Correction of Disfluencies in Spontaneous Speech using a Noisy-Channel Approach

Matthias Honal, Tanja Schultz

Interactive Systems Laboratories
University of Karlsruhe Germany, Carnegie Mellon University USA
honal@ira.uka.de, tanja@cs.cmu.edu

Abstract

In this paper we present a system which automatically corrects disfluencies such as repairs and restarts typically occurring in spontaneously spoken speech. The system is based on a noisy-channel model and its development requires no linguistic knowledge, but only annotated texts. Therefore, it has large potential for rapid deployment and the adaptation to new target languages. The experiments were conducted on spontaneously spoken dialogs from the English VERBMOBIL corpus where a recall of 77.2% and a precision of 90.2% was obtained. To demonstrate the feasibility of rapid adaptation additional experiments on the spontaneous Mandarin Chinese CallHome corpus were performed achieving 49.4% recall and 76.8% precision.

1. Introduction

Spontaneous spoken speech usually contains disfluencies such as filler words, repairs or restarts of whole sentences which do not contribute to the meaning of the spoken utterance. Disfluencies cause sentences to be ill-formed, longer, and thus harder to process for natural language understanding (NLU) components applied for example in summarization systems or speech-to-speech translation engines. The aim of this work was to develop a disfluency correction component that removes disfluencies from the given input text to ease the subsequent NLU task.

1.1. Disfluencies

For the description of the disfluencies structure we follow [4]. As figure 1 shows, a disfluency consists of four essential components, the reparandum, the interruption point (IP), the interregnum, and the repair. The reparandum contains the words which will we edited (i.e. repeated, corrected or abandoned completely). The offset of the reparandum is marked by the IP which is the point, where the speaker interrupts the utterance in order to repair it. The interregnum can consist of a short silence, of an editing term indicating that the reparandum will be edited, or it can even be empty at all. The editing finally takes place in the repair. This pattern can be used to describe both, complex disfluencies such as repairs and restarts of sentences and simpler disfluencies, like filler words. In the latter case reparandum and repair are simply assumed to be empty.

The correction of disfluencies involves the removal of all parts of the disfluency which do not belong to the utterance originally intended by the speaker. Given the structure presented above, it is easy to see that a disfluency can be corrected by simply deleting the reparandum and the interregnum. Hence the corrected version of the sentence in figure 1 is: "We need three tickets to Boston."

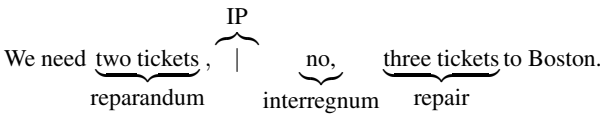

Figure 1: Typical structure of disfluencies

1.2. Related Work

A number of systems for automatic disfluency correction have been presented in the past. Rule based approaches are proposed by Hindle [3] and Bear et al. [1]. In [3] the focus is to apply editing and restart correction rules to disfluencies, which are identified in a text by hand labeled interruption points. In [1] disfluencies are detected using pattern matching and syntactic knowledge. Correction is then done by applying several correction patterns. Using pattern matching a recall of 44% and a precision of 48% on the ATIS corpus is reported. Heeman proposes a system that applies statistical models for disfluency correction [2]. This system uses a Part-of-Speech (POS) based language model which predicts not only words and their corresponding POS-categories, but also marks reparandum onsets, interruption points and filler words using decision trees asking questions about words, POS-tags, and silences. On the corpus "Trains" a recall of 65.9% and a precision of 74.3% is reported for repair correction. Zechner introduces disfluency dependent treatments [7]. Filler words and editing terms are detected by a rule based POS-tagger, the reparanda of repairs are detected by a pattern matching component, and for the detection of the reparanda of false starts he uses shallow syntactic knowledge and decision trees applying questions about words, POS-tags and syntactic entities. On the English CallHome and CallFriend corpora the system achieves a recall of 51.9% and a precision of 59.4%.[1] Spilker et al. [5] propose a different statistical approach for disfluency correction by using acoustic cues to identify potential interruption points. The extend of repair and reparandum is determined by using statistical machine translation methods to translate one into the other. Using this approach they obtain a recall of 64% and a precision of 84% on the German pendant of the VERBMOBIL corpus.

1.3. Approach

The system presented here is based on a noisy-channel approach which is adopted from statistical machine translation (SMT). Following the SMT terminology we define the disfluent text to

[1] These results are not reported in [7], but achieved by evaluating his engine in our lab.

be the source language which has to be translated into the target language, namely the fluent text free of any disfluencies. As shown in section 1.1 disfluencies are corrected by deletions, hence translation here simply means the deletion of disfluent words. Like in SMT we search for the most likely target language sentence given a sentence in a source language. This search takes all possible hypotheses for target sentences into account which can be generated from the source sentence by deletions. In order to assign probabilities to these hypotheses, a number of models for different properties of disfluencies are used as described below.

