Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 2327–2336
Marseille, 20-25 June 2022

LARD: Large-scale Artificial Disfluency Generation

Tatiana Passali[*†], **Thanassis Mavropoulos**[*], **Grigorios Tsoumakas**[†], **Georgios Meditskos**[*†], **Stefanos Vrochidis**[*]

[*]Centre for Research and Technology Hellas, Thessaloniki, Greece
{tpassali, mavrathan, gmeditsk, stefanos}@iti.gr

[†]School of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece
{scpassali, greg, gmeditsk}@csd.auth.gr

Abstract

Disfluency detection is a critical task in real-time dialogue systems. However, despite its importance, it remains a relatively unexplored field, mainly due to the lack of appropriate datasets. At the same time, existing datasets suffer from various issues, including class imbalance issues, which can significantly affect the performance of the model on rare classes, as it is demonstrated in this paper. To this end, we propose LARD, a method for generating complex and realistic artificial disfluencies with little effort. The proposed method can handle three of the most common types of disfluencies: repetitions, replacements, and restarts. In addition, we release a new large-scale dataset with disfluencies that can be used on four different tasks: disfluency detection, classification, extraction, and correction. Experimental results on the LARD dataset demonstrate that the data produced by the proposed method can be effectively used for detecting and removing disfluencies, while also addressing limitations of existing datasets.

Keywords: disfluency detection, synthetic dataset, data augmentation

1. Introduction

Virtual assistants and spoken dialogue systems are increasingly used in many applications, such as automated customer service (Gnewuch et al., 2017; Sheehan et al., 2020), education (Clarizia et al., 2018; Heryandi, 2020) and healthcare (Tian et al., 2019; Sezgin et al., 2020). Recent systems, which are powered by powerful Deep Learning (DL) models (Golovanov et al., 2019; Wu et al., 2019; Majumder et al., 2020; Zhong et al., 2020; Zhao et al., 2020), led to significantly more accurate systems compared to previous approaches. However, despite the considerable progress in this area, many difficult problems still remain unresolved. One of them is the fluidity of spoken language, which often includes various errors and self-corrections made on the fly, rendering the deployment of such systems in the wild especially challenging.

Indeed, when humans recognize that they have made an error in their speech, they automatically attempt to correct it by editing, reformulating, or completely restarting themselves. This is part of the natural and intuitive process that happens subconsciously during a spontaneous human dialogue flow. These interruptions and self-corrections of a speaker's utterances are called *disfluencies* (Shriberg, 1994) and are a common trait of spontaneous speech. There is a large number of studies about the structure of disfluencies and their influence on the conversation between humans in theoretical level (Sparks, 1994; Shriberg, 1994; Plejert, 2004; Colman and Healey, 2011; Emrani and Hooshmand, 2019) which led to the development of various practical applications and models for disfluency detection (Zayats et al., 2016; Wang et al., 2016; Wang et al., 2020; Rocholl et al., 2021).

A practical virtual assistance system should be able to seamlessly handle these cases of disfluency and understand the actual intention of the user. This includes promptly detecting the disfluency, categorizing it, and then appropriately correcting the input to the system. However, despite its importance in these situations, disfluency detection remains a relatively unexplored field. This can be attributed to a large degree to the lack of appropriate datasets for training disfluency detectors. It is worth mentioning that Switchboard (Godfrey et al., 1992), the largest dataset in the literature that includes - as a side task - disfluency detection, contains less than 40,000 examples of disfluencies, with most of them (more than 50%) Shriberg (1996) belonging to the most trivial class (simple repetitions) (Zayats et al., 2016; Zayats et al., 2019). On the other hand, other datasets, such as Fisher (Cieri et al., 2004), may contain a larger amount of data, which however are not annotated. As a result, they cannot be easily used to develop a system that would meet the aforementioned requirements.

Therefore, the need for a dataset for fine-grained disfluency detection becomes obvious. However, manually collecting and annotating such datasets can be expensive, since it requires appropriately trained personnel. An option to overcome this limitation is to use theoretical knowledge regarding the structure of disfluencies to generate synthetic ones. However, this requires very carefully designed methods, since in many cases the difficulty and the variety of the synthetic examples can

be low, as highlighted in Gupta et al. (2021), which can negatively impact the performance of DL models.
Revisiting disfluency detection in the wild, we demonstrate that models trained on existing datasets fail to recognize simple types of disfluency. To overcome this limitation, we propose an automated method for generating a vast amount of synthetic disfluencies on English text based on existing textual corpora that contain only fluent text. More specifically, we start with a large-scale dialogue dataset that contains real dialogues. Then, we use advanced Natural Language Processing (NLP) tools that, along with carefully designed generation methods, ensure the production of realistic and coherent examples with high variety. The proposed method can handle the three most common types of disfluency (repetitions, replacements, and restarts) (Chalak et al., 2015; Emrani and Hooshmand, 2019), while it can also support a variety of different tasks. The code is publicly available[1].
The contribution of our work is summarised in the following:

- We propose a method for generating realistic and coherent disfluency datasets.
- We release a new dataset for disfluency detection based on the proposed method.
- We experimentally evaluate our framework in synthetic and real-world datasets using state-of-the-art NLP tools.

