Survey: Disfluency Correction For Indo-European Languages

Vineet Bhat, Pushpak Bhattacharyya
Indian Institute of Technology Bombay, India
vineetbhat2104@gmail.com, pb@cse.iitb.ac.in

Abstract

This survey paper provides a comprehensive overview of Disfluency Correction (DC), emphasizing on the key concepts, techniques and datasets used in this subject. DC is the process of removing disfluent elements like fillers, repetitions and corrections from spoken utterances to create readable and interpretable text. DC is a vital post-processing step applied to Automatic Speech Recognition (ASR) outputs, before subsequent processing by downstream language understanding tasks. Existing DC research has primarily focused on English due to the unavailability of large-scale open-source datasets. Through this survey paper, we also present a high-quality human-annotated DC corpus covering four important Indo-European languages: English, Hindi, German and French.

1 Introduction

Humans often think and speak simultaneously in conversations, introducing erroneous words in utterances (Gupta et al., 2021). These words do not contribute to semantics of a sentence and hence can be removed to create fluent and easy-to-interpret utterances. Disfluency Correction (DC) is defined as the removal of such disfluent elements from spoken utterances (Shriberg, 1994). Different types of disfluencies are described in table 2.

Apart from making sentences readable and interpretable, DC also helps downstream natural language processing tasks like Machine Translation (MT) (Rao et al., 2007; Wang et al., 2010). Removing disfluencies shortens sentences, making it easier for automatic MT systems to translate these utterances. Moreover, the removed erroneous words are not translated which makes the output translation fluent containing all semantics from the source sentence. Table 1 illustrates examples where Google MT produces disfluent and difficult-to-read English translations of disfluent sentences in 3 languages - Hindi, German and French, establishing the need for DC.

Disfluent Sentence	Google MT output
वाच में रनिंग अह्ह्ह स्मार्ट वॉच में रोका हुआ रनिंग टाइमर रिज्यूम करो	running in watch ahhh resume running paused timer in smart watch
je veux je veux euh enregistrer une une euh vidéo sur instagram	I want I want uh record a uh video on instagram
ich brauche eine fahrt äh eine fahrt zum bahnhof in einer stunde	I need a ride er a ride to the train station in an hour

Table 1: English translations produced by Google MT for disfluent sentences in Hindi, French and German. All disfluent words are marked in red.

1.1 Problem Statement

Disfluency correction is a significant challenge in speech-to-speech machine translation. The fundamental problem it aims to address is the identification and removal or correction of disfluencies in the translation process. Disfluencies are often described as breaks in the fluidity of speech, including phenomena like filled pauses (e.g., ”um,” ”uh”), false starts, repetitions, or repairs. These phenomena, which are very common in natural spoken language, can cause confusion or misunderstandings when a machine tries to interpret and translate the spoken text.

The overarching goal in disfluency correction is to construct a machine learning model that can accurately discard disfluencies, thereby ensuring the downstream translated output is both grammatically correct and semantically accurate. This area of research becomes even more vital in real-world applications where seamless, efficient, and effective communication is paramount, such as in international diplomacy, business meetings, healthcare, and more.

1.2 Motivation

The impact of disfluencies in spoken language, particularly in the context of machine translation, is significant. The presence of disfluencies can significantly degrade the performance of machine translation systems. First, disfluencies can cause the translation to be incorrect or hard to understand. Second, they can confuse the syntax and semantics of the original speech, causing misinterpretations. Lastly, disfluencies can make the translated speech seem unnatural or stilted, thereby undermining the goal of achieving natural, human-like speech in translation.

Hence, addressing the disfluency problem is crucial for improving the quality of speech-to-speech machine translation. A system that can effectively remove or correct disfluencies will not only increase translation accuracy but will also improve the naturalness and fluidity of the translated speech, making the conversation more human-like.

Developing machine learning models for disfluency correction in speech-to-speech translation presents several notable challenges:

- **Lack of Quality Training Data:** There is a scarcity of comprehensive, high-quality training datasets that contain both fluent and disfluent speech for various languages. Constructing such datasets requires time-consuming manual annotation.

- **Contextual Understanding:** Disfluencies are heavily reliant on the context in which they occur. For a machine to identify disfluencies accurately, it requires a deep understanding of the language, topic, and even the speaker's intent, which is difficult to achieve.

- **Complexity of Disfluencies:** Disfluencies can take many forms and aren't limited to mere hesitations or repetitions. They can encompass complex language phenomena like self-corrections or sentence reformulations. The sheer variety of these patterns adds to the complexity of the problem.

- **Balancing Accuracy and Fluency:** Striking the right balance between removing disfluencies and preserving the original meaning can be challenging. Overcorrection might lead to the loss of crucial information or distort the original message.

- **Real-Time Processing:** For many applications, translations need to be performed in real-time, which puts constraints on the processing time. This makes the task more challenging as the model needs to detect and correct disfluencies on the fly.

2 Background

In this section, we cover all important terminologies needed to study disfluency correction and perform a variety of experiments.

2.1 Disfluency Correction

Disfluency correction is the task of correcting non-fluent conversational elements such as filler words and pauses in speech transcripts. In day-to-day spontaneous conversations, humans are tasked with thinking, developing constructive thoughts and speaking almost simultaneously resulting in disfluent words and pauses like *uh, um, ah, etc.* as a part of spoken language. These irregularities significantly impact conversational clarity & affect readability in speech transcripts when Automatic Speech Recognition (ASR) systems convert spoken utterances to text. If these transcriptions are not fluent, it may lead to further problems in downstream processing of speech records. Here is an example of a disfluent sentence with all the filler words marked in bold -

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

In the above example, the essence of the sentence is represented only through the phrase - "This is a good plan". Rest of the words act as filler words and should be removed by a disfluency correction system.