Our approach has several advantages for the development of the correction system: (1) Portability: no linguistic knowledge is needed (e.g. knowledge about the grammar for parsing or POS-tagging of the language in question), rather a text containing annotated disfluencies is sufficient to train the system. As a result our system can easily be adapted to various languages. (2) Granularity: rather than matching rules, statistical models are used instead to make decisions about deletions which allow for case-to-case decisions depending on a number of features. (3) Flexibility: our approach allows to easily incorporate new models that make use of disfluency properties yet to be investigated.

2. Data

All experiments are conducted on spontaneously spoken dialogs in two languages, American English and Mandarin Chinese. The English data (EVM) consists of the transcriptions of 127 dialogs from the VERBMOBIL corpus which consist of spontaneous spoken clean speech between two partners scheduling appointments and doing travel arrangements in a face-to-face scenario. The Mandarin data (MCC) consists of 100 transcribed fragments of telephone conversations (80 from the training set, 20 from the evaluation test set) between two native speakers of Mandarin Chinese. Since Chinese text is written without spaces the transcriptions were automatically segmented into the most likely sequence of words.

	EVM	MCC
Dialogs	127	100
Sentences	16583	24275
Words	118356	202099
Vocabulary	2290	7503

Table 1: *Statistics from the EVM and the MCC corpus*

As shown in table 1 the vocabulary of the MCC corpus is more than three times larger than the vocabulary of the EVM corpus while the MCC corpus itself is not even two times larger than the EVM corpus. This suggests that disfluency correction for the MCC corpus is more difficult than for the EVM corpus due to the data sparseness.

3. Disfluencies

As already mentioned the complexity of disfluencies ranges from simple types like filler words to very complex types of disfluencies which involve editing or abandoning of whole phrases. In this work we distinguish the following types of disfluencies:

- False Start (FS): A sentence is aborted before completion and a new sentence is started. Example: I have - I am free the twenty third.
- Repetition/Correction (REP): A phrase is repeated twice or corrected by deleting, substituting or inserting words. This disfluency type can be characterized by word correspondences between reparandum and repair. Example: We need two tickets, no, three tickets to Boston.
- Editing Term (ET): Editing terms occur in the interregnum of FSs or REPs and indicate that the reparandum will be edited (or abandoned completely). "No" in the sentence above is a disfluency of that type.
- Filler word (FP): Filler words (sometimes referred to as filled pauses or discourse markers) are those words which have no semantic content but provide some discourse functions like helping the speaker to start or to keep his turn. Example: Alright, well, this is a good idea.
- Interjection (IN): Interjections are defined as non-lexicalized sounds which indicate affirmation or negation like "mhm", "uh-uh" etc. In our data they are basically used as back-channeling, and as a result have been removed.

In case of the EVM corpus we used all disfluency types defined above, however in case of the MCC corpus only disfluencies of the types FS, REP and FP are annotated in the transcriptions. Table 2 shows the frequencies of disfluency types and tokens for both corpora.

	FS	REP	FP	ET	IN	Total
EVM	1209	2303	7024	155	336	11027
MCC	9603	241	11745	-	-	21589

Table 2: *Types of disfluencies in the EVM and the MCC corpus*

In order to detect disfluencies in the text our system uses various disfluency properties. One example is the length of the deletion region of a disfluency, i.e. the number of words which have to be deleted for correction. Table 3 shows the distribution of disfluencies over the length of their deletion region for the EVM corpus, the distribution for the MCC corpus is very similar. As can be seen the one-word deletion regions are clearly dominant. This behavior corresponds with our intuition that filler words mostly consist of one word that is deleted for correction. However, surprisingly it is not only true for filler words but also for repetitions/corrections and false starts.

Furthermore, we investigated the position of disfluencies in a sentence and observed that disfluencies are much more likely to occur at the beginning of a sentence than in the middle. Additionally, word fragments (i.e. words which are aborted prematurely) which occur at the end of the reparandum of a disfluency can be helpful for disfluency detection. In the EVM corpus we found that 753 out of 914 fragments occur at the end of the reparandum of a disfluency. Therefore, we conclude that fragments are helpful for identifying disfluencies but do not account for all of them.

4. Noisy-channel Approach

The noisy-channel approach is a basic concept of SMT [6]. When adapting it to the problem of disfluency correction, the underlying idea is that "clean" i.e. fluent speech gets passed through a noisy channel. The channel adds noise to the clean speech and thus creates "noisy", i.e. disfluent speech as output. Given a "noisy" string N the goal is to recover the "clean"

Length of DR	1	2	3	4	5	7	8-11
Number of DFs	8402	1747	443	193	111	73	58

Table 3: *Disfluencies (DFs) grouped by the