The rest of the paper is organized as follows. Disfluencies in human dialogue are discussed in Section 2. Related work is reviewed in Section 3. The proposed method is introduced in Section 4. Experimental results are provided in Section 5. Finally, conclusions and future research directions are discussed in Section 6.

2. Disfluencies in Human Dialogue

This section introduces the structure, as well as the different categories of disfluencies.

2.1. Structure of Disfluencies

Shriberg (1994) introduced a standard annotation scheme, called *reparandum/interregnum*, for identifying disfluencies. This annotation scheme involves the following fragments: a) the *reparandum*, b) the *interruption point*, c) the *repair*, and optionally d) the *interregnum*, which is located right before the repair.
The reparandum indicates the disfluent part of the utterance: the part that is not correct and must be replaced or ignored. Typically, the speaker attempts to rephrase, edit or restate the reparandum. The reparandum is usually short and involves 2 to 3 words. Even though the reparandum is usually considered as a "rough" copy of the repair, it can also be completely irrelevant with the repair.

[1]https://github.com/tatianapassali/artificial-disfluency-generation

$$Can\ we\ meet\ on\ \underbrace{Tuesday}_{\text{reparandum}}\ \overbrace{+}^{\substack{\text{interruption}\\ \text{point}}}\ \underbrace{I\ mean}_{\text{interregnum}}\ \underbrace{on\ Friday}_{\text{repair}}?$$

Figure 1: An example of a disfluency, where the reparandum, interruption point, interregnum and repair are illustrated.

The interruption point indicates the start of the repair, if no interregnum exists. If an interregnum exists, then the interruption point indicates the start of the interregnum, after which the repair follows. The interruption point does not indicate an actual word, it rather marks the moment of speech when a speaker realizes the error and initiates the correction process. In other words, the interruption point typically occurs right after the last word that is being told before the interruption of speech.
The interregnum consists of repair cues, such as those shown in Table 1, that are typically used to fill the gap between the disfluent speech and the associated repair. These types of fillers are generally ignored from dialogue systems as they do not contain any useful information. Most of the time, the presence of a repair cue indicates the presence of a disfluency in the utterance. Interregnums are relatively easy to be detected as they are usually fixed phrases, in contrast to reparandums that require a deeper understanding of the complex dialogue flow. For this reason, many methods ignore the interregnum and focus only on the detection of the reparandum and the repair (Zayats et al., 2016; Zayats et al., 2019; Jamshid Lou et al., 2018; Bach and Huang, 2019).

Table 1: Different types of repair cues as described byShriberg (1994)

Repair cues	**Examples**
filled pauses	um, uh
editing phrases	oops, no, sorry, wait, I meant to say
discourse markers	well, actually, okay, you know, I mean

We typically meet a repair in a speaker's utterance either after the presence of an interregnum or directly after the interruption point. The repair is the corrected fragment of speech and is usually different from the reparandum. However, in some cases the repair can be exactly the same as the reparandum or even completely empty.
An example of the aforementioned annotation scheme is described in Fig. 1. In this example, the speaker mistakenly says Tuesday instead of Friday. The disfluent part of this example consists of the words *"on Tues-*

day" (reparandum), which is followed by an interregnum (*"I mean"*) right after the interruption point and eventually by the correct choice of day (*"on Friday"*), that makes the particular repair. To simplify the presentation of disfluencies under this scheme, the following notation is typically used to indicate the different parts of disfluencies:

[reparandum + {interregnum} repair]

In this notation, the brackets ("[" and "]") are used to indicate the disfluency along with the repair. The interruption point is indicated by the plus ("+") symbol, while the interregnum, if present, is indicated by curly brackets ("{", "}").

2.2. Categories of Disfluencies

Disfluencies can be categorized into three distinct classes based on Shriberg (1994) speech repair typology: a) repetitions, b) replacements, and c) restarts, as described below:

- **Repetitions**: The speaker repeats a word, a phrase or a sequence of words. In repetitions, the reparandum and the corresponding repair are actually the same. This type of disfluency is the most common and is particularly easy to be detected (Zayats et al., 2014; Jamshid Lou et al., 2019; Zayats et al., 2019).
- **Replacements**: The speaker replaces the disfluent word(s) or phrase with the fluent one. In replacements, the reparandum is replaced by the repair.
- **Restarts**: The speaker abandons completely the initial utterance and restarts it. This type of disfluency does not actually involve a repair as the speaker begins a completely new utterance.