2.2 Speech to Speech Machine Translation

Speech to Speech Machine Translation integrates all the above mentioned modules together to convert speech in source language to speech in target language. Cascaded approaches to solve this problem use a pipeline of ASR, DC, MT and TTS models whereas End-to-End approaches use a deep neural network capable of directly learning the mapping between the speech in source-target language pair.

We aim to find the most likely target translation T* from the set T of possible translations in the MT hypothesis space using input speech features X. S denotes the transcription of of the speech features X.

2.2.1 Cascaded Approach

Our goal is to calculate the following -

$$T^* = argmax(P(T|X)) \tag{1}$$

Marginalizing over all transcriptions S ∈ H , we can rewrite this equation as -

$$T^* = argmax(\sum_{S \in H} P(T,S|X)) \tag{2}$$

Using chain rule,

$$T^* = argmax(\sum_{S \in H} P(T|S,X)P(S|X)) \tag{3}$$

We use the conditional independence assumption of the input S and output T given the transcript X to write the following final equation -

$$T^* = argmax(\sum_{S \in H} P_{MT}(T|S,X)P_{ASR}(S|X)) \tag{4}$$

The predicted text T* then passes through the TTS model to produce the required speech output. The figure below demonstrates the pipeline used for training -

2.2.2 End-to-End Approach

Cascaded approaches to SSMT often have a problem of propogating errors. Any errors in the ASR system may lead to bad transcription which causes further translations to be erroneous. End-to-end approaches aim to solve this problem by training the model directly to learn speech from source language to speech in target language. Vanilla RNNs are used for this purpose in an encoder-decoder architecture. The encoder converts the variable length input (speech features in source language) to a fixed length context vector. This vector is then used by the the decoder to produce the speech features in the target language.

Consider the input sequence X = $(x_1, x_2, ...x_n)$. The encoder RNN processes each input timestep x_t one step at a time and changes its hidden state with using the following equation -

$$h_t = f(h_{t-1}, x_t) \tag{5}$$

Once the entire input is processed and the context vector c is created, the RNN decoder changes its hidden state using the equation -

$$h_t = f(h_{t-1}, y_{t-1}, c) \tag{6}$$

Thus the decoder uses the previous hidden state, the previously generated output and the context vector to change the current hidden state and predict the final output. Probability of the next output token is given by -

$$P(y_t|y_1, y_2, ...y_{t-2}, y_{t-1}, c) = g(h_t, y_{t-1}, c) \tag{7}$$

where f and g are the activation functions.

The figure below demonstrates a typical encoder-decoder architecture that can be used -

2.2.3 Attention Mechanism

Since the encoder-decoder architectures use a fixed context vector to encode information, in case of a long input the context vector might lose out on long term dependencies. To solve this problem we use attention mechanism. Instead of predicted a fixed vector c, we predict distinct context vectors c_i using the following formula -

$$c_i = \sum_{t=1}^{T} \alpha_{ij} h_j \tag{8}$$

where,

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{t=1}^{T} exp(e_{it})} \tag{9}$$

$$e_{ij} = a(s_{i-1}, h_j) \tag{10}$$

and a is a feed forward network.S

2.3 Models

In this section, we describe all the models we use for our experiments.

2.3.1 Conditional Random Fields

Conditional Random Fields (CRFs) are a class of statistical modeling methods often applied in pattern recognition and machine learning, where they are used for structured prediction. Unlike other techniques that make their predictions independently, CRFs make their predictions based on the context within the input, considering the correlations between adjacent predictions. They were introduced in (Lafferty et al., 2001).

The key defining property of a CRF is that it models the conditional distribution p(y|x) of the output y given the input x, and thus the dependencies among the y are modeled, given the x. CRFs are undirected graphical models, a special case of which are commonly used for sequence modeling.

The standard form of CRF applicable to sequence modeling and natural language processing (NLP) tasks is the linear-chain CRF, where the graph structure forms a simple chain. The conditional probability of a state sequence (y) given an observation sequence (x) in linear-chain CRFs is defined as:

Here, Z(x) is a normalization factor ensuring the probabilities sum to 1, f_k is a feature function which maps input-output pairs to real values, and λ_k is the learned weight associated with feature f_k. The functions f_k are usually chosen to express the intuition about the task at hand.

2.3.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of artificial neural network designed to recognize patterns in sequences of data, such as text, genomes, handwriting, or spoken words. A distinctive feature of RNNs is their internal memory used to process sequences, making them uniquely suitable for tasks where context from earlier inputs is required to understand current ones.

The standard RNN updates its hidden state h at each time step t using both the current input x_t and the previous hidden state h_{t-1}, computed as:

Here, W and U are weight matrices, b is a bias vector, and f is an activation function like tanh or ReLU. The output y_t at each time step can be computed using a separate weight matrix V:

where g is typically a softmax function for classification problems, and c is another bias vector.

Despite their expressiveness, standard RNNs suffer from a significant drawback: they struggle to learn long-range dependencies due to the "vanishing or exploding gradients" problem. This limitation has been largely addressed by variants like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), which introduce gating mechanisms to control and manage the flow of information (Hochreiter and Schmidhuber, 1997; Cho et al., 2014). Figure 1 depicts the structure of RNN.

2.3.3 Transformers

Recurrent Neural Networks process input tokens sequentially and hence cannot be parallelized. Transformers were introduced in [(Vaswani et al., 2017)] and provided a method to parallelize learning context vectors through self attention mechanisms thereby reducing the time taken to train models. We discuss the key components of a transformer model below.

Encoder and Decoder Stack: Transformers fall under the category of encoder-decoder architectures. The encoder consists of a stack of six identical encoder layers with different weights. Each encoder layer consists of two sub layers - Multi head self attention layer and Feed Forward network layer. The decoder also consists of 6 identical layers and has another additional sub layer called Multi head encoder-decoder attention layer in addition to the other sub layers. The figure below shows an example of a simple transformer model.

