A Phrase-Level Machine Translation Approach For Disfluency Detection Using Weighted Finite State Transducers

Sameer Maskey [†], Bowen Zhou, Yuqing Gao

[†]Columbia University, New York, NY 10021
IBM T.J Watson Research Center, Yorktown Heights, NY 10598
smaskey@cs.columbia.edu, zhou@us.ibm.com

Abstract

We propose a novel algorithm to detect disfluency in speech by reformulating the problem as phrase-level statistical machine translation using weighted finite state transducers. We approach the task as translation of noisy speech to clean speech. We simplify our translation framework such that it does not require fertility and alignment models. We tested our model on the Switchboard disfluency-annotated corpus. Using an optimized decoder that is developed for phrase-based translation at IBM, we are able to detect repeats, repairs and filled pauses for more than a thousand sentences in less than a second with encouraging results.

Index Terms: disfluency detection, machine translation, speech-to-speech translation.

1. Introduction

Disfluency is common in speech. Detecting disfluency in speech can be useful for readability of speech transcripts as well as for further processing by natural language models such as summarization, machine translation or parsing. We are interested in the removal of disfluency for improving a speech-to-speech translation system.

We follow the definition of Shriberg [16] and divide disfluency into three main components: reparandum (the words that are repaired), interregnum (filler words or filled pauses) and resumption (the new set of words that repair the reparandum). In this paper we detect three types of disfluencies: **repeats** (reparandum edited with the same sequence of words), **repairs** (reparandum edited with different sequence of words) and **filled pauses** (words in interregnum region).

Repeats	**I want to buy three glasses * three glasses of tea**
Repairs	**I want to buy three glasses * no five cups of tea**
Fillers	**I want to buy three glasses * umm four glasses please**

Table 1: Example of Disfluencies

Fillers are placed at the interruption point of the speaker's turn. Fillers include filled pauses such as 'um', 'uh', 'well'; discourse markers such as 'well', 'then', 'you know' and editing terms. The filled pauses may serve to signal hesitation or confusion in the speaker or to signify change in a given topic of conversation depending on the type of filled pause a speaker uses. In Table 1 'umm' is a filled pause placed at the interruption point '*'.

Repeats are one of the most common types of disfluencies. In the above example in Table 1, 'three glasses' is a repeat. Any such occurrences of verbatim repetition of a portion of a spoken utterance are 'repeats'.

Repairs may signify confusion in the speaker. In the above example in Table 1, the speaker is confused if he/she wants to order 'three glasses' or 'five cups' of tea. The phrase 'three glasses' is reparandum, which is repaired with 'five cups' after the interruption point. Repairs may also signify hesitation of the speaker.

In this paper we describe our method for detecting filled pauses and reparandum region of repeats and repairs. We describe related work in section 2 and our approach in section 3. In section 4 we describe our experiments and results and we conclude in section 5.

2. Related Work

There has been a significant amount of work in disfluency detection [11, 12, 3, 9, 5, 7, 14, 13]. Some of the disfluency detection systems have been built pertaining to DARPA EARS Rich Transcription program. Most of the disfluency detection systems that have been proposed use combinations of prosodic and lexical features though some systems are lexically driven [14] without any use of acoustic features. Snover et. al. [14] rely exclusively on word based lexical information and they have shown that a reasonable performance can be obtained without using acoustic features. Johnson and Charniak [6] use Tree Adjoining Grammar (TAG) noisy channel model and [3] focus on similarity and the length of the repeated sequence of words in the ASR transcripts.

On the other hand Nakatani and Hirschberg [12] have shown the advantages of using acoustic/prosodic features. They successfully detected interruption points (IP) by building a decision tree with acoustic features. Shriberg et. al [15] also proposed a method of detecting IPs by building a decision tree model based on prosodic features only. [17] improved this system with the addition of hidden event language model (LM) to detect boundaries and various types of disfluencies.

The addition of prosodic features to word based features has some clear advantages. For example, usually the intonation of a speaker is disrupted at the interruption point that indicates some form of restart. Such information is useful and has been shown to be significant [9, 10]. Another advantage of using prosodic features is its utility in disfluency detection for languages that lack adequate natural language tools.

