosch, and E. Krahmer. Paraphrase generation as monolingual translation: Data and evaluation. In *Proceedings of the 6th International Natural Language Generation Conference*, 2010.

Wyzowl. Video marketing statistics 2023, 2023. URL `https://www.wyzowl.com/video-marketing-statistics/`. Surveyed 528 unique respondents in November 2022.

L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel. mt5: A massively multilingual pre-trained text-to-text transformer, 2021.

D. Yu and L. Deng. Deep learning and its applications to signal and information processing [exploratory dsp]. *IEEE Signal Processing Magazine*, 28(1):145–154, 2010.

C. Zhou, G. Neubig, J. Gu, M. Diab, P. Guzman, L. Zettlemoyer, and M. Ghazvininejad. Detecting hallucinated content in conditional neural sequence generation. *arXiv preprint arXiv:2011.02593*, 2020.

J.-W. Zwart. An argument against the syntactic nature of verb movement. *Order and structure in syntax I: Word order and syntactic structure*, pages 29–47, 2017.