Self Attention Layers: Self attention layers use the query and a set of key-value pairs to calculate the output token. These three variables are obtained from the previous layer or from the input in case of the first layer and thus the encoder/decoder layers attend over all the positions of the output of the previous

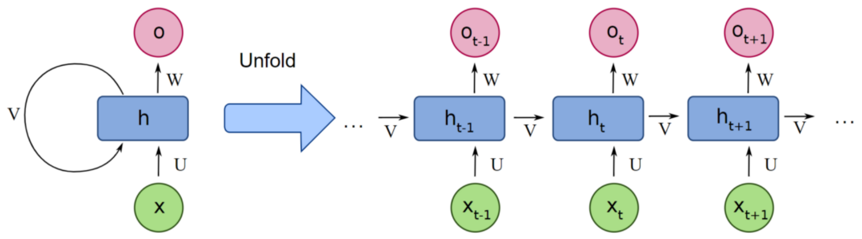

Figure 1: Unfolding of RNNs

encoder/decoder layer. Let Q be the matrix of query vectors, K be the matrix of key vectors and V be the matrix of value vectors. Attention is calculated using the formula -

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{11}$$

2.3.4 Encoder - Decoder Attention Layers

The encoder-decoder attention layer is similar to the self attention layers but with a key distinction of the source of queries. The query for these layers come is received from the last encoder layer whereas the key-value pair comes from the last decoder layer. This mechanism allows the decoder to attend over all the tokens of the last encoder layer thereby attending over all the tokens of the input.

Multi Head Attention Layers: Multi head attention layers use weight matrices to project the query, key and values. Attention is calculated on each of these projected vectors and is concatenated followed by a final projection to calculate the output token. Thus, multi head attention layers allow the model to attend over multiple positions of the input.

Positional Embeddings: Before transformers, RNNs were used as an encoder-decoder structure. They calculated the output by processing each input token sequentially. Since transformers do not do this process sequentially, positional encodings need to be embedded along with the input embeddings so that the model remembers the position of each token. Positional embeddings are calculated using the formula below -

$$PE(pos, 2i) = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{12}$$

$$PE(pos, 2i+1) = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{13}$$

2.3.5 Multilingual Representations for Indian Languages (MuRIL) Transformer

MuRIL (Khanuja et al., 2021) is a variant of the popular BERT model (Devlin et al., 2019), a transformer-based model for natural language understanding. Similar to BERT, MuRIL is trained on large amounts of text data and produces context-sensitive representations of text, meaning that the same word can be represented differently depending on the context it is used in. These representations can be used in various downstream tasks such as text classification, named entity recognition, or sentiment analysis.

What makes MuRIL stand out is its ability to handle transliterated text and its strong performance on Indian languages. Transliterated text is text that has been converted from one writing system to another. This is a common occurrence in Indian languages, many of which are often written using the Latin alphabet on digital platforms.

MuRIL is trained not only on English text but also on text in several Indian languages. Importantly, it treats transliterated text as belonging to the same language as the original text. For example, Hindi written in the Latin script is treated as Hindi, not as English. This ability to handle both native scripts and transliteration makes MuRIL particularly useful for processing Indian language data, where transliteration is common.

Another advantage of MuRIL is that it provides representations for the so-called "zero-shot" languages. That is, even for Indian languages for which no training data was available, MuRIL is able to generate useful representations based on its multilingual training, allowing it to be used with these languages.

The addition of MuRIL to the range of multilingual models marks a step forward in the ability to process less-resourced languages and brings us closer to the goal of universal language understanding.

2.3.6 Generative Adversarial Neural Networks

Generative Adversarial Networks (GANs) are a class of artificial intelligence algorithms, introduced by Ian Goodfellow and his colleagues in 2014 (Goodfellow et al., 2014). They are used in unsupervised machine learning, employing two neural networks contesting with each other in a zero-sum game framework. This setup enables the generation of new, previously unseen data that mimic the distribution of the training data.

GANs consist of two parts: a generator network and a discriminator network. The generator network attempts to create synthetic data samples that mimic the original data, while the discriminator network attempts to differentiate between real and synthetic (generated) data. The generator's goal is to generate data so that it is indistinguishable from the real data in the discriminator's view. On the other hand, the discriminator improves itself to differentiate between the real and the generated data better. This mutual competition results in the generator producing high-quality synthetic data that resemble the original data closely.

The objective function of a GAN can be described by the following minimax game:

Here, D(x) represents the discriminator's estimate of the probability that real data instance x is real. $E_x\ P_data(x)$ denotes the expected value over all real data instances. G(z) represents the data generated by the generator network, z is a noise variable, and $E_z\ P_z(z)$ denotes the expected value over all noise variables z.

GANs in Textual Machine Learning: While GANs have been quite successful for continuous data such as images, their application to discrete data like text is nontrivial, primarily due to the discontinuity of discrete data, which poses challenges for backpropagation in training. However, researchers have devised various methods to apply GANs in text generation tasks.

One approach is the SeqGAN model proposed by (Yu et al., 2017). SeqGAN (figure 2) uses Reinforcement Learning (RL) to bypass the generator's differentiation problem and directly performs gradient policy update. SeqGAN models the data generator as a stochastic policy in reinforcement learning and uses a discriminator to return reward signals. However, unlike traditional reinforcement learning, in SeqGAN, rewards are sparse and delayed due to the long generation sequences, for which a Monte Carlo search is used to approximate the reward for intermediate state-actions.

Another approach is the Gumbel-Softmax GANs (Jang et al., 2016). They introduce a Gumbel-Softmax approximation that lets the model generate discrete tokens while still allowing gradients to flow backward through the generator.