Liu et. al [10, 11] successfully combined lexical and prosodic features and detected the onset of reparandum. Even though the use of combined lexical and prosodic features has some clear advantages, it should be noted that the prosodic features are not always easily available for some specific applications. Especially for online systems such as speech-to-speech translation any additional delay added for extra processing of speech signal to obtain various acoustic features may degrade the overall user experience. Therefore, in this paper, we will focus on describing the proposed framework using only lexical features.

3. Approach

We can view the disfluency removal problem as a process that transforms the "noisy" disfluent transcript into a "clean" one[4, 5]. Such a transformation can be described using statistical machine translation models. Particularly, Zhou et. al [19] has formulated a fast phrase-based statistical translation using FST's for a speech-to-speech (S2S) translation. We are motivated to apply a similar framework as [19] to address disfluency detection.

There are several advantages of the proposed scheme. This approach enables disfluency component to be easily integrated with other FST-based components such as machine translation engine by simple FST composition operation providing a unified search space for **disfluent speech translation**. Secondly, the proposed approach obtains very high speed and memory efficiency using the optimized decoder for phrase-based statistical machine translation [19], which is several orders of magnitude faster than the conventional classification based approaches.

3.1. Translation Model

Based on a source channel model approach to statistical machine translation, translating [1] a foreign token sequence n_1^J to a target token sequence c_1^I can be viewed as a stochastic process of maximizing the joint probability of $p(n, c)$ as stated in the equation 1

$$\hat{c} = argmax_{c_1^I} Pr(n_1^J, c_1^I) \tag{1}$$

The joint probability can be obtained by summing over all the hidden variables that can be approximated by maximization. For machine translation purposes these random variables take account of alignment, permutation, and fertility models.

For our purpose we view disfluency detection as translation from noisy token sequence $n_1^J := n_1, n_2, ..., n_J$ to a clean token sequence $c_1^I := c_1, c_2, ..., c_I$. Since the removal of disfluency will entail removal of words from n_1^J we still require alignment and fertility models as $I < J$.

We simplify the training of our translation model by retokenizing the c_1^I sequence. Instead of clean speech transcript without any disfluent words, we append a tag that signifies the type of disfluency for each disfluent word in n_1^J. This retokenization produces c_1^I with the same number of words as n_1^J such that $I = J$. The retokenization of our previous example of repair in Table 1 produces the following parallel text.

- Noisy Data: I want to buy three glasses no five cups of tea
- Clean Data: I want to buy REPAIR0 REPAIR1 FP0 five cups of tea

These modifications to the standard machine translation model simplify our model in the following ways: i) We do not require fertility model since the number of words in clean and disfluent speech are equal and words in noisy speech transcript can neither go to null nor generate more than one word. ii) With disfluent words retokenized ($I = J$) we have a perfect alignment between the noisy and clean transcripts in the parallel corpora, removing the need of alignment model.

The above approaches for simplifying our translation model reduce the data sparsity problems that are abound in machine translation methods. Instead of removing disfluent words all together, the model produces tags describing the type of disfluency that may be useful for further processing by other natural language processing modules such as detecting intonational boundaries.

3.2. Phrase Level Translation

Repeats and Repairs are difficult to detect because reparandum of these disfluencies can be more than one word. In our example in Table 1 the reparandum is "three glasses" - a two word phrase. Reparandum phrase can be of any length though phrases longer than five words are very unlikely. Word-based disfluency detection algorithms have difficulty in detecting such disfluent phrases because the classifier not only has to classify words as disfluent or not but also has to detect the start and end boundaries of the reparandum. This added complexity in repeat and repair detection can be addressed if we assume that disfluency occurs in phrases. Since we define phrase as a sequence of one or more words, single word disfluency are also addressed with such phrase assumption.

In order to detect repairs and repeats at the phrase level we build a phrase level translation model. Our phrase level translation model is built in a process that is identical to the one described in Zhou et. al [19]. The methods include techniques for phrase pair extraction, the phrase translation model estimation, and the WFST implementations. The process also includes techniques to determinize and minimize the transducers to optimize the search procedure. Since our bi-text alignment is a perfect one-to-one mapping, the phrase pair extraction procedure in this study is straightforward, and the only variable to consider is the phrase length limit.

The length limit on the phrase size makes a significant difference in the size of the dictionary. We chose a maximum phrase size of five as 99.9% of the disfluent phrase were smaller than five words in our training corpus.