Some examples of the different categories of disfluencies are shown in Table 2.

Table 2: Different types of disfluencies annotated based on the reparandum/interregnum scheme.

Type	Example
repetition	Let's meet [today + today].
replacement	I want [the blue + {no} the red] one.
restart	[Why don't you +] I will do it later.

3. Related Work

The related work is divided into three broad categories: a) models and methodologies for detecting and classifying disfluencies, b) existing datasets that can be used for training the aforementioned methods and c) data augmentation methods that attempt to alleviate the issues arising from the nature of existing datasets (e.g., limited size and variety) for tackling these tasks.

3.1. Methodologies and Models

Current research in disfluency detection is based on four methodologies: *parsing-based*, *noisy-channel*, *sequence tagging* and *disfluency correction* approaches. Parsing-based approaches work by identifying both the syntactic structure and the disfluencies in a text sequence (Honnibal and Johnson, 2014; Rasooli and Tetreault, 2015; Wu et al., 2015; Yoshikawa et al., 2016; Jamshid Lou et al., 2019). Noisy-channel methods (Charniak and Johnson, 2001; Johnson and Charniak, 2004; Zwarts et al., 2010) compute the similarity between the reparandum and the repair of an utterance in order to indicate a disfluency. Jamshid Lou and Johnson (2017) combine a noisy-channel model to extract the best candidates for disluency analyses while a Long Short-Term Memory (LSTM) model is used to score these candidates and conclude to the most plausible one.

Sequence tagging models detect disfluencies in dialogues by predicting a tag/label to classify tokens as fluent or disfluent. Sequence tagging methodologies include Conditional Random Fields (CRF) (Georgila, 2009; Ostendorf and Hahn, 2013), Hidden Markov Models (HMMs) (Liu et al., 2006), Semi-Markov CRF (Ferguson et al., 2015), Auto-Correlation CNNs (Jamshid Lou et al., 2018), RNNs (Hough and Schlangen, 2015; Zayats et al., 2016; Wang et al., 2016), Transformers (Dong et al., 2019; Wang et al., 2020) and on-device lightweight BERT architectures (Rocholl et al., 2021).

Finally, a recent approach for disfluency correction, which was inspired by machine translation models, tackles disfluency detection as a sequence to sequence task where the disfluent sequence is "translated" into a clean and corrected version Saini et al. (2021).

3.2. Datasets

One of the most popular datasets that is used for training and evaluating the aforementioned models is Switchboard (Godfrey et al., 1992). Switchboard consists of 2,400 transcriptions of two-sided telephone conversations and approximately 190K utterances about a specific topic from a pre-defined list of 70 different topics. Switchboard is annotated with different types of speech tags and also contains metadata about some characteristics of the participants such as gender, education, area, etc. Disfluencies were annotated manually following the reparandum/interregnum structure (Shriberg, 1994) that it will be extensively described in Section 4.

Even though Switchboard is the most commonly used dataset for disfluency detection, only 5.9% tokens of the dataset are actually annotated as disfluent (Charniak and Johnson, 2001). In addition, more than 50% of the disfluency examples are classified as repetitions (Shriberg, 1996), which is considered as the easiest type of disfluency to identify (Zayats et al., 2016; Zayats et al., 2019).

Fisher English Training Transcripts (Cieri et al., 2004) corpus is another dataset that contains disfluencies and consists of 5,850 telephone conversations and approximately 1.3M utterances between two speakers, compiled by Linguistic Data Consortium (LDC)[2]. Each call, which lasts up to 10 minutes, is about an assigned topic, from a randomly generated list of topics. Some of the assigned topics are inherited from Switchboard corpus. Even though Fisher is clearly more extensive than Switchboard, it does not provide annotations for the existing disfluencies.

Last but not least, Gupta et al. (2021) recently released a new disfluency dataset for question answering of 12,000 human-annotated disfluent questions.

3.3. Data Augmentation

The existing DL methods for disfluency detection and classification critically rely on the quantity and quality of the annotated training data, which can be a limiting factor as discussed. Some limited steps to alleviate this problem have been made (Wang et al., 2018) using semi-supervised techniques leveraging knowledge from unlabeled datasets. Jamshid Lou et al. (2019) use a self-training augmentation technique to obtain silver annotations for the Fisher (Cieri et al., 2004) dataset using a trained model on the Switchboard (Godfrey et al., 1992) to augment the training data. Yang et al. (2020) use a simple planner-generator model in order to generate disfluent text, where a planner decides where to place the disfluent text which is created by the generator. A multi-task self-supervised learning method was proposed in Wang et al. (2020) where two self-supervised pre-training tasks were employed: a) a *tagging* task and b) a *sentence classification* task. At the same time, a pseudo-training dataset was created by randomly repeating, inserting or removing tokens from unlabeled sentences. The added tokens were randomly selected, thus the semantic consistency between the created disfluent sentence and the fluent one can not be guaranteed. Thus, unlike the proposed method, the aforementioned approaches either rely on very simple rules and techniques that are not capable of generating all different kinds of disfluencies that can be encountered in real scenarios. The proposed method is carefully designed to generate different types of disfluencies while preserving the semantic consistency between the fluent and the generated disfluent sentences, being closer to the disfluencies that can occur in a natural dialogue flow.