GANs in textual machine learning have been applied to various tasks such as text generation, machine translation, text summarization, dialogue systems, etc. While there are still challenges to be overcome, such as mode collapse and training instability, these networks have shown the capability to generate high-quality, diverse text, making them a promising direction for future research in natural language processing.

3 Related Work

The study of disfluencies as a spoken language phenomenon was first proposed in Shriberg (1994). DC has been established as a vital post-processing task for ASR transcripts (Rao et al., 2007; Wang et al., 2010). Although earlier DC systems were based on translation methods (Honal and Schultz, 2003), current research covers two additional methods: parsing-based and sequence tagging-based techniques. Translation-based methods use a noisy channel approach towards DC hypothesizing that disfluent sentences are fluent sentences with noisy elements (Jamshid Lou and Johnson, 2017; Johnson and Charniak, 2004; Zwarts and Johnson, 2011). Parsing-based methods use techniques such as dependency parsing to predict syntactic structure of an utterance along with disfluent elements (Rasooli and Tetreault, 2015; Honnibal and Johnson, 2014;

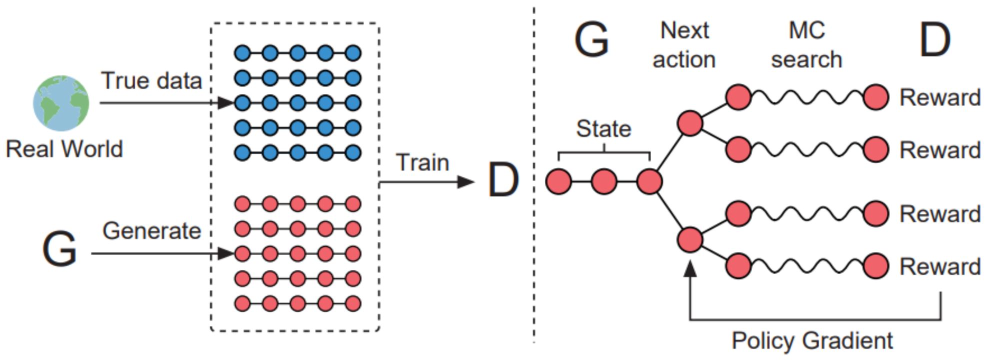

Figure 2: The illustration of SeqGAN. Left: D is trained over the real data and the generated data by G. Right: G is trained by policy gradient where the final reward signal is provided by D and is passed back to the intermediate action value via Monte Carlo search.

Wu et al., 2015). Sequence tagging methods work well for disfluency removal from real-life spoken utterances, assigning disfluent/fluent label to every word in the sentence (Hough and Schlangen, 2015; Ostendorf and Hahn, 2013; Zayats et al., 2016; Chen et al., 2022). There is a notable gap in literature regarding real data annotation in DC, with Switchboard (Godfrey et al., 1992) being the most extensive open-source labeled dataset for DC in English. Although Gupta et al. (2021) introduced a dataset for disfluencies in English question answering, they have not been annotated for disfluent words. Without labeled data, various zero-shot, few-shot, and multi-task learning techniques have been proposed, which train on multilingual data, creating and utilizing synthetically generated disfluent sentences (Wang et al., 2018; Passali et al., 2022; Kundu et al., 2022; Bhat et al., 2023). In this work, we experiment with sequence tagging methods for DC.

4 Types of Disfluencies

We study four types of disfluencies observed: Filler, Repetition, Correction and False Start. Additionally, there are some fluent sentences present in our datasets. Table 2 describes each type of sentence with some real examples.

5 Surface Structure of Disfluencies

Shriberg (1994) defines disfluencies as a composition of Reparandum, Interregnum and Repair (Figure 3). *Reparandum* refers to words erroneously uttered by the speaker. The speaker acknowledges that a previous utterance might be incorrect using *interregnum*, whereas *repair* contains words spoken to correct mis-spoken words. Disfluent utterances might consist of an interruption point- a spoken phenomena like speech pauses. DC removes reparandum and interregnum while retaining repair to make the output sentence more fluent.

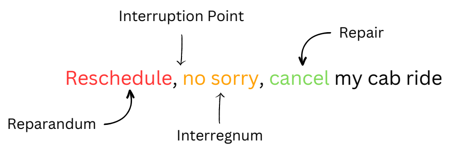

Figure 3: A disfluent utterance in English, marked with various components of disfluencies.

6 DISCO: A dataset for Disfluency Correction in Indo-European Languages

This section analyzes the DISCO corpus, created with the help of English, Hindi, German and French language experts. DISCO contains parallel disfluent-fluent sentence pairs in the above four languages and English translations of fluent sentences in Hindi and German along with disfluency and domain labels.

Disfluency Type	Description	Example
Filler	Words like *uhh*, *err*, *uhmm* that are often uttered to retain turn of speaking. Each language has a different set of filler words commonly uttered.	EN: Write a message to um Sarah. DE: Fortsetzen ähm meines Lauftrainings. FR: Montre euh mes applications. HI: मेरा उम्म पल्सरेट फिटबित में चेक करो
Repetition	Consists of words or phrases that are repeated in conversational speech	EN: Add this number to my to my contacts. DE: ein Instagram-Foto machen machen. FR: Enregistre mes 400 calories enregistre. HI: क्या तुम हॉस्पिटल का हॉस्पिटल का एक नोट बना सकते हो?
Correction	Disfluencies that consist of words incorrectly spoken and immediately corrected with a fluent phrase	EN: Get me the order my order status on the desk chair I ordered from Overstock. DE: HD Video auf aufnehmen. FR: Reprendre l'exercice d'étirem d'étirement HI: रा राहु राहुल का मैसेज पढ़ो
False Start	Examples where the speaker changes their chain-of-thought mid sentence to utter a completely different fluent phrase	EN: In an email let's email Tom Hardy about Saturday's video shoot. DE: Facebook uh Jahr Facebook bitte. FR: Envoi de le envoi du SMS à maman. HI: कल उम्म आज की ब्लड प्रेशर रीडिंग बताओ
Fluent	Examples which do not contain any disfluent words or phrases	EN: Can you make a note for Johnny that says dinner at eight on my laptop? DE: Nummer zu Kontakten hinzufügen.. FR: Je veux j'aimerais ouvrir TikTok.. HI: क्या आप योसेमाइट नेशनल पार्क को ईमेल कर सकते हैं?