We denote the phrase segmentation by introducing a hidden variable p_1^K to the Eq. 2 summing over the joint probability. In addition, we can approximate the sum over the hidden variables using a maximum operator.

$$\hat{c} = argmax_{c_1^J} \sum_{p_1^K} Pr(p_1^K, n_1^J, c_1^J) \tag{2}$$

$$\approx argmax_{c_1^J} max_{p_1^K} Pr(p_1^K, n_1^J, c_1^J) \tag{3}$$

3.3. Weighted Finite State Transducer Implementation

We can implement our equation 2 using weighted finite state transducers. Using the chain rule we can easily decompose the joint probability into a chain of condition probabilities as follows, in a similar way to [20, 19]:

$$Pr(p_1^K, n_1^I, c_1^I) = P(c_1^J). \tag{4}$$

$$P(p_1^K | c_1^J). \tag{5}$$

$$P(n_1^I | p_1^K, c_1^J) \tag{6}$$

We can compute the conditional probabilities of equations 4, 5 and 6 by using the parallel corpus and the phrase dictionary. Furthermore, we can build WFST for each probability distribution modeling the input and output - L, N and P where L is a language model, N is the translation model and P is the phrase segmentation model respectively.

The arc probabilities for the translation model N are computed by computing the relative frequencies from the collected phrase pairs.

$$P(c|n) = \frac{N(c,n)}{N(c)} \tag{7}$$

where $N(c,n)$ is the number of times a clean phrase c is translated by a noisy phrase n. The above equation overestimates the probabilities of rare phrases. In order to take account of such overestimation we smoothen our translation probability by performing a delta smoothing. We add a small numerical quantity δ on the numerator 7 and add $\delta.|V|$ on the denominator where V is the size of the translation vocabulary for a given phrase.

The language model plays an important role in a source channel model like ours. Our language model L is a standard trigram language model with the n-gram probability computed from the clean corpus that has disfluent words tagged as REPEAT, REPAIR and FP (filled pauses). In other words, we use the annotated side of the parallel corpus as the language model training data. We built a back-off 3-gram language model, and encoded it as a weighted acceptor as described in [19] to be employed by our translation decoder.

After building all three types of WFSTs we can perform a *cascaded composition* of these finite state transducers to obtain one translation lattice that translates sequence of noisy words to a clean phrases.

$$T = P \circ N \circ L \tag{8}$$

3.4. Decoding

One of the reasons why WFST-based approaches are attractive is due to the availability of efficient algorithms for decoding and optimization. In this framework we are able to combine heterogeneous statistical information with a composition operation of transducers representing the different knowledge sources. In our case such cascaded composition generates a lattice that can segment and translate phrases in a globally optimal framework.

For decoding, we employed the decoder that is developed in [19] using a multilayer search algorithm. Specifically, we have one layer for each of the input FSMs: the noisy input/acceptor, the translation model lattice, and the language model lattice. At each layer, the search process is performed via a state traversal procedure starting from the start state sr0, and consuming an input word in each step in a left-to-right manner. This can be viewed as an optimized version of on-the-fly or dynamic composition. However, specialized versions of composition have the advantage of not only having the potential of being many times faster than general composition implementations found in FSM toolkits, but also in incorporating information sources that cannot be easily or compactly represented using WFSTs. For example, the decoder can allow us to apply the translation length penalties and phrase penalties to score the partial translation candidates during search. In addition, they can incorporate new parameter values (e.g., language model weight) at runtime without the need for any modification of the input WFSTs.

3.5. Feature Extraction

One significant disadvantage of using WFST based translation method is a lack of simple feature based probabilistic training algorithm for WFSTs. Even though Eisner [8] has proposed a training algorithm for "probabilistic WFST", it is not an easy task to add features for parameterization of WFSTs. For example, if we want to add a feature that defines "if previous word same as the current one" in the WFST we would need to add self looping arcs for all the words in the vocabulary adding significant amount of confusability. It is obvious that any more complicated rule for parameterization would add many arcs to the translation lattice. We propose a simple method to add such features in our WFST framework.