4. The LARD Method

This section describes our Large-scale ARtificial Disfluency (LARD[3]) method for generating disfluencies. In addition, it presents the large-scale dataset that we generated based on this method.

4.1. Generation Pipeline

LARD synthesizes disfluencies that belong to one of the three different types that were presented in Section 2.2: repetitions, replacements, and restarts. It is carefully designed to generate coherent and realistic synthetic examples, in order to ensure that the trained DL models will correctly generalize on real examples.

Algorithm 1 Repetition algorithm

Require: (A fluent sequence S_f, the degree of repetition $d_r \in \{1, 2, 3\}$)
Ensure: A disfluent sequence S_{dis}
1: **procedure** CREATE REPETITIONS(S_f, d_r)
2: $l_s \leftarrow length(S_f)$
3: $random_{idx} \leftarrow uniform(0, l_s - d_r)$
4: $S_{dis} \leftarrow$ Repeat the subsequence of tokens in S_f starting from index $random_{idx}$ to degree d_r.
5: **return** S_{dis}
6: **end procedure**

The approach used for generating repetitions is shown in Algorithm 1. More specifically, we select randomly an index from a sequence and we extend this sequence by repeating once the token in the corresponding index. Following this pattern, we create repetitions subcases for one, two and three consecutive words. For example, given a fluent sequence S_f = "I need to find a flight", a possible disfluent sequence that includes one-word repetition will be S_{dis} = "I need to [find + find] a flight" where the word "find" in position 4 is repeated. Similarly, given the same sequence for generating a two-word repetition, a possible generated disfluent sequence will be S'_{dis} = "[I need + I need] to find a flight", where the two consecutive words "I" and "need" are repeated.

Similarly, the approach used for generating replacements is shown in Algorithm 2. In particular, we extract all the nouns, verbs and adjectives with NLTK (Bird et al., 2009) POS tagger, for each sequence. We also create a list that contains different repair cues including all of the fixed words of Table 1, apart from filled pauses. We add some variations of these words and phrases such as "no, wait", "I am sorry", "no I meant to say", "no wait a minute", "well I actually mean" etc. We do not include filled pauses in this list because these types of repair cues can easily be identified and removed from recent dialogue generation models.

We create simple one-word replacements or replacements that involve more than one word in both reparandum and repair. The steps for *one-word replacement* generation are explained below:

1. Given a sequence, we extract all the available words that belong to the selected part-of-speech for replacement, if exist.

[2]https://www.ldc.upenn.edu/

[3]We can poetically view disfluencies as the *lard* that on one hand, you know, spices up human dialogues, but which on the other hand should be removed to improve their vigor.

Algorithm 2 Replacement algorithm

Require: (A fluent sequence S_f, part-of-speech $s_{pos} \in \{\text{noun, verb, adjective}\}$, a boolean variable cue

Ensure: A disfluent sequence S_{dis}

procedure CREATE REPLACEMENT(S_f, cue)
  if there is s_{pos} in S_f **then**
    $W_{pos} \leftarrow$ All the available words of the sequence S_f that belong to the selected input part-of-speech s_{pos}
    (a) Select randomly one of the available words from W_{pos} as a repair candidate.
    (b) Find all possible synonyms and antonyms for the selected repair candidate.
    $S_{repl} \leftarrow$ Select randomly an antonym or synonym as a reparandum candidate for the repair candidate
    $idx_{s_{pos}} \leftarrow$ index of the selected repair candidate in S_f
    $d_r \leftarrow uniform(0, idx_{s_{pos}})$ ▷ Define randomly the degree of replacement
    (c) Repeat d_r tokens before the $idx_{s_{pos}}$
    (d) Place the reparandum candidate
    if cue == True **then** Randomly select a repair cue from list and place it after the reparandum candidate.
    end if
    (e) Continue the rest of S_f.
  else
    Select another sequence.
  end if
end procedure

2. We randomly extract a repair candidate, a word whose POS is noun, verb or adjective, according to the desired POS.

3. We generate a list of possible synonyms and antonyms for the selected repair candidate word using WordNet (Bird et al., 2009).

4. We randomly select a synonym or antonym from the list as a reparandum candidate. The reparandum candidate (synonym or antonym) is placed in the reparandum right before the repair candidate. Note that we do not place it after the repair candidate in order to ensure that the fluent version of the sequence will be realistic. The selected repair candidate involves always only one word, however the reparandum candidate can vary from 1 to 4 words.

5. Optionally, we add randomly a word or phrase from the list of repair cues as interregnum depending on the sub-class that we generate (i.e., with or without repair cue).

6. Finally, the repair of the reparandum is the initially selected repair candidate.

For example, given a fluent sequence S_f = "Find me a different one", a possible disfluent sequence that includes an one-word replacement - the simplest form of a replacement - will be S_{dis} = "Find me a [same + {sorry} different] one". In this example, the selected repair candidate is the adjective "different" and the adjective "same" is generated as a possible antonym (reparandum candidate) for the selected repair candidate. Then, "same" is replaced by the adjective "different", which was initially in the fluent sequence. Note that between the replacement and the repair, we randomly place a repair cue (i.e., "sorry") from a fixed list with possible repair cues.

For replacements that involve *more than one word* in both reparandum and repair, we repeat the same steps, but we also include some previous tokens from the sequence before the reparandum candidate. To do that we randomly select a degree d_r for the replacement in order to define the number of tokens that will be repeated in the reparandum. The same tokens are repeated in the repair right before the selected repair candidate. For example, given another sequence S'_f = "I 'm looking for a salon in San Mateo" with degree $d_r = 1$ a possible generated disfluent sequence will be S'_{dis} = "I 'm looking for [a beauty shop + {I mean} a salon] in San Mateo". This is a more complicated example of the replacement class that involves more than one word in the reparandum. In particular, the noun "beauty shop" is generated as a synonym of the selected repair candidate "salon" and it is placed in the reparandum along with the previous token "a" ($d_r = 1$). Then, the subsequence "a beauty shop" is replaced by the subsequence "a salon", after the phrase "I mean", that is placed in the interregnum.

Eventually, the replacement class includes six different sub-classes: a) noun replacement with repair cue, b) noun replacement without repair cue, c) verb replacement with repair cue, d) verb replacement without repair cue, e) adjective replacement with repair cue and f) adjective replacement with repair cue.

Finally, Algorithm 3 is used for the generation of restarts. This algorithm assumes the existence of a fluent set that consists of two or more fluent sequences. First, we randomly select two sequences from the corresponding fluent set. Then, we split the first one in a random position and we combine the broken sequence together with the second unbroken one. For example, given a fluent sequence S_{f1} = "Do you want to check out on March 11th?" and a fluent sequence S_{f2} = "When is the check-out date?", a possible disfluent sequence that includes a restart will be S_{dis} = "[Do you want to +] When is the check-out date?". In this example, the initial fluent sequence S_{f1} is randomly broken in a position and the whole fluent sequence S_{f2} is placed right after the interruption point. Here, the repair is empty, since the speaker abandons the initial sequence and begins a new one. During this process, we make sure that we do not select two same sequences

Algorithm 3 Restart algorithm

Require: Two fluent sequences S_{f1} and S_{f2}
Ensure: A disfluent sequence S_{dis}
procedure CREATE RESTARTS(S_{f1}, S_{f2})
  if $S_{f1} \neq S'_{f2}$ **then**
    $S'_{f1} \leftarrow S_{f1}$ broken in a random position.
    if $S'_{f1} \neq$ beginning of S_{f2} **then**
      $S_{dis} \leftarrow join(S'_{f1}, S_{f2})$
    end if
  else
    Pick another sequence S_{f2}
  end if
  return S_{dis}
end procedure

and that the beginning of the second sequence is different from the first broken sequence. This avoids the accidental generation of repetition examples, instead of restarts.

4.2. Dataset Generation

To construct artificial disfluencies based on LARD, we use an existing large-scale dialogue dataset: the Schema-Guided Dialogue (SGD) dataset (Rastogi et al., 2019). SGD is a multi-domain task-oriented dataset that contains approximately 18K conversations between users and virtual agents with more than 300K total individual utterances. It covers a large number of different domains including conversations for various typical scenarios, such as hotel reservation, flight booking, finding a movie and weather prediction.

SGD is selected as a basis for our synthetic dataset for two main reasons. First, it contains a large number of available dialogues that can be used to generate synthetic disfluencies with a large variety. This is especially critical given the data-demanding nature of DL models that rely significantly on annotated data. Second, the existence of different multi-domain conversations in SGD allows for training more robust models that perform well on different tasks and settings.

Dataset generation is summarized as follows. First, we extract all the conversations from the SGD training set and split each conversation into sequence-level. Then, the total number of examples is split into four equal parts. The first part of the dataset remains unaltered, in order to include fluent sequences in our synthetic dataset. The others three parts are used for generating repetitions, replacements and restarts, respectively.