Table 2: Types of sentences observed in the DISCO corpus. All disfluencies are marked in red; EN-English, DE-German, FR-French, HI-Hindi.

6.1 Data Collection Method

Goel et al. (2023) released an open-source dataset containing real-life utterances of humans with AI agents for task-oriented dialogue parsing. We extract disfluent sentences and domain labels in English, Hindi, German and French from this corpus. These utterances consist of human dialogues like making notes, monitoring fitness, adding new contacts, opening apps, etc. All sentences are shared with respective language experts for fluent sentence creation and disfluency-type annotation.

6.2 Annotation Protocol and Challenges

For each language, we hired external annotators from reputed translation agencies with experience in data annotation. They were asked to create fluent sentences corresponding to disfluent utterances along with disfluency type labels. Each annotator was paid competitively based on current market standards (approximately $ 0.018 per word). Since we follow a sequence tagging approach towards DC, the annotators were asked to only remove disfluent words from utterances without changing word order or correcting original words/phrases.

Due to budget constraints, we could not utilize the entire dataset in German and French from Goel et al. (2023). However, we carefully select sentences in these languages to sufficiently cover all disfluency types with varied length and complexity of utterances. Table 3 summarizes the total amount of data created and the amount of disfluency present in the corpus.

Lang	No. of sentence pairs	No. of words	% disfluent words
En	3479	31994	18.99
Hi	3180	32435	18.99
De	3096	22451	20.93
Fr	3005	22489	17.72

Table 3: Total count of disfluent-fluent pairs in DISCO and percentage of disfluency present; En-English, Hi-Hindi, De-German, Fr-French.

Since for every language, only one annotator created fluent sentences and disfluency type labels, ensuring high quality data was very important. We strongly encouraged the annotators to flag all dubious instances, after which the authors take a majority vote of retaining/removing doubtful disfluent words.

6.3 Key Statistics

The DISCO corpus is carefully created to ensure healthy representation of various disfluency types and complexity of sentences. Table 4 describes average length of disfluent and fluent sentences for each language. Our analysis shows that in similar context, commands to AI agents are shorter in German and French than in English and Hindi. The standard deviation of the disfluent sentences demonstrates that the dataset also contains longer utterances, more than ten words long, in each language that are relatively difficult to correct. We showcase the distribution of disfluent sentences across disfluency types in figure 4.

Lang	Mean length of disfluent sentences	Mean length of fluent sentences
En	9.19 ± 2.85	7.45 ± 2.59
Hi	10.18 ± 3.60	8.24 ± 3.12
De	7.25 ± 3.12	5.71 ± 2.84
Fr	7.42 ± 3.05	6.08 ± 2.87

Table 4: Average length of disfluent and fluent utterances in the DISCO corpus for each language; En-English, Hi-Hindi, De-German, Fr-French.

6.4 Domain Level Analysis

We use domain type labels from Goel et al. (2023). To better understand our data, we describe each domain type under broader categorization. In each example, disfluent utterances are marked in red. The number of sentences in each domain type for each language is specified in table 5.

- **Health & Fitness:** Sentence pairs belonging to this domain consist of utterances where the user wants to perform a fitness or health checkup task like recording his/her exercises, nutrition or blood sugar. Any interaction where the user discusses any fitness query can be tagged in this category. Domains such as Get health stats, Log exercise, Log nutrition, Start exercise, Stop exercise, Pause exercise and Resume exercise fall under this type.

  Example - Go to Fitbit and show me my um my blood sugar reading

- **Order Status:** Sentence pairs belonging to this domain consist of utterances where the user wants to check the status of the already placed order. The Check order status domain falls under this type.

  Example - Check the status of um of my Poshmark order with FedEx.

- **Finance :** Sentence pairs belonging to this domain consist of utterances where the user wants to perform a finance task like checking stock market prices or getting information from a finance app. The domain Get security price falls under this type.

  Example - I want to um check stock prices.

- **Bill Payment or Purchase:** Sentence pairs belonging to this domain consist of utterances where the user wants to complete a bill payment or is instructing the AI agent to purchase something for him/her. Domains such as Get bill, Pay bill, Get product, BuyEventTickets, GetGenericBusinessType, Order menu item fall under this type.

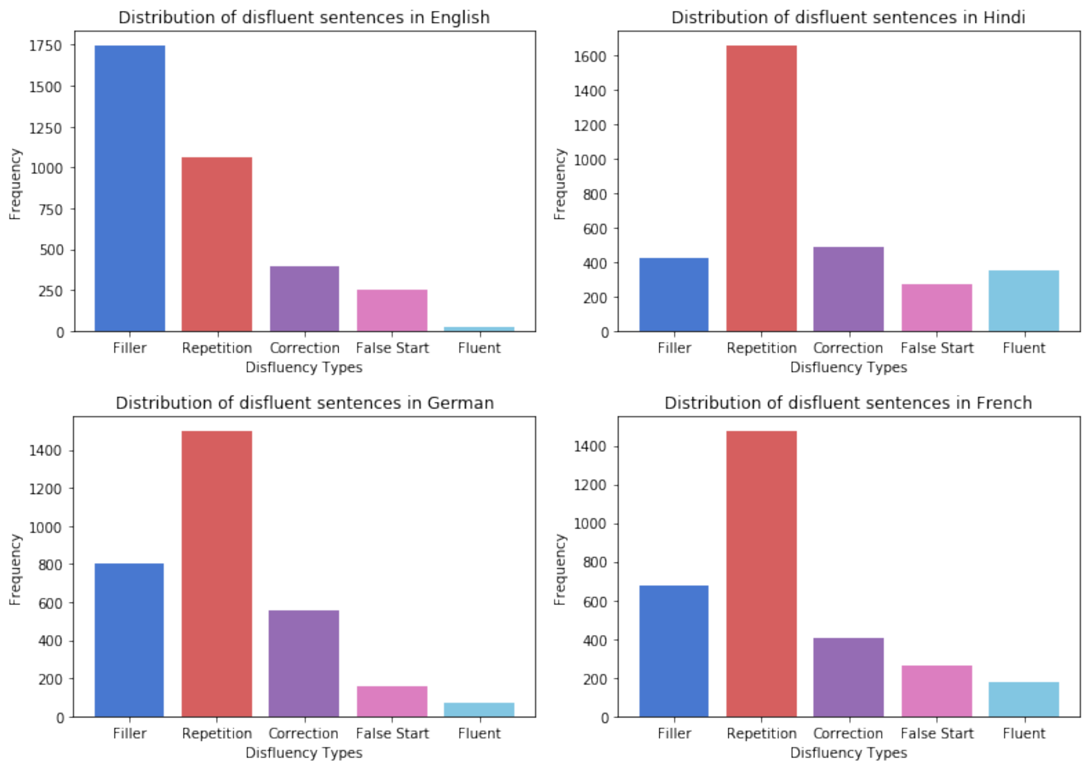

Figure 4: Distribution of sentences across disfluency types for all four languages in DISCO.

Example - Pay my um my phone bill for this month.

- **Internal Task:** Sentence pairs belonging to this domain consist of utterances where the user wants the AI agent to perform a task which does not involve any extra application. Examples could be sending a message/email to someone, cancelling some plan, taking some notes, etc. If a third-party application is used in the utterance, it is an ”External Task”; if not, it is an ”Internal Task”. Domains such as Get message content, Add contact, Create note, Open app, Take photo, Add item to list fall under this type.

  Example - I want to e-mail Zane this photo and cc um and cc Zach.

- **External Task:** Sentence pairs belonging to this domain consist of utterances where the user wants the AI agent to perform a task with the help of a third-party application. In this domain, you will find utterances where the user specifies the AI agent and which application should the AI agent use to complete the task. Domains such as Cancel ride, Order ride, Post message fall under this type.

  Example - Use WhatsApp to to send location to Jim.

6.5 Important plots

We also depict the word cloud of disfluent sentences across the four languages. Our analysis shows the most common disfluent words across four languages. Since the Filler class occupies a majority in the distribution, for each language, we see filler words like um, uh, er, and umm occupy a considerable size in the cloud for English. Similarly, common fillers in Hindi, German and French are the biggest in the respective word clouds (figure 5).

Correlation analysis between original Hindi and German sentences and their respective English translations was also performed to ensure that the number of outliers was minimum and the slope of points followed a natural straight line. Figure 6 depicts the straight-line scatter plots observed.

The disco corpus contains a good representation of shorter and longer disfluent sentences across each language, increasing the complexity of corrections needed. Figure 7 depicts the box plot of disfluent sentences, indicating the average sentence lengths of spoken utterances across four languages.

7 Summary

In this paper, we thoroughly discuss disfluency correction and the work done in CFILT lab over the last two years. We cover all necessary theory and terminologies in disfluencies including common types of disfluencies and its surface structure. We also provide a thorough literature review of all papers in disfluency correction. We motivate DC as a post processing step to ASR transcripts to improve machine translation accuracies. We introduce our dataset for Indo-European languages DC with details about disfluency types and some preliminary data analysis. This corpus is rich in many important disfluency phenomena and future work must focus on experimenting with this dataset using different types of techniques in DC.

References

Vineet Bhat, Preethi Jyothi, and Pushpak Bhattacharyya. 2023. Adversarial training for low-resource disfluency correction.

Angelica Chen, Vicky Zayats, Daniel Walker, and Dirk Padfield. 2022. Teaching BERT to wait: Balancing accuracy and latency for streaming disfluency detection. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 827–838, Seattle, United States. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

John J. Godfrey, Edward Holliman, and J. McDaniel. 1992. Switchboard: telephone speech corpus for research and development. *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1:517–520 vol.1.

Rahul Goel, Waleed Ammar, Aditya Gupta, Siddharth Vashishtha, Motoki Sano, Faiz Surani, Max Chang, HyunJeong Choe, David Greene, Kyle He, Rattima Nitisaroj, A. A. Trukhina, Shachi Paul, Pararth Shah, Rushin Shah, and Zhou Yu. 2023. Presto: A multilingual dataset for parsing realistic task-oriented dialogs. *ArXiv*, abs/2303.08954.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Aditya Gupta, Jiacheng Xu, Shyam Upadhyay, Diyi Yang, and Manaal Faruqui. 2021. Disfl-QA: A benchmark dataset for understanding disfluencies in question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3309–3319.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Matthias Honal and Tanja Schultz. 2003. Correction of disfluencies in spontaneous speech using a noisy-channel approach. *8th European Conference on Speech Communication and Technology (Eurospeech 2003)*.

Matthew Honnibal and Mark Johnson. 2014. Joint Incremental Disfluency Detection and Dependency Parsing. *Transactions of the Association for Computational Linguistics*, 2:131–142.

Julian Hough and David Schlangen. 2015. Recurrent neural networks for incremental disfluency detection.

Paria Jamshid Lou and Mark Johnson. 2017. Disfluency detection using a noisy channel model and a deep neural language model. In *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.

Eric Jang, Shixiang Shane Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *ArXiv*, abs/1611.01144.

Mark Johnson and Eugene Charniak. 2004. A tag-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL

Domain Type	English	Hindi	German	French
Send digital object	222	116	116	121
Get health stats	258	105	140	95
Get message content	191	68	69	100
Add contact	286	143	106	118
Create note	68	62	80	45
Check order status	274	150	90	72
Get bill	255	131	82	68
Get security price	238	117	85	68
Open app	238	156	185	226
Pay bill	251	128	109	105
Get product	218	94	116	98
Other	223	72	89	62
Post message	256	134	178	169
Record video	25	129	152	154
Log exercise	228	129	128	73
Log nutrition	248	88	98	87
Take photo	0	108	140	150
Cancel ride	0	191	109	170
Order ride	0	138	86	148
BuyEventTickets	0	98	118	85
Play game	0	115	133	151
GetGenericBusinessType	0	103	101	38
Start exercise	0	164	151	133
Stop exercise	0	172	170	164
Pause exercise	0	129	157	100
Resume exercise	0	140	108	165
Order menu item	0	0	0	25
Add item to list	0	0	0	15

Table 5: Language wise distribution of domain of disfluent sentences in DISCO corpus

'04, page 33–es, USA. Association for Computational Linguistics.

Simran Khanuja, Diksha Bansal, Sarvesh Mehtani, Savya Khosla, Atreyee Dey, Balaji Gopalan, Dilip Kumar Margam, Pooja Aggarwal, Rajiv Teja Nagipogu, Shachi Dave, Shruti Gupta, Subhash Chandra Bose Gali, Vish Subramanian, and Partha Talukdar. 2021. Muril: Multilingual representations for indian languages.

Rohit Kundu, Preethi Jyothi, and Pushpak Bhattacharyya. 2022. Zero-shot Disfluency Detection for Indian Languages. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4442–4454, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

M. Ostendorf and S. Hahn. 2013. A sequential repetition model for improved disfluency detection. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 2624–2628.

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

Sharath Rao, Ian Lane, and Tanja Schultz. 2007. Improving spoken language translation by automatic disfluency removal: evidence from conversational speech transcripts. In *Proceedings* *of Machine Translation Summit XI: Papers*, Copenhagen, Denmark.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

Elizabeth Shriberg. 1994. Preliminaries to a theory of speech disfluencies.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv:1706.03762 [cs]*. ArXiv: 1706.03762.

Feng Wang, Wei Chen, Zhen Yang, Qianqian Dong, Shuang Xu, and Bo Xu. 2018. Semi-supervised disfluency detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3529–3538, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Wen Wang, Gokhan Tur, Jing Zheng, and Necip Fazil Ayan. 2010. Automatic disfluency removal for improving spoken language translation. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5214–5217.

Shuangzhi Wu, Dongdong Zhang, Ming Zhou, and Tiejun Zhao. 2015. Efficient disfluency detection with transition-based parsing. In *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.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1).

Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. Disfluency detection using a bidirectional lstm. pages 2523–2527.

Simon Zwarts and Mark Johnson. 2011. The impact of language models and loss functions on repair disfluency detection. In *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.

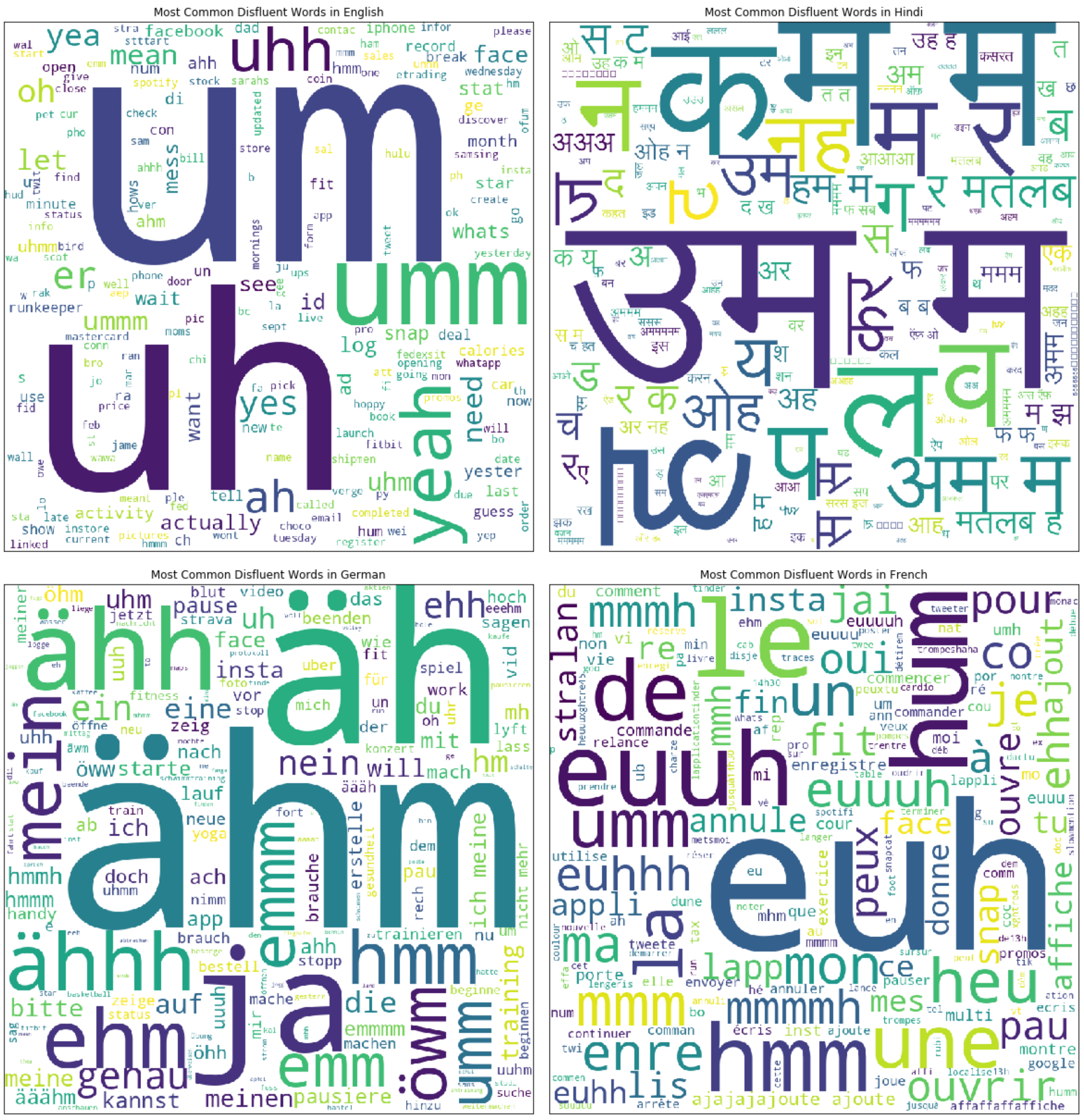

Figure 5: Word cloud of disfluent sentences across each language in the DISCO corpus, showcasing the most common disfluent words observed in spoken utterances

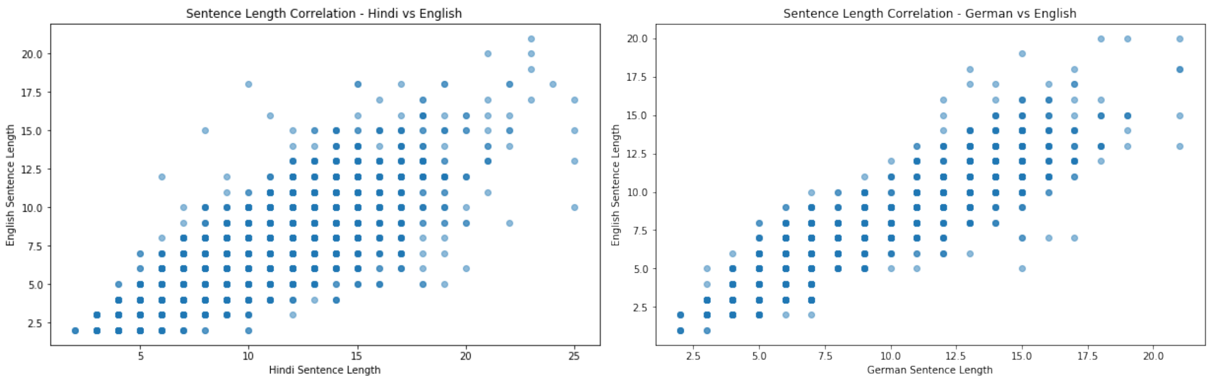

Figure 6: Plotting the correlation between disfluent sentences and their English translations. These graphs indicate that the number of words in any English translation of Hindi or German sentences can be estimated using a straight line slope as depicted with minimum outlier cases.

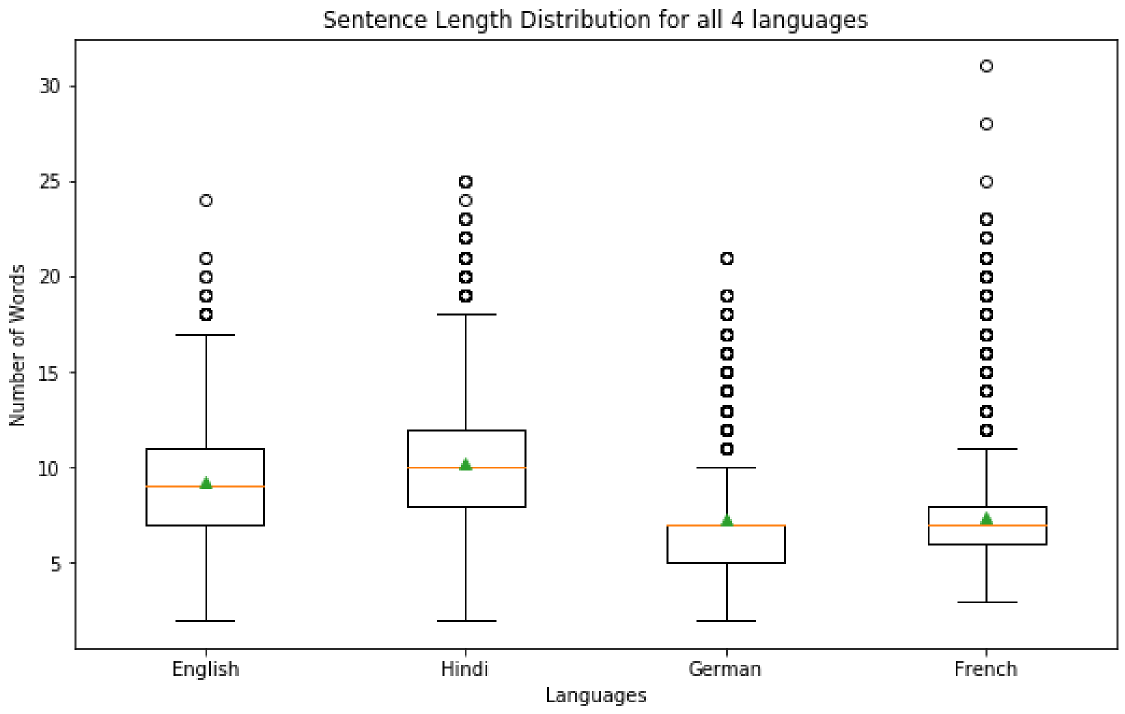

Figure 7: Box plot of disfluent sentence lengths across all languages in DISCO corpus