We limit ourselves to nominal features. We extract nominal features for each word and convert them to a binary string. We append the binary string to the end of each noisy word. For example if f_1, f_2 were the binary features mentioned above each of our noisy word n is appended with $f_1 f_2$ producing $n_{f_1 f_2}$. Adding too many features can increase data sparsity problems, so we have to be careful on the number and type of features we use.

4. Evaluation

4.1. Corpus

We tested our method on Penn III Switchboard tree bank corpus. We split our data in a training set of 1221502 words and held out a test set of 199520 words. We performed all of our experiments on manual transcripts. We do not assume that we have sentence boundaries or interruption point information. But we do assume that we have turn boundaries so that the decoder can handle sizable chunks of speech without severe memory problems. Since our model is not a feature-based model, we do not use turn boundary information to extract any feature that may provide unfair advantage to the model.

For training, we use the annotation for repeat, repair and filled pauses that are provided with the corpus. We convert the switchboard annotation to produce a parallel text corpus of clean and noisy speech transcripts where all the disfluent words in a noisy transcript align with disfluent token IDs. These disfluent token IDs signify the type of disfluency for the corresponding word in the noisy transcript as shown in the example in Section 3.1. Let us define C as the annotated clean corpus and N as the corresponding noisy parallel corpus.

4.2. Experiment and Results

We first extracted two lexical features for all the 1.22 million words in corpus C and appended these feature values to each word, thus obtaining a corpus CF. The two features we extracted were "*are part of speech tags of any of the next two words the same as the current one*" and "*are any of the next two words the same as the current word*." In order to obtain the features based on part-of-speech, we use an automatic part-of-speech tagger [18]. The training corpus had 13287 repair phrases with only 48 phrases longer than 5 words. Most of the disfluent phrases are composed of one or two words. 11.8% of all the phrases constitute more than 2 words.

Using the same phrase extraction algorithm in [19] we extracted all the parallel phrases from the corpora N and CF. We set the limit of phrase size as five words. We obtained a phrase dictionary with 2.85 million entries. Each entry consists of a noisy phrase, corresponding clean phrase and a probability of its translation. We computed the translation probability as described in Section 3.3 for each clean and noisy phrase pair. We smoothened the transition probability as described in Section 3.3 with a δ of 0.01.

Type	# of states	# of transitions
Translation	1,635,398	3,426,379
LM	234,118	1,073,678

Table 2: The Size of the Translation Lattice and LM

We built the WFSTs as described 3.3 and composed them to produce a noisy to clean speech translation lattice. We built the language model using the IBM language model toolkit. The size of the final translation lattice is listed in Table 2.

When we tested our model on our held out test set we obtained the results listed in Table 3. We tested our method by using the standard precision, recall and F-measure. The scores were computed at the word level. The train and test data are heavily skewed with very few positive examples of disfluency. In our test set of 199520 words only 6.93% of words were disfluent so F-measure is a reasonable metric for the system evaluation.

	Disfluency	Precision	Recall	F-measure
3*w/o LM	REPEAT	0.695	0.809	0.747
	REPAIR	0.479	0.256	0.334
	FILLED PAUSE	0.953	0.998	0.975
3*with LM	REPEAT	0.743	0.860	0.797
	REPAIR	0.509	0.331	0.401
	FILLED PAUSE	0.955	0.998	0.976

Table 3: Results on Held-out Test Set

We built two different type of translation lattices to examine the effects of language model. When we added the language model (LM) to the translation model that did not have LM information, the F-measure improved for repeats by 4.9%, precision by 4.8% and recall by 5.1%. Similarly for repairs, F-measure improved by 6.7%, precision by 3% and recall by 7.5%. The addition of LM only slightly improved filled pause results signifying that LM is more critical for detecting repeats and repairs than filled pauses. We also note that the improvement in F-measure and recall with the addition of LM for repairs is significantly higher than it is for repeat, possibly showing that taking account of surrounding word context is more important for repair detection. F-measure for repeat detection is 39.6% higher than F-measure for repair detection signifying the difficulty of repair detection. We obtain very high recall of 0.86 and F-measure of 0.797 for repeat detection.

We are able to detect filled pauses with a greater accuracy than repeats and repairs. We obtain an F-measure of 0.976 for filled pause detection. One of the reasons that the same translation model does very well for filled pauses but not for repeat and repairs is because most of the filled pauses are unigrams and a few set of words constitutes most of the filled pauses. The most common filled pause in our test corpus was “uh” constituting 82.7% of all the filled pauses. The least occurring filled pauses were “ah”, “anyway” that occurred only once.