After applying this process on the SGD dataset, a total of 95,992 disfluencies were generated. Table 3 presents the statistics of the generated data. The LARD dataset is balanced among the four different possible classes (fluency, replacement, repetition and restart). We split the dataset into 57,595, 19,198 and 19,199 examples for training, validation and testing, respectively. The dataset is publicly available at Zenodo[4].

[4]https://zenodo.org/record/6451984

Table 3: Statistics of the LARD dataset.

Dataset Statistics	
# repetitions (one word)	8035
# repetitions (two words)	7964
# repetitions (three words)	7999
# repetitions	**23998**
# noun replacements	5023
# noun replacements + repair cue	4900
# verb replacements	4910
# verb replacement + repair cue	4814
# adjective replacements	2098
# adjective replacements + repair cue	2253
# replacements	**23398**
# restarts	**23998**
# fluencies	**23998**
# total examples	**95992**

We create two different types of annotations. At the sentence-level, we define four different classes: fluency, repetition, replacement and restart. We also mark each token as fluent (repair or any other word outside the reparandum and interregnum) or disfluent (any word inside reparandum and interregnum).

5. Empirical Evaluation

We present evaluation results on the proposed dataset in four different disfluency tasks: a) disfluency detection, b) disfluency classification, c) disfluency extraction, and d) disfluency correction. For disfluency classification, we consider all four classes (fluency, repetition, replacement and restart), while for disfluency detection we merge the repetition, replacement and restart classes into a disfluency class. For disfluency extraction, we classify each token of the sequence as fluent or disfluent. For disfluency correction, we use the synthetic sequence as input and the original fluent sequence as the desired output. Finally, we examine the behavior of pre-trained models fine-tuned on existing state-of-the-art datasets for disfluency detection like Switchboard. We demonstrate that even though models fine-tuned on Switchboard are particularly successful in detecting disfluencies like repetitions, they actually fail to recognize more complex types such as restarts and replacements.

5.1. Experimental Setup

For all the experiments we use Transformer-based models from the Hugging Face library (Wolf et al., 2020). For the first three tasks, we use the pre-trained BERT base uncased model (Devlin et al., 2019), which consists of 12 encoder layers with 12 attention heads. This model is trained on large lower-cased English corpora: English Wikipedia and BookCorpus (Zhu et al., 2015). We fine-tune this model on the LARD dataset

Table 4: Experimental results on models fine-tuned on LARD dataset. We report precision (prec), recall (rec) and f-measure (f1) (%) for the classification tasks and BLEU (%) for the correction task.

LARD dataset	Prec	Rec	F1	BLEU
Detection	97.63	97.61	97.62	-
Classification	97.31	97.30	97.29	-
Extraction	98.12	96.6	97.30	-
Correction	-	-	-	86.48

Table 5: Experimental results of models fine-tuned on Switchboard.

Switchboard	Prec	Rec	F1	BLEU
Detection	94.44	94.31	94.37	-
Extraction	90.02	86.91	88.44	-
Correction	-	-	-	71.49

for each task by adding a binary, multi-class and sequence classification head, respectively. For disfluency correction, we use T5-base (Raffel et al., 2020), which is a sequence-to-sequence model for text generation, with 12 encoder and decoder layers and 12 attention heads trained on Colossal Clean Crawled Corpus (C4). We set the learning rate to 0.00005 and fine-tune the pre-trained models for 20,000 steps with 16 batch size. Note that we do not extensively fine-tune these models, as this lies beyond the scope of this paper. Further fine-tuning and more extensive hyper-parameter searching could potentially lead to fine-tuned models with improved accuracy.

5.2. Experimental Results

The results for all the models on the four different tasks fine-tuned on the LARD dataset are shown in Table 4. For all the classification tasks, we report precision, recall and f-measure, while for disfluency correction, we report BLEU (Papineni et al., 2002). We can see that after fine-tuning on existing pre-trained models, we can easily reach a high accuracy on the LARD test set detecting, identifying and removing successfully different types of disfluencies.

We also examine the behavior of models with the same architecture fine-tuned on the Switchboard dataset (Godfrey et al., 1992; Zayats et al., 2019), following the common train/dev/test splits, as they are established from Charniak and Johnson (2001). The experimental results reported in Tables 4 and 5 reveal that models trained on Switchboard perform worse (on the corresponding test set) compared to models trained on LARD. This observation led us to further examine the behavior of models fine-tuned on Switchboard using the LARD test set.

The experimental results are reported in Table 6, where the accuracy for different disfluency classes is reported.

Table 6: Accuracy (%) for different disfluency classes (repetitions, replacements and restarts) and models trained on different datasets. Switchboard dataset supports only detection task. LARD method supports both detection and classification tasks.

	Switchboard (detection)	LARD (detection)	LARD (classification)
Repetitions	85.42	99.57	99.50
Replacements	54.52	99.67	98.39
Restarts	19.6	95.08	93.89

Even though models fine-tuned on Switchboard can achieve high accuracy in the repetition class, their performance significantly degenerates in the replacement ($\sim$55%) and restart classes ($\sim$20%). On the contrary, the model fine-tuned with the proposed method on the detection task achieves over 95% accuracy on all classes, while the one fine-tuned on the more challenging classification task, achieves over 93% accuracy on all classes. This finding can be attributed to the large number of repetitions included in Switchboard (Shriberg, 1996), along with the significantly smaller number of replacements and restarts. This finding also demonstrates the potential of the LARD method for generating complex disfluencies, allowing for increasing models' performance and robustness compared to existing and more imbalanced datasets.

6. Conclusion

We proposed LARD, a carefully designed mechanism for generating large-scale synthetic disfluencies from fluent text. We also compiled a large-scale dataset based on the proposed method which consists of approximately 96K examples, building upon an existing dialogue dataset: SGD (Rastogi et al., 2019). The generation pipeline is carefully designed in order to create complex, yet realistic and coherent disfluencies from fluent sequences. Experiments on the generated synthetic dataset using the LARD method on four different tasks (disfluency detection, classification, extraction and correction) showed that fine-tuned models can be effectively used to detect, identify and remove disfluencies from disfluent sequences. Furthermore, the thorough analysis conducted on models fine-tuned both on the generated dataset and Switchboard dataset revealed a significant weakness of the latter, arising from the imbalanced number of training examples among different classes, which can be successfully addressed using the proposed method.

Several interesting future work directions exist, including extending the proposed method to generate multiple disfluencies for training models that can handle these more difficult cases. Also, the proposed method highlights another research direction for modeling other linguistic phenomena, such as coreference resolution (Lee et al., 2018) and discourse relations (Qin et al., 2017), e.g., contrast, elaboration,

clarification, and others, providing an efficient way for generating highly realistic datasets that include them, based on existing large-scale fluent datasets.

7. Acknowledgements

This work is partially funded by the European Commission as part of its H2020 Programme, under the contract number 870930-IA (WELCOME).

8. Bibliographical References

Bach, N. and Huang, F. (2019). Noisy biLSTM-based models for disfluency detection. In *INTERSPEECH*.

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

Chalak, A., Talebi, A., Khodaeian, N., Pourakbari, A., and Danesh, J. (2015). Replacement operation in self-initiated repair practices in oral reproduction of short stories. *International Journal of Language Learning and Applied Linguistics World*, 8(1):237–246.

Charniak, E. and Johnson, M. (2001). Edit detection and parsing for transcribed speech. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

Cieri, C., Miller, D., and Walker, K. (2004). The fisher corpus: a resource for the next generations of speech-to-text. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May. European Language Resources Association (ELRA).

Clarizia, F., Colace, F., Lombardi, M., Pascale, F., and Santaniello, D. (2018). Chatbot: An education support system for student. In *International Symposium on Cyberspace Safety and Security*, pages 291–302. Springer.

Colman, M. and Healey, P. (2011). The distribution of repair in dialogue. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (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, June. Association for Computational Linguistics.

Dong, Q., Wang, F., Yang, Z., Chen, W., Xu, S., and Xu, B. (2019). Adapting translation models for transcript disfluency detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6351–6358.

Emrani, F. and Hooshmand, M. (2019). A conversation analysis of self-initiated self-repair structures in advanced iranian efl learners. *Online Submission*, 13(1):57–76.

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

Georgila, K. (2009). Using integer linear programming for detecting speech disfluencies. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, NAACL-Short '09, page 109–112, USA. Association for Computational Linguistics.

Gnewuch, U., Morana, S., and Maedche, A. (2017). Towards designing cooperative and social conversational agents for customer service. In *ICIS*.

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

Golovanov, S., Kurbanov, R., Nikolenko, S., Truskovskyi, K., Tselousov, A., and Wolf, T. (2019). Large-scale transfer learning for natural language generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6053–6058.

Gupta, A., Xu, J., Upadhyay, S., Yang, D., and Faruqui, M. (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, Online, August. Association for Computational Linguistics.

Heryandi, A. (2020). Developing chatbot for academic record monitoring in higher education institution. In *IOP Conference Series: Materials Science and Engineering*, volume 879, page 012049. IOP Publishing.

Honnibal, M. and Johnson, M. (2014). Joint incremental disfluency detection and dependency parsing. *Transactions of the Association for Computational Linguistics*, 2:131–142.

Hough, J. and Schlangen, D. (2015). Recurrent neural networks for incremental disfluency detection. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 849–853. ISCA.

Jamshid Lou, P. and Johnson, M. (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 Computa-*

tional Linguistics (Volume 2: Short Papers), pages 547–553, Vancouver, Canada, July. Association for Computational Linguistics.

Jamshid Lou, P., Anderson, P., and Johnson, M. (2018). Disfluency detection using auto-correlational neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4610–4619, Brussels, Belgium, October-November. Association for Computational Linguistics.

Jamshid Lou, P., Wang, Y., and Johnson, M. (2019). Neural constituency parsing of speech transcripts. 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 2756–2765, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Johnson, M. and Charniak, E. (2004). A tag-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, page 33–es, USA. Association for Computational Linguistics.

Lee, K., He, L., and Zettlemoyer, L. (2018). Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 687–692, New Orleans, Louisiana, June. Association for Computational Linguistics.

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.

Majumder, B. P., Li, S., Ni, J., and McAuley, J. (2020). Interview: Large-scale modeling of media dialog with discourse patterns and knowledge grounding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8129–8141.

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

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

Plejert, C. (2004). *To fix what's not broken: Repair strategies in non-native and native English conversation*. Ph.D. thesis, Linköping University Electronic Press.

Qin, K., Wang, L., and Kim, J. (2017). Joint modeling of content and discourse relations in dialogues. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 974–984, Vancouver, Canada, July. Association for Computational Linguistics.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Rasooli, M. S. and Tetreault, J. (2015). Yara parser: A fast and accurate dependency parser. *arXiv preprint arXiv:1503.06733*.

Rastogi, A., Zang, X., Sunkara, S., Gupta, R., and Khaitan, P. (2019). Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *arXiv preprint arXiv:1909.05855*.

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

Saini, N., Trivedi, D., Khare, S., Dhamecha, T., Jyothi, P., Bharadwaj, S., and Bhattacharyya, P. (2021). Disfluency correction using unsupervised and semi-supervised learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3421–3427.

Sezgin, E., Huang, Y., Ramtekkar, U., and Lin, S. (2020). Readiness for voice assistants to support healthcare delivery during a health crisis and pandemic. *NPJ Digital Medicine*, 3(1):1–4.

Sheehan, B., Jin, H. S., and Gottlieb, U. (2020). Customer service chatbots: Anthropomorphism and adoption. *Journal of Business Research*, 115:14–24.

Shriberg, E. (1994). *Preliminaries to a Theory of Speech Disfluencies*. University of California, Berkeley.

Shriberg, E. (1996). Disfluencies in switchboard. In *Proceedings of international conference on spoken language processing*, volume 96, pages 11–14. Citeseer.

Sparks, R. B. (1994). *The structure of self-repair in English conversation*. Ph.D. thesis, University of Colorado at Boulder.

Tian, S., Yang, W., Le Grange, J. M., Wang, P., Huang, W., and Ye, Z. (2019). Smart healthcare: making medical care more intelligent. *Global Health Journal*, 3(3):62–65.

Wang, S., Che, W., and Liu, T. (2016). A neural attention model for disfluency detection. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 278–287, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Wang, F., Chen, W., Yang, Z., Dong, Q., Xu, S., and Xu, B. (2018). Semi-supervised disfluency detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3529–

3538, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

Wang, S., Che, W., Liu, Q., Qin, P., Liu, T., and Wang, W. Y. (2020). Multi-task self-supervised learning for disfluency detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9193–9200, Apr.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October. Association for Computational Linguistics.

Wu, S., Zhang, D., Zhou, M., and Zhao, T. (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, July. Association for Computational Linguistics.

Wu, W., Guo, Z., Zhou, X., Wu, H., Zhang, X., Lian, R., and Wang, H. (2019). Proactive human-machine conversation with explicit conversation goals. *arXiv preprint arXiv:1906.05572*.

Yang, J., Yang, D., and Ma, Z. (2020). Planning and generating natural and diverse disfluent texts as augmentation for disfluency detection. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1450–1460.

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

Zayats, V., Ostendorf, M., and Hajishirzi, H. (2014). Multi-domain disfluency and repair detection. In *INTERSPEECH*.

Zayats, V., Ostendorf, M., and Hajishirzi, H. (2016). Disfluency detection using a bidirectional lstm. In *INTERSPEECH*.

Zayats, V., Tran, T., Wright, R. A., Mansfield, C., and Ostendorf, M. (2019). Disfluencies and human speech transcription errors. In Gernot Kubin et al., editors, *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 3088–3092. ISCA.

Zhao, X., Wu, W., Xu, C., Tao, C., Zhao, D., and Yan, R. (2020). Knowledge-grounded dialogue generation with pre-trained language models. *arXiv preprint arXiv:2010.08824*.

Zhong, P., Zhang, C., Wang, H., Liu, Y., and Miao, C. (2020). Towards persona-based empathetic conversational models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6556–6566, Online, November. Association for Computational Linguistics.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Zwarts, S., Johnson, M., and Dale, R. (2010). Detecting speech repairs incrementally using a noisy channel approach. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1371–1378, Beijing, China, August. Coling 2010 Organizing Committee.