Reduplication vs. Repetition

RnD Report

Submitted in partial fulfillment of requirements for the degree of

Honours

By

Mothika Gayathri Khyathi
Roll No.: 200050077

under the guidance of

Prof. Pushpak Bhattacharyya

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

May 5, 2024

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Acknowledgment

I extend my sincere gratitude to my esteemed guides Prof. Pushpak Bhattacharyya, whose invaluable guidance and direction have been instrumental in shaping my research work. Their expertise and constant support have been invaluable throughout the project, and I am grateful for their mentorship.

Abstract

The importance of Speech Translation, whether in the form of speech-to-speech or speech-to-text, has increased tremendously due to globalization. Automatic Speech Recognition (ASR) and Machine Translation (MT) play crucial roles in cascaded models. However, disfluencies in speech result in disfluencies in ASR transcriptions, which can degrade the performance of downstream NLP activities such as MT since they are trained on well-formed data. In this study, we propose a post-processing approach for ASR transcriptions to improve MT performance. Disfluency correction models can be used for this purpose, but the lack of labeled data in low-resource languages poses a challenge to their development. Additionally, disfluency correction systems may struggle to differentiate between disfluencies like repetition and morphological phenomena like reduplication, which are common in Indic and South Asian languages.

Reduplication and repetition, though similar in form, serve distinct linguistic purposes. Reduplication is a deliberate morphological process used to express grammatical, semantic, or pragmatic nuances, while repetition is often unintentional and indicative of disfluency. This report examines the first large-scale study of **reduplication-repetition** in speech using computational linguistics. We begin by discussing the motivations for this problem and the methods that can be used to address it. We introduced **IndicRedRep**, a new publicly available dataset containing Hindi, Telugu, and Marathi text annotated with reduplication and repetition at the word level. We evaluate transformer-based models for multi-class reduplication and repetition token classification, utilizing the Reparandum-Interregnum-Repair structure to distinguish between the two phenomena. Our models achieve macro F1 scores of up to 85.62% in Hindi, 83.95% in Telugu, and 84.82% in Marathi for reduplication-repetition classification. Our work can aid in the development of disfluency correction systems for low-resource languages, improving Speech Translation performance and downstream NLP tasks.

Contents

Chapter 1

Introduction

1.1 Problem statement

Speech utterances are classified into two types: read speech and conversational speech. Read speech involves speakers reading directly from a written source, while conversational speech is more spontaneous and dynamic. Conversational speech often contains irregularities like disfluencies, which can make the speech less clear and fluent compared to read speech. These disfluencies, such as repetitions or hesitations, are not only common in human conversations but also pose challenges for natural language processing tasks like Machine Translation. However, research on disfluencies in NLP is limited due to a lack of datasets, particularly in local languages like Telugu, Marathi etc. Our focus is on studying one specific type of disfluency: **repetition**. This repetition is often mistaken for **reduplication** in disfluency correction systems. To address this, we aim to create a dataset in Hindi, Telugu and Marathi that includes labels for both reduplication and repetition word-wise, and then train models to accurately distinguish between them in speech utterance transcripts.

1.2 Motivation

- India is a diverse country with a multitude of languages. Although the Constitution recognizes 23 official languages, there are over 700 languages spoken across the nation. Catering to the linguistic diversity is crucial for providing essential services like healthcare, legal aid, and tourism information in various Indian languages. However, educational resources are predominantly available in English, posing a challenge for children who primarily speak their mother tongue or regional language.
- Manual translation is both time-consuming and expensive. Machine Translation (MT) offers a promising solution as it is fast, cost-effective, and versatile across different domains.

Yet, MT technology remains inaccessible to a significant portion of India's population, particularly in rural areas where illiteracy rates are higher. Accessibility to translation technology can be greatly enhanced through Speech-to-Speech Machine Translation systems, leveraging the ease of speech interaction and eliminating the need for text input.

- While current Speech-to-Speech Machine Translation systems excel with fluent speeches, they struggle with spontaneous speeches that often contain irregularities like disfluencies. Automatic Speech Recognition (ASR) accurately transcribes words but may capture these irregularities. Such disfluencies hinder the performance of Machine Translation systems, which are typically trained on well-formed sentences. Detecting and correcting disfluencies is crucial as they pose challenges not only for Machine Translation but also for other Natural Language Processing (NLP) tasks.

- Disfluency detection and correction serve as vital preprocessing steps for NLP, aiming to identify and remove disfluent words. Although disfluency detection has been extensively studied in English, it has received less attention in other languages. This underscores the importance of exploring disfluencies and their repair mechanisms across different linguistic contexts.

- However, distinguishing reduplication from repetition presents its own challenges. Reduplication, a linguistic phenomenon found in many languages, involves repeating all or part of a word to convey nuances in meaning, intensify the original word, or indicate plurals, tense, or aspect. Reduplication can occur at different levels within a word and serves various linguistic functions.

- In this study, we focus on discriminating reduplication from repetition given a speech utterance transcript.

1.3 Challenges

The challenges faced in studying reduplication vs. repetition classification for Indian languages are,

- **Lack of data:** The major challenge in building a system for classifying reduplication from repetition is that no data is available for any Indian languages (Telugu and Marathi) that specifically marks reduplications from repetitions.

- We need to translate Hindi data to Telugu and Marathi. Also collect data from sources, with speakers speaking in a conversational setting.

- The complexity of reduplication lies in its nuances, including the distinction between total and partial reduplication, as well as the exactness or inexactness of repetition. These variations add intricacy to the task of annotating data, particularly when dealing with partial reduplications or repetitions.

1.4 Contributions

- We propose a new task in natural language processing (NLP) that involves distinguishing reduplications and repetitions in a spoken setting.
- Creation of ”IndicRedRep,” a new dataset available to the public that includes over 4.5K Hindi, 1.6K Telugu, and 1.6K Marathi sentences, all annotated with labels for reduplication and repetition. This is the first dataset of its kind to offer token-level annotations for these features in any language.
- Development and implementation of a novel methodology using the Reparandum-Interregnum-Repair (RiR) structure, which has improved the macro F1 score by 2% across all tested languages.
- Empirical evaluation of various classical sequence labeling models and transformer-based models on the new dataset to classify reduplication and repetition accurately.
- Detailed linguistic analysis of the dataset across three languages—Hindi, Telugu, and Marathi—to understand the unique challenges and behaviors of models when dealing with different linguistic contexts.

1.5 Roadmap of Report

Below is an outline of the sequence of topics covered chapterwise in this report:

- ✓ Chapter 1 describes the problem statement as well as the motivation, challenges for the topic. It also gives a list of contributions of the work.
- ✓ Chapter 2 gives a brief history of ASR systems with more focus on deep learning based systems.
- ✓ Chapter 3 gives a brief literature review of Disfluency Correction.
- ✓ Chapter 4 discusses the basics of Disfluency correction as well as the various terminologies associated with this pehonomena.
- ✓ Chapter 5 discusses about reduplication from a linguistic point of view and views it as a morphological operation.
- ✓ Chapter 6 brings to attention the importance of Repetition vs. Reduplication and discusses the phenomena of reduplication and repetition in English, Hindi and other regional languages like Telugu and Marathi.
- ✓ Chapter 7 gives the details of the dataset annotated for reduplication and repetition, as well as perform a detailed data analysis of this dataset.
- ✓ Chapter 8 lists the results from training various models on the reduplication-repetition dataset, and gives the results and performance metrics.
- ✓ Chapter 9 discusses plan for future publication.
- ✓ Chapter 10 concludes the report and gives a list of future work to be done.

Chapter 2

Background

2.1 Automatic Speech Recognition

Humans have always shown a keen interest in speech technologies, particularly in the realm of recognition. Speech recognition serves as a foundational task in Natural Language Processing (NLP) systems and plays a crucial role in downstream applications. For example, a robust speech recognition system is essential for advancing machine translation in speech-to-speech translation scenarios. However, progress in speech recognition was relatively slow until the first half of the 20th century. It wasn't until the 1950s that corporations worldwide began significant investments in recognition technologies, leading to substantial advancements in research and production.

Automatic Speech Recognition (ASR) is the automated process of converting speech signals into text output. While there are over 7000 languages globally, just 23 of these languages cover more than half of the world's population. ASR plays a vital role in bridging communication gaps across different languages. With the emergence of Machine Learning and Deep Learning, ASR has witnessed remarkable improvements in its performance across various languages. Like any other AI application, the availability of data plays a significant role in developing state-of-the-art ASR systems.

Considerable research has been conducted on languages such as English, Spanish, and German, known as High Resource Languages due to the abundance of data and open-source toolkits available for these languages. However, low resource languages like Hindi, Marathi, and Tamil lack high-performance systems due to the scarcity of transcribed data.

We treat the acoustic input signal as $O = o_1, o_2, o_3, o_4, \ldots$ a series of observations and define a sequence of words as the desired output $W = w_1, w_2, w_3, w_4, \ldots$.

We would like to get those sequence of words W from the language L, which maximizes the following condition given the acoustic input O.

$$\hat{W} = \text{argmax}_{W \in L} P(W|O) \tag{2.1}$$

We can use Bayes rule to rewrite this as -

$$\hat{W} = \text{argmax}_{W \in L} \frac{P(O|W)P(W)}{P(O)} \tag{2.2}$$

For every possible sequence of words W, the denominator of equation 2 is the same. Since we are dealing with the argmax operator, we can ignore the denominator and write the final expression as -

$$\hat{W} = \text{argmax}_{W \in L} P(O|W)P(W) \tag{2.3}$$

Each component in a traditional system plays an important role in calculating the above two probabilities. For evaluating an ASR system, Word Error Rate (WER)[1] is a common metric used. The WER is derived from the Levenshtein distance, working at the word level instead of the phoneme level. Word error rate can then be computed as:

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C} \tag{2.4}$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions, C is the number of correct words, N is the number of words in the reference.

2.2 Disfluency Correction

Disfluency correction involves rectifying non-fluent conversational elements such as filler words and pauses present in speech transcripts. In spontaneous everyday conversations, individuals often engage in simultaneous thinking and speaking, leading to the insertion of disfluent words like ”uh,” ”um,” ”ah,” etc. into their speech. These irregularities significantly impact conversational clarity and affect the readability of speech transcripts, particularly when Automatic Speech Recognition (ASR) systems convert spoken utterances into text. If transcriptions lack fluency, it can pose challenges for downstream processing of speech data. Consider the following example sentence with filler words highlighted in bold:

Well, this is this is **you know** a good plan.

In this instance, the core message of the sentence is conveyed through the phrase ”This is a good plan,” while the remaining words serve as filler and should be eliminated by a disfluency correction system. Chapter 4 delves into the theory behind disfluency correction and explores various types of disfluency structures.

2.3 Transformers

Vasawani in (37) proposes the Transformer architecture which makes use of self-attention to get rid of the recurrence. Transformer architecture is more parallelizable and requires less amount of

time than the other models. Main components of the Transformer model are discussed below.

2.3.1 Encoder and Decoder Stack

The Transformer model, known for its effectiveness in various natural language processing tasks, is structured into two fundamental components: the encoding component and the decoding component. Within the encoding section, there exists a sequence of 6 identical encoder layers by default. These encoder layers, crucial for transforming input data into a meaningful representation, are not only identical but also possess unshared weights. Each of these layers is intricately composed of two distinct sub-layers: the Multi-Head Self Attention Layer, which facilitates the model's ability to focus on different parts of the input sequence simultaneously, and the Feed-Forward Neural Network Layer, responsible for processing the attention-weighted representations further.

On the other hand, the decoding component of the Transformer model mirrors its encoding counterpart in structure. It also houses 6 identical decoder layers, each with unshared weights and specifically designed to decode the encoded representations generated by the encoder layers. However, unlike the encoding layers, the decoder layers feature an additional sub-layer known as the Multi-Head Encoder-Decoder Attention Layer. This layer enables the decoder to attend to different parts of the input sequence during the decoding process, contributing significantly to the model's overall performance in tasks such as machine translation and text generation. Figure 2.1 shows the Transformer model with two encoder layer and two decoder layer (Alammar).

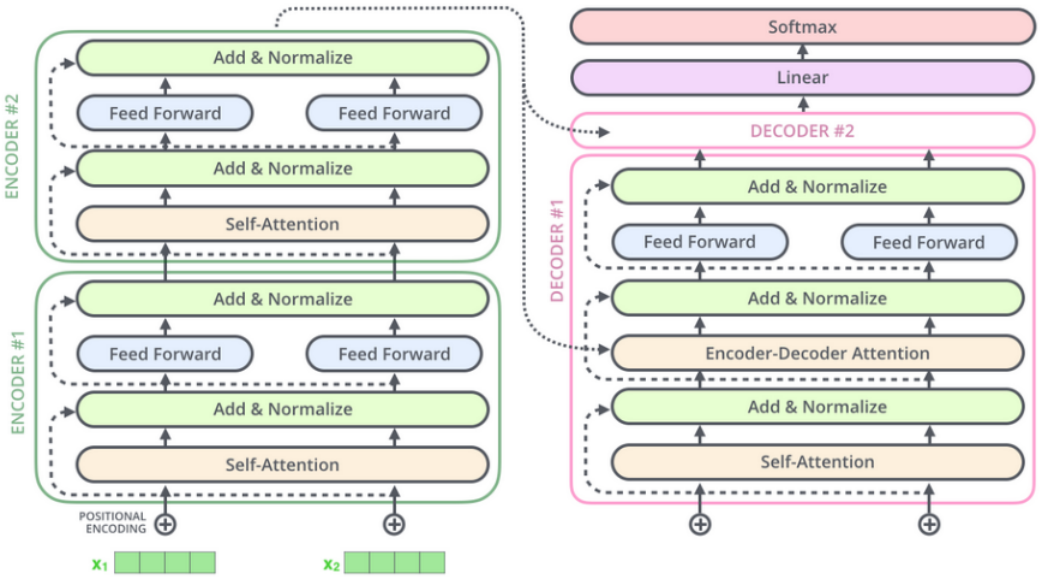

Figure 2.1: Transformer Model with 2 stack encoders & decoders (Alammar)

2.3.2 Self Attention

The Self Attention function operates by taking a query along with a collection of key-value pairs as input, resulting in an output vector. Each of these vectors—query, key, value, and output—is represented as a vector, and the output vector is derived as the weighted sum of the value vectors.

Within a self-attention layer, all the key, value, and query vectors are received from the preceding layer, or directly from the input in the case of the initial layer. This mechanism enables the encoder/decoder layer to attend to all positions within the output of the previous encoder/decoder.

To streamline computation, we organize the key vectors into a matrix denoted as K. Similarly, the query and value vectors are packed into matrices Q and V, respectively. Attention is computed using Formula 2.5, which facilitates efficient processing within the self-attention mechanism.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.5}$$

2.3.3 Encoder-Decoder Attention

Within an Encoder-Decoder attention layer, key and value vectors are sourced from the output of the preceding encoder layer, while query vectors are obtained from the previous decoder layer. This attention mechanism empowers the decoder layer to attend to all positions within the output of the final encoder layer, thereby allowing comprehensive access to the input sequence. Despite this interaction, the mathematical formulation (Formula 2.5) remains consistent for encoder-decoder attention.

2.3.4 Multi-Head Attention

Different projection matrices W_i^Q, W_i^K, W_i^V are used to project the query, key, value vectors. We perform attention mechanism on each of these projected ver- sions. Then, we concatenate the output values and again project it using W^O to get the final output values. Equation 2.6 shows how to calculate multi-head attention:

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O \tag{2.6}$$

where $head_i = Attention(W_i^Q, W_i^K, W_i^V)$.

With single attention head, we were able to attend to different positions of a representation. But, multi-head attention allows the model to attend over different positions of different representation sub-spaces. Generally, we use 8 attention heads.

2.3.5 Positional Embedding

Previous encoder-decoder architectures did not require any positional information of tokens in the sequence because it sequentially processed the tokens. But, Transformer model does not sequen-

tially process the tokens. Hence, it requires positional information of each token. Transformer adds the *Positional Encodings* (which is of the same dimension as of input embeddings) to the input embeddings according to Equation 2.7 and 2.8 where pos is the position and i is the dimension.

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{2.7}$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{2.8}$$

Chapter 3

Disfluency Correction

3.1 Motivation

The cascaded Speech to Speech Machine Translation pipeline comprises an Automatic Speech Recognition (ASR) module, followed by a Machine Translation (MT) module, and finally, a Speech to Text (STT) module. Each of these systems is independently trained to translate utterances from one language to another. The primary responsibility of the Machine Translation (MT) module is to translate text from one language to another, drawing from clean parallel text corpora. To ensure the quality of these corpora, noise is eliminated from the source text to be translated. Unlike written text, which is typically formal and well-structured, spontaneous speech often contains numerous disfluent sentences. However, the Machine Translation (MT) module struggles to handle such sentences with disfluencies, resulting in subpar translation quality. To address this issue, we propose the integration of a Disfluency Correction module, which corrects the disfluencies present in the ASR output, ensuring that fluent sentences are fed into the MT model. Thus, Disfluency Correction can be viewed as a translation task, wherein a disfluent sentence is translated into a fluent one.

3.2 Disfluency Structure and Terminology

Shriberg (34) describes the structure of disfluencies, to consist of four parts: Redarandum, Interruption Point, Interregnum, and Repair. Figure **??** shows an eaxmple of a disfluency following this structure in English as well as in Hindi. Interruption Point is equivalent to the “moment of interruption” and is not explicity present in the transcript, but a part of the speech signal. Hence, we donot capture it in our modelling strategy.

The Reparandum-Interregnum-Repair structure serves as the foundation of our classification methodology. It captures the distinctive patterns associated with reduplication and repetition:

- **Reparandum**: This is the initial segment of the utterance, containing the word or phrase that undergoes repetition or reduplication.

- **Interregnum**: The interregnum is an optional segment that may exist between the repetitions or reduplications. It plays a crucial role in distinguishing the two phenomena, as it often contains disfluent elements or markers.
- **Repair**: The repair segment is where the repetition or reduplication occurs. This is the part of the utterance that replicates the Reparandum, potentially with variations.

3.3 Types of Disfluencies

Numerous types of disfluency phenomena, such as unfilled pauses and uncorrected prosodic errors, are not typically transcribed by Automatic Speech Recognition (ASR) systems and fall outside the scope of the current study. Therefore, our focus is specifically on disfluencies where a continuous segment of linguistic content needs to be removed to achieve the sequence the speaker ”intended” or would likely utter upon a request for repetition. These instances encompass a broad range of phenomena, including filled pauses, repetitions, false starts, repairs, and various other terms.

According to (14), disfluencies are classified into two types: simple and complex disfluencies. Simple disfluencies include filled pauses like “uh,” “ah,” and “um,” as well as discourse markers like “yeah,” “well,” “you know,” and “okay.” This category also includes interjections like “oops” and “ugh.” Many times, words like “yeah”, “okay” are also marked as filled pauses. Where as complex disfluencies often involve multiple elements, such as repetitions, false starts, and repairs, intertwined within a single utterance. Understanding and accurately identifying complex disfluencies pose significant challenges in speech processing tasks. Below are some sorts of disfluencies and an example of each in Telugu with transliteration -

- **Filler Pause**

ఈ ఆట ఎలా ఆడాలి అంటే...మ్మ్...అందరూ అన్ని పేర్లు గుర్తుపెట్టుకోవాలి
Ee aata ela adali ante..umm..andharu anni perlu gurthu pettukovali
This-game-how-play-is…umm…everyone-all-names-should remember
How we play this game is… umm.. Everyone should remember all names

- **Interjection**

నేను రోడ్ పైన నడుస్తున్నాను, ఆహ్ , ఒక కుక్క మీదకి వచ్చింది
Nenu road paina nadusthunnanu, ah, oka kukka na meedaki vachindi
I-on road-was walking-ah-a dog-on me-came
I was walking on the road and… oh! A dog came on to me

- **Discourse marker**

స్క్వాష్ ఒక బాడ్మింటన్ లాంటి ఆట .... ఎలా చెప్పాలి.... మనిషి బదులు గోడ ఉంటది
Squash oka badminton lanti aata…ela cheppali…manishi badhulu goda untadhi
Squash- badminton-like-game…how do I say…human-instead of-wall-is there
Squash is game like badminton…how do I say…there is a wall instead of human

- **Repetition**

నాకు ఆ సినిమా చాలా, చాలా నచ్చింది
Naku aa cinema chaala, chaala nachindi
I-that movie-very-very much-liked
I liked that movie very very much

- **Correction**

నేను ముంబై లో బాంద్రాలో, కాదు, పోవైలో ఉంటున్నాను
Nenu Mumbai lo bandra lo, kadhu, powai lo untunnanu
I-in Mumbai-in bandra-no-in powai-am staying
I am staying in bandra, no, in powai in Mumbai

- **False Start**

వాళ్ళు రాకపోతే...మ్మ్...వాళ్ళు కచ్చితంగా రావాలండి
Vallu rakapothe…umm..vallu kachitanga ravali
They-if won't come-umm-they-must-come
If they won't come…umm…they must come

- **Edits**

నేను నిన్న షాపింగ్ చేశాను.. ర్ సిటీ మాల్ లో షాపింగ్ చేశాను
Nenu ninna shopping chesanu..r-city mall lo shopping chesanu
I-yesterday-shopping-did…r-city mall-in- shopping-did
I went for shopping yesterday…Went for shopping in r-city mall

Chapter 4

Reduplication

Prior to delving into the topic of reduplication, it is essential to grasp some fundamental concepts of morphology. Morphology, a field within linguistics, focuses on analyzing the internal structure of words and their formation from smaller units known as morphemes. A morpheme represents the smallest meaningful unit in a language.

4.1 Morphological Operations

Two primary types of morphological operations exist: inflection and derivation. Inflectional morphological operations alter a word's form to convey grammatical details such as tense, number, and gender. On the other hand, derivational morphological operations generate new words by attaching affixes to a base word.

4.2 Morphological Processes

Morphological processes encompass various methods by which morphemes are assembled to create words. These processes include compounding, derivation, inflection, and the significant process of reduplication.

4.3 Reduplication in English

In English, reduplication serves as a morphological process wherein a word or a segment of a word is duplicated to generate a novel term. For instance, both ”boo-boo” and ”ping-pong” exemplify reduplication.

Reduplication in English frequently functions to emphasize or generate new words with distinct connotations. For instance, the term ”flip-flop” may refer to a style of sandal or denote an abrupt change in perspective or opinion.

4.4 Reduplication in Hindi / Urdu

Reduplication is prevalent in Hindi and Urdu, two closely linked languages spoken in South Asia, serving various linguistic functions such as intensification, plurality, and repetition.

For instance, in Hindi/Urdu, the term ”thanda” signifies ”cold,” whereas ”thanda-thanda” conveys ”very cold” or ”ice-cold.” Similarly, ”paani” denotes ”water,” while ”paani-pani” suggests ”water everywhere” or ”flooding.”

Moreover, reduplication in Hindi/Urdu also denotes plurality, as seen in ”kapda,” meaning ”cloth,” and ”kapda-kapde,” meaning ”clothes.”

Additionally, in Hindi/Urdu, reduplication can express repetition, as exemplified by ”jaldi-jaldi,” signifying ”quickly and repeatedly.”

In essence, reduplication stands as a captivating morphological phenomenon observed in numerous languages worldwide, including Hindi/Urdu. Exploring reduplication aids in gaining a deeper understanding of the intricate processes involved in word formation and usage across diverse linguistic contexts.

4.5 Reduplication in Regional languages

Reduplication is a widespread phenomenon found in languages across the globe, serving various grammatical and semantic functions.

In Telugu, a Dravidian language spoken in India, reduplication is primarily employed for semantic purposes, enhancing or intensifying the meaning of a word. For instance, the word ”rallu” translates to ”stone,” while ”rallu-rallu” denotes ”hard as a rock” or ”stone-like.” Similarly, ”gola” means ”round” or ”ball,” whereas ”gola-gola” conveys ”very round” or ”perfectly spherical.”

In Marathi, an Indo-Aryan language spoken in the Indian state of Maharashtra, reduplication also serves semantic functions, often used to create new words or to intensify meaning. For example, the word ”dada” means ”elder brother,” while ”dada-dada” may suggest ”respected elder” or ”highly revered brother.” Similarly, ”chavat” translates to ”naughty,” while ”chavat-chavat” indicates ”extremely mischievous” or ”playfully naughty.”

Overall, reduplication in Telugu and Marathi, like in other languages, contributes to the richness and flexibility of linguistic expression, allowing for nuances in meaning and emphasis.

For example, in Telugu, a language spoken in the Andhra Pradesh, reduplication is used to intensify the context. The word ”pedha” means ”big,” while ”pedha-pedha” means ”very big.”

Chapter 5

Repetition vs. Reduplication

5.1 Reduplication

The act of repeating all or part of a word for emphasis or to communicate a meaning is known as reduplication. It is common in Indian languages; some examples of reduplication (26) in Telugu are given below:

అతను ఆడటానికి **చిన్ని చిన్ని** కథలు చెప్పిస్తాడు
క్రికెట్ **ఆడి ఆడి** ఆయాసం అనిపిస్తుంది

In the context of disfluencies, it's important to note that reduplication can sometimes be mistaken for repetition disfluencies. Reduplications are intentional and grammatically valid repetitions, distinct from disfluencies. Therefore, we delve into a detailed exploration of reduplications in this chapter to clarify their distinction.

5.2 Reduplication as Multiword Expression

Multiword expressions (MWEs) are a cornerstone of linguistic studies and pose significant challenges in natural language processing (NLP) due to their complex, non-compositional nature. They necessitate specialized computational methods to address their unique syntactic and semantic features. Recent research highlights a framework for integrating MWE processing into NLP systems to improve linguistic understanding.

The identification of reduplication is closely related to the broader task of identifying MWEs, which has been extensively studied in the field of natural language processing. Effective recognition and classification of reduplication can significantly enhance the performance of NLP applications and language understanding systems.

Reduplication in multi-word expressions has been thoroughly studied using different computational models, providing deep insights into its complex behavior. Dolatian and Heinz (9) notably advanced this area by employing two-way finite-state transducers (2-way FSTs) to model virtually

all reduplicative processes, showcasing the versatility of these models in capturing the intricacies of reduplication across languages (9). Further contributing to the computational treatment of reduplication, Wang (40) introduced finite-state buffered machines (FSBMs), which extend regular language frameworks to include reduplication, thereby more accurately characterizing the linguistic phenomena observed in natural languages (40).

On the applied side of NLP, the integration of reduplicated multiword expressions into machine translation systems has shown promising improvements in translation quality. Doren Singh and Bandyopadhyay (11) demonstrated this by incorporating reduplicated expressions and named entities into a Phrase Based Statistical Machine Translation system for the Manipuri language, which significantly enhanced both the BLEU and NIST scores over baseline systems (11). From a database and resource perspective, Dolatian and Heinz (10) introduced RedTyp, a significant resource comprising an SQL database cataloging reduplicative morphemes across numerous languages, modeled using finite-state machines (10). This database serves as a foundational tool for both theoretical and computational linguists studying reduplication.

Significant efforts have been made to computationally address reduplication across languages such as Bengali, Cantonese, Mandarin Chinese, Indonesian, Sanskrit, Hindi, and Marathi (6; 22; 7; 25; 21; 35). Notable computational models include the use of two-way finite-state transducers (2-way FSTs) and finite-state buffered machines (FSBMs), which effectively model reduplicative processes (9; 40). Additionally, integrating reduplicated expressions into machine translation has enhanced translation accuracy (11). The creation of the RedTyp database marks a significant advancement in the cataloging of reduplicative morphemes, aiding both theoretical and computational studies (10). While these studies offer significant theoretical insights, no previous work has released a large-scale dataset specifically for the study of reduplication and repetition.

5.3 Reduplication in English and Telugu

In Telugu, reduplication is also utilized in various forms, each serving specific linguistic functions:

- **Rhyming reduplication:** When the repeated word rhymes with the original word, the repetition is called a rhyming compound, or a rhyming reduplicative.
  Examples: Easy-peasy, hocus-pocus, hokey-pokey, itsy-bitsy, super-duper, etc.
  Similar to English, Telugu employs rhyming reduplication to create expressive compounds. Words are repeated with slight modifications to rhyme and convey emphasis or vividness.
  Examples: అడబ్బు-డబ్బు , బంగారు-గంగారు

- **Exact reduplication:** When the repeated word rhymes with the original word, it is called an exact reduplication. Exact reduplications can be employed to emphasise thenintensity of a word in several versions of English (”He wants it now now”); in South African English, ’now-now’ means’relatively soon.’

Examples: Bye-bye, Aye-aye, back-to-back, blah-blah, boo-boo, night-night, pom-pom.
Telugu also utilizes exact reduplication to emphasize certain words or actions. This form of reduplication repeats the word exactly to highlight its significance or intensity.
Examples: ప్రేమ-ప్రేమ , సిరి-సిరి , నిద్ర-నిద్ర

- **Ablaut reduplication:** In ablaut reduplications, the first vowel is almost always a high vowel (typically ɪ as in hit) and the reduplicated vowel is a low vowel (typically æ as in cat or ɒ as in top).
  Examples: Chit-chat, flip-flop, hip-hop, jibber-jabber, ping-pong, sing-song, tick-tock
  In Telugu, ablaut reduplication involves altering the vowels within a word to create emphasis or intensity. This form of reduplication often conveys a sense of repetition or continuous action.
  Examples: గడ్డగడ్డలు, చిక్కిచిక్కి, వెల్లుల్లు

- **Contrastive focus reduplication:** Exact reduplication can be used in conjunction with contrastive emphasis (usually when the first word is stressed) to imply a literal, rather than metaphorical, example of a noun, or possibly a type of Platonic ideal of the noun, as in ”Is it carrot cheesecake or carrot CAKE-cake?” This is comparable to the above described Finnish use. It is also used to contrast ”genuine” or ”pure” goods with imitations or less pure versions. In a coffee establishment, for example, one may be asked, ”Do you want soy milk?” and answer, ”No, I want milk milk.” This suggests that they desire ”genuine” milk.
  Telugu also utilizes reduplication for contrastive emphasis, highlighting differences or contrasting aspects within a sentence or phrase.
  Examples: నీతో చెందుతున్నాను కదా? నాకు మొదటినే, అడుగు తగిలేందుకు మొదటినే నాకు

5.4 Repetition as Speech Disfluency

Repetition is a well-known speech disfluency often observed in spontaneous and unscripted speech (34). It refers to the unintentional recurrence of words, phrases, or sounds, which may occur due to hesitations, corrections, or cognitive processing.

It is tackled using various computational techniques aimed at enhancing speech recognition and processing. These techniques include Sequence Tagging, Parsing-based, and Noisy Channel models, each leveraging different aspects of machine learning and syntactic analysis (23; 13; 28; 44; 43; 12; 38; 39; 31; 16; 32; 15; 41; 42; 18; 19; 45; 17). Inspired from these works, we move forward with sequence tagging based modeling as this approach has its merits of allowing direct and explicit tagging of disfluencies at the word level, which enables fine-grained detection and classification, critical for developing robust speech recognition systems.

Chapter 6

Dataset Creation and Annotation

6.1 Introduction

In order to differentiate between reduplication and repetition accurately, annotated speech corpora containing both disfluencies and their transcriptions are indispensable. However, as highlighted in section [4.1], disfluencies are frequently overlooked or disregarded as irregularities in many speech corpora. Consequently, annotations of disfluencies are not universally present across all speech datasets. In this context, we outline several existing speech corpora that include annotations of disfluencies, along with synthetic techniques to incorporate disfluencies into datasets lacking such annotations.

6.2 IndicRedRep Dataset

This section discusses the formation of the IndicRedRep dataset, which includes data collection, annotation, and key statistics across three Indic languages: Hindi, Marathi, and Telugu, focusing on token-level labels for reduplication and repetition. Hindi resources are more plentiful, necessitating different collection strategies compared to Marathi and Telugu.

6.2.1 Data Collection

To the best of our knowledge, there currently exists no dataset explicitly annotated for both reduplication and repetition. We employed the GramVaani (GV) corpus, a spontaneous telephone speech corpus in Hindi, to establish the Hindi subset of the dataset, addressing the lack of datasets annotated for reduplication and repetition(8). For Marathi and Telugu, where similar datasets are absent, we extrapolated from the Hindi data using the Gemma Instruction Tuned models(36) for sentence generation and engaged native speakers for manual creation of test sets.

It was important to use a dataset containing spontaneous speech rather than read speech, as disfluencies are more commonly observed in spontaneous speech. However, in datasets such as

the **Shrutilipi corpus** (5), **Indian Language Corpora** (1), and **Mozilla Common Voice** (4), which predominantly feature read speech, the majority of word duplications are the result of either reduplication or transcription errors. True instances of repetition were significantly rarer in these sources.

6.2.2 Annotation and Quality Control Process

The collected data was annotated by three trained linguists, who are language majors with a focus on Hindi. They noted the poor quality and many errors in the transcripts of the GramVaani (GV) corpus. Consequently, data annotation occurred in two stages: initially correcting the speech transcripts and subsequently labeling the tokens as reduplication, repetition, or other.

Given the noisy annotations in the GV corpus, which often skipped repetition words, we filtered the corpus to select sentences likely containing reduplication or repetition. After filtering, these sentences were reannotated to properly account for missed instances. This resulted in a clean, ground truth dataset with real-world instances of reduplication and repetition in Hindi, which were then labeled at the token level. For Marathi and Telugu, we used the cleaned and annotated Hindi subset to bootstrap sentence generation with the Gemma Instruction Tuned models (36). Native speakers of Marathi and Telugu were later engaged to manually create sentences for the test sets, using the Hindi data as a reference.

Annotation guidelines were developed based on existing works (27) and are detailed in Appendix **??**. To ensure consistency, interannotator agreement was assessed using Fleiss' kappa, resulting in an agreement level of 83.29%, indicating substantial consistency. Quality control included independent re-annotation by multiple annotators, with discrepancies resolved during regular meetings (33).

6.2.3 Dataset Statistics

The GramVaani corpus, inherently rich in colloquial expressions and spontaneous speech patterns, provided an ideal foundation for our specific annotations. Table **??** shows the number of sentences and words, across each split in the dataset. As showcased in Table **??**, our annotated dataset comprises of labels: *reduplication*, *repetition* and *other*. The presence of 3,263 instances of repetition and 2,340 of repetition underscores the diversity and richness of this corpus in capturing these linguistic phenomena. Additionally, there are 586 tokens labeled as 'other', indicating elements that did not fit into the primary categories but were distinct from the neutral 'O' category.

6.2.4 Data Splits

The data was divided into training, validation, and test sets following the standard 80:10:10 ratio. The splits were stratified to ensure that the distribution of reduplication and repetition instances was similar across all subsets as can be seen in Table

Language	Data Splits	#sentences	#words	Split Size
Hindi	Training	3622	103602	80%
	Validation	453	12950	10%
	Test	453	12950	10%
Telugu	Training	1289	36860	80%
	Validation	161	4608	10%
	Test	161	4608	10%
Marathi	Training	1322	37822	80%
	Validation	165	4728	10%
	Test	165	4728	10%

Table 2: Dataset statistics across three languages for a token classification task

	Training	Validation	Test	Total
repetition	2598	335	330	3263
reduplication	1875	230	235	2340
other	462	62	62	586
Total	4935	627	627	6189

Table 3: Number of labels of each type in Training, Validation, Test splits for the **IndicRedRep** dataset

Chapter 7

Conclusion

7.1 Summary

In this thesis, we explored the significance of Disfluency Correction in refining outputs from Automatic Speech Recognition (ASR) systems before their integration into downstream Natural Language Processing (NLP) applications like Machine Translation. We conducted a comprehensive examination of the disfluency phenomenon, encompassing various types of disfluencies and associated terminology. Additionally, we reviewed prior research on Disfluency Correction and the diverse methodologies employed to address this task. Our motivation for incorporating the RiR structure into our classification approach stems from its ability to disentangle complex linguistic structures especially when both reduplication and repetition are present in the same sentence. Furthermore, we underscored the prevalence of reduplication in Indic languages and the potential confusion between reduplication and repetition. We also discussed distinct phenomena observed in reduplication across English, Hindi, Telugu and Marathi

Reduplication is repeating all or part of a word to express various grammatical, semantic, or pragmatic nuance. *Repetition* is the recurrence of a word, phrase, or sound in speech, often unintentional and indicative of disfluency. Both reduplication and repetition are widespread in communication and conversational speech and across many languages, and they are used for various linguistic and communicative purposes, such as adding emphasis, expressing hesitation, or signaling clarification. Given the scarcity of datasets for disfluency correction, particularly in Indian languages, we outlined commonly used datasets and proposed techniques for generating synthetic datasets using existing corpora. To the best of our knowledge, we present the first large scale study of reduplication and repetition together in speech using computational linguistics, bringing together previous work in linguistics and speech disfluencies. To facilitate this, we introduce a new publicly available data set IndicRedRep; consisting of Hindi, Telugu and Marathi text annotated with reduplication and repetition at the word level. We empirically evaluate different transformer-based models for multi-class reduplication and repetition token classification.

7.2 Results

Results for all the models across the three languages, fine-tuned on the IndicRedRep dataset, are shown in Table.

Lang	Type of Error	Sentence	Prediction	Comments
Hi	Reduplication	को घर घर ये सेवा पहुँचे तो इसके माध्यम से मैं ये बताना चाहता हूँ की हमारे जो झारखण्ड झारखण्ड	को घर घर ये सेवा पहुँचे तो इसके माध्यम से मैं ये बताना चाहता हूँ की हमारे जो झारखण्ड झारखण्ड	घर (ghar, 'house') is an example of reduplication class, but is confused with repetition.
Hi	Repetition	यह हमारे समाज के लिए नहीं बल्कि प्राचीन समय समय से ही हमारा समाज जूझ रहा है अगर हमारे समाज में कहीं भी कोई घरेलु हिंसा होती है तो इसका शिकार महिलाओं को ही	यह हमारे समाज के लिए नहीं बल्कि प्राचीन समय समय से ही हमारा समाज जूझ रहा है अगर हमारे समाज में कहीं भी कोई घरेलु हिंसा होती है तो इसका शिकार महिलाओं को ही	समय (samay, 'time') is is an example of repetition, but incorrectly predicted as reduplication.
Te	Reduplication	మరల మరల సహాయం చేసినందుకు ధన్యవాదాలు ధన్యవాదాలు	మరల మరల సహాయం చేసినందుకు ధన్యవాదాలు ధన్యవాదాలు	మరల (marala, 'again') is incorrectly predicted as repetition, while the correct label is reduplication
Te	Repetition	ఉదయం ఉదయం గుడ్ మార్నింగ్ చెప్పాలి ఎందుకంటే నా నమ్మకం పిల్లలు పిల్లలు తొలి పాఠశాల ఇల్లే ఇల్లే ఉంటుంది మరియు ఉపాధ్యాయురాలు తల్లే అవుతుంది.	ఉదయం ఉదయం గుడ్ మార్నింగ్ చెప్పాలి ఎందుకంటే నా నమ్మకం పిల్లలు పిల్లలు తొలి పాఠశాల ఇల్లే ఇల్లే ఉంటుంది మరియు ఉపాధ్యాయురాలు తల్లే అవుతుంది.	ఇల్లే (ille, 'house')is predicted as reduplication by the model, but it is an example of repetition.
Mr	Reduplication	दूध विकत असताना रुग्णालयाच्या विभागासाठी वेगवेगळी वेगवेगळी तारीख ठरवली आहे.	दूध विकत असताना रुग्णालयाच्या विभागासाठी वेगवेगळी वेगवेगळी तारीख ठरवली आहे.	वेगवेगळी (vegvegali, 'different') is an example of reduplication in Marathi which is incorrectly predicted as repetition.
Mr	Repetition	सतरा आदि आदि जिल्ह्यांमधून दोनशे दोनशे कार्यकर्ते सहभागी होतील, धन्यवाद.	सतरा आदि आदि जिल्ह्यांमधून दोनशे दोनशे कार्यकर्ते सहभागी होतील, धन्यवाद.	आदि (ādi, 'so on') is an example of repetition in Marathi, but it is mis-classified as reduplication.

To evaluate the performance of all models used in our experiments, we use precision, recall, and F1 score; metrics that are commonly used across classification tasks in natural language processing (24; 20). These metrics have also been used in previous related works focusing on disfluency detection and similar tasks (17; 29).

7.3 Limitations

Given the complexities of disambiguating reduplication and repetition in different languages, our study, while rigorous, presents limitations that are acknowledged below:

- **Generalization across Languages**: Our experiments were limited to three languages: Hindi, Telugu, and Marathi. The generalizability of our results to other languages, particularly non-Indo-European languages, is uncertain. Future studies should explore the application of the RiR structure in a broader linguistic context to verify its effectiveness across a wider array of language families.

- **RiR Structure**: Our approach heavily relies on the RiR structure to classify disfluencies. While effective, this method assumes that the disfluency markers and structures are consistent and distinguishable, which may not hold in more colloquial or less structured speech datasets.

- **Other Subword Representations**: Our study focused exclusively on transformer-based models (BERT and XLMR) with the addition of the RiR structure. We did not include other potent subword representations like ELMo (30) and contextual string embeddings (2), which might offer different advantages in handling complex language phenomena. The lack of availability of these models in multiple languages restricted their inclusion in our study.

To address these limitations, future research should aim to include a more diverse set of languages and linguistic structures. Moreover, experimenting with additional subword representations and extending the RiR framework to accommodate more varied disfluency types could enhance model robustness. An exploration of the impacts of different preprocessing techniques on the model's ability to recognize and classify speech patterns accurately would also be beneficial.

7.4 Future Researh Directions

Our study introduced and validated a model that employs the Reparandum-Interregnum-Repair (RiR) structure to enhance the classification of linguistic phenomena such as reduplication and repetition in multilingual contexts. Our experiments, as detailed in the table, demonstrated that incorporating the RiR structure consistently improves the performance across multiple languages (Hindi, Telugu, and Marathi), as evidenced by higher F1 scores when compared to baseline models, including both traditional ML-based and advanced transformer models.

The RiR structure's utility in distinguishing complex linguistic patterns is particularly notable. This approach provided clear benefits over traditional models like Logistic Regression and BiLSTM-CRF, and even showed marked improvement over advanced models like the multilingual BERT and XLMR in their standard configurations. The most significant improvements were observed with the XLMR-large + RiR model, highlighting the effectiveness of integrating structural linguistic insights into sophisticated neural architectures for NLP tasks.

Future research should expand our approach to include more languages, especially those under-represented in NLP, and explore additional linguistic structures beyond the RiR to enhance understanding of language processing. There is also potential to integrate our RiR like models to localize context in other NLP tasks like syntactic parsing or semantic analysis, which could lead to improvements in complex language understanding systems. Moreover, applying our model's insights to real-world applications such as speech recognition could significantly enhance their effectiveness and accuracy.

Bibliography

[1] Abraham, B., Goel, D., Siddarth, D., Bali, K., Chopra, M., Choudhury, M., Joshi, P., Jyoti, P., Sitaram, S., and Seshadri, V. (2020). Crowdsourcing speech data for low-resource languages from low-income workers. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2819–2826, Marseille, France. European Language Resources Association.

[2] Akbik, A., Blythe, D., and Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In *Proceedings of the 27th international conference on computational linguistics*, pages 1638–1649.

[Alammar] Alammar, J. The illustrated transformer.

[4] Ardila, R., Branson, M., Davis, K., Kohler, M., Meyer, J., Henretty, M., Morais, R., Saunders, L., Tyers, F., and Weber, G. (2020). Common voice: A massively-multilingual speech corpus. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4218–4222, Marseille, France. European Language Resources Association.

[5] Bhogale, K., Raman, A., Javed, T., Doddapaneni, S., Kunchukuttan, A., Kumar, P., and Khapra, M. M. (2023). Effectiveness of mining audio and text pairs from public data for improving asr systems for low-resource languages. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

[6] Chakraborty, T. and Bandyopadhyay, S. (2010). Identification of reduplication in bengali corpus and their semantic analysis: A rule based approach. In *Proceedings of the 2010 Workshop on Multiword Expressions: from Theory to Applications*, pages 73–76.

[7] Chen, F.-y., Mo, R.-p., Huang, C.-R., and Chen, K.-J. (1992). Reduplication in mandarin chinese: Their formation rules, syntactic behavior and icg representation. In *Proceedings of rocling v computational linguistics conference v*, pages 217–233.

[8] Deekshitha, G., Singh, A., et al. (2022). Gram vaani asr challenge on spontaneous telephone speech recordings in regional variations of hindi. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, volume 2022, pages 3548–3552. International Speech Communication Association.

[9] Dolatian, H. and Heinz, J. (2018). Modeling reduplication with 2-way finite-state transducers. In Kuebler, S. and Nicolai, G., editors, *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 66–77, Brussels, Belgium. Association for Computational Linguistics.

[10] Dolatian, H. and Heinz, J. (2019). Redtyp: A database of reduplication with computational models. *Society for Computation in Linguistics*, 2(1).

[11] Doren Singh, T. and Bandyopadhyay, S. (2011). Integration of reduplicated multiword expressions and named entities in a phrase based statistical machine translation system. In Wang, H. and Yarowsky, D., editors, *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1304–1312, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

[12] Ferguson, J., Durrett, G., and Klein, D. (2015). Disfluency detection with a semi-Markov model and prosodic features. In Mihalcea, R., Chai, J., and Sarkar, A., editors, *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 257–262, Denver, Colorado. Association for Computational Linguistics.

[13] Georgila, K., Wang, N., and Gratch, J. (2010). Cross-domain speech disfluency detection. In Katagiri, Y. and Nakano, M., editors, *Proceedings of the SIGDIAL 2010 Conference*, pages 237–240, Tokyo, Japan. Association for Computational Linguistics.

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

[15] Honnibal, M. and Johnson, M. (2014). Joint Incremental Disfluency Detection and Dependency Parsing. *Transactions of the Association for Computational Linguistics*, 2:131–142.

[16] Hough, J. and Schlangen, D. (2015). Recurrent neural networks for incremental disfluency detection. In *Proc. Interspeech 2015*, pages 849–853.

[17] Jamshid Lou, P. and Johnson, M. (2017). Disfluency detection using a noisy channel model and a deep neural language model. In Barzilay, R. and Kan, M.-Y., editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 547–553, Vancouver, Canada. Association for Computational Linguistics.

[18] Jamshid Lou, P. and Johnson, M. (2020). Improving disfluency detection by self-training a self-attentive model. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3754–3763, Online. Association for Computational Linguistics.

[19] Johnson, M. and Charniak, E. (2004). A tag-based noisy-channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 33–39.

[20] Jurafsky, D. (2000). *Speech & language processing*. Pearson Education India.

[21] Kulkarni, A., Paul, S., Kulkarni, M., Nelakanti, A. K., and Surtani, N. (2012). Semantic processing of compounds in indian languages. In *Proceedings of COLING 2012*, pages 1489–1502.

[22] Lam, C. (2013). Reduplication across categories in cantonese. In *Proceedings of the 27th Pacific Asia Conference on Language, Information, and Computation (PACLIC 27)*, pages 277–286.

[23] Liu, Y., Shriberg, E., Stolcke, A., Hillard, D., Ostendorf, M., and Harper, M. (2006). Enriching speech recognition with automatic detection of sentence boundaries and disfluencies. *IEEE Transactions on audio, speech, and language processing*, 14(5):1526–1540.

[24] Manning, C. and Schutze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

[25] Mistica, M., Arka, I. W., Baldwin, T., and Andrews, A. (2009). Double double, morphology and trouble: Looking into reduplication in indonesian. In *Proceedings of the Australasian Language Technology Association Workshop 2009*, pages 44–52.

[26] Montaut, A. (2009). Reduplication and echo words in hindi/urdu. *Annual review of South Asian languages and linguistics*, pages 21–91.

[27] Murthy, R., Bhattacharjee, P., Sharnagat, R., Khatri, J., Kanojia, D., and Bhattacharyya, P. (2022). Hiner: A large hindi named entity recognition dataset. *arXiv preprint arXiv:2204.13743*.

[28] Ostendorf, M. and Hahn, S. (2013). A sequential repetition model for improved disfluency detection. In *Proc. Interspeech 2013*, pages 2624–2628.

[29] Passali, T., Mavropoulos, T., Tsoumakas, G., Meditskos, G., and Vrochidis, S. (2022). LARD: Large-scale artificial disfluency generation. In Calzolari, N., Béchet, F., Blache, P., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Odijk, J., and Piperidis, S., editors, *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2327–2336, Marseille, France. European Language Resources Association.

[30] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (1802). Deep contextualized word representations. corr abs/1802.05365 (2018). *arXiv preprint arXiv:1802.05365*, 42.

[31] Qian, X. and Liu, Y. (2013). Disfluency detection using multi-step stacked learning. In Vanderwende, L., Daumé III, H., and Kirchhoff, K., editors, *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 820–825, Atlanta, Georgia. Association for Computational Linguistics.

[32] Rasooli, M. S. and Tetreault, J. (2013). Joint parsing and disfluency detection in linear time. In Yarowsky, D., Baldwin, T., Korhonen, A., Livescu, K., and Bethard, S., editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 124–129, Seattle, Washington, USA. Association for Computational Linguistics.

[33] Sabou, M., Bontcheva, K., Derczynski, L., and Scharl, A. (2014). Corpus annotation through crowdsourcing: Towards best practice guidelines. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 859–866, Reykjavik, Iceland. European Language Resources Association (ELRA).

[34] Shriberg, E. E. (1994). *Preliminaries to a theory of speech disfluencies*. PhD thesis, Citeseer.

[35] Singh, D., Bhingardive, S., and Bhattacharyya, P. (2016). Multiword expressions dataset for indian languages. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2331–2335.

[36] Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., et al. (2024). Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

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

[38] Wang, F., Chen, W., Yang, Z., Dong, Q., Xu, S., and Xu, B. (2018). Semi-supervised disfluency detection. In Bender, E. M., Derczynski, L., and Isabelle, P., editors, *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3529–3538, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

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

[40] Wang, Y. (2021). Recognizing reduplicated forms: Finite-state buffered machines. In Nicolai, G., Gorman, K., and Cotterell, R., editors, *Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 177–187, Online. Association for Computational Linguistics.

[41] Wu, S., Zhang, D., Zhou, M., and Zhao, T. (2015). Efficient disfluency detection with transition-based parsing. In Zong, C. and Strube, M., editors, *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 495–503, Beijing, China. Association for Computational Linguistics.

[42] Yoshikawa, M., Shindo, H., and Matsumoto, Y. (2016). Joint transition-based dependency parsing and disfluency detection for automatic speech recognition texts. In Su, J., Duh, K., and Carreras, X., editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1036–1041, Austin, Texas. Association for Computational Linguistics.

[43] Zayats, V., Ostendorf, M., and Hajishirzi, H. (2014). Multi-domain disfluency and repair detection. In *INTERSPEECH*, pages 2907–2911.

[44] Zayats, V., Ostendorf, M., and Hajishirzi, H. (2016). Disfluency Detection Using a Bidirectional LSTM. In *Proc. Interspeech 2016*, pages 2523–2527.

[45] Zwarts, S. and Johnson, M. (2011). The impact of language models and loss functions on repair disfluency detection. In Lin, D., Matsumoto, Y., and Mihalcea, R., editors, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 703–711, Portland, Oregon, USA. Association for Computational Linguistics.