One of the key advantages of using our method for disfluency detection is speed. We are able to detect disfluency in one thousand sentences in less than a second using a memory optimized WFST decoder developed for phrase-based translation [19].

5. Conclusion

We presented a novel approach that is based on a phrase-level translation framework for detecting repeats, repairs and filled pauses. We also proposed methods for simplifying the phrase-level translation technique by retokenization of words in the speech transcript such that we do not require fertility and alignment models. We showed a simple method of incorporating features in a weighted finite state transducer. Our method only requires parallel corpora of noisy and clean speech transcripts reducing the amount of natural language resources needed for disfluency detection. In addition, using the optimized decoder, our proposed method for disfluency detection has key advantages of a fast speed and a small memory footprint, compared to other approaches.

6. References

[1] Brown, P.F., Pietra, S.D., Pietra, V.J.D., Mercer, R.L., The Mathematics of Statistical Machine Translation: Parameter Estimation, Computational Linguistics, Vol.19 No 2

[2] Charniak, E., and Johnson, M., Edit Detection and Parsing for Transcribed Speech, Proc. of NAACL, 118-126.

[3] Heeman, P.A. and Allen, J.F., Combining the Detection and Correction of Speech Repairs, Proc. of ICSLP, 363-365.

[4] Honal, M., and Schultz, T., , Correction of Disfluencies in Spontaneous Speech Using a Noisy Channel Approach, Eurospeech 2003.

[5] Honal, M., and Schultz, T., , Automatic Disfluency Removal on Recognized Spontaneous Speech Rapid Adaptation to Speaker Dependent Disfluencies, ICASSP 2005.

[6] Johnson, M. and Charniak, E.,, A TAG-based Noisy Channel Model of Speech Repairs, Proc. of ACL, 33-39.

[7] Johnson, M., Charniak, E., Lease, M., An Improved Model for Recognizing Disfluencies in Conversational Speech, RT04.

[8] Jason Eisner, Parameter Estimation for Probabilistic Finite-State Transducers, Proc. of the 40th Meeting of the Association for Computational Linguistics.

[9] Kim, J., Schwarm, S.E., Ostendorf, M., Detecting Structural Metadata with Decision Trees and Transformation-Based Learning, Proc. of HLT/NAACL.

[10] Liu Y. Shriberg E. Stolcke A., Automatic Disfluency Identification in Conversational Speech Using Multiple Knowledge Sources, EuroSpeech 2003.

[11] Liu, Y., Shriberg, E., Stolcke, A., Hilliard, D., Ostendorf, M., Peskin, B., Harper, M., , The ICSI-SRI-UW MetaData Extraction System, ICLSP, 2004.

[12] Nakatani, C. and Hirschberg, J., A Corpus Based Study of Repair Cures on Spontaneous Speech , Journal of the Acoustical Society of America, 1603 - 1616.

[13] Spiker, J., Klarner, M., Gorz, G., Processing Self Corrections in a Speech to Speech System, Proc. of 18th Conference in Computational Linguistics.

[14] Snover, M., Dorr, B., Richard, S., A Lexically Driven Algorithm for Disfluency Detection, Short Paper Proc. of HLT-NAACL 2004.

[15] Shriberg, E., Bates, R, Stolcke, A., A Prosody only Decision Tree Model for Disfluency Detection , Proc. of Eurospeech 1997.

[16] Shriberg, E., Preliminaries to a Theory of Speech Disfluencies, PhD Thesis.

[17] Stolcke, A., Shriberg, E., Bates, R., Ostendorf, M., Automatic Detection of Sentence Boundaries and Disfluencies Based on Recognized Words, Proc. of ICSLP 1998.

[18] Tufis, D., and Mason, O., Tagging Romanian Text: A Case Study for QTAG, A Language Independent Probabilistic Tagger, LREC, 1998.

[19] Zhou, B., Chen, S., Gao, Y., A Memory Efficient Phrase-based Machine Translation Using Statistical Integrated Phrase Lattices, To be published.

[20] Zhou, B., Chen, S., Gao, Y., Constrained Phrases-Based Translation using Weighted Finite State Transducer , ICASSP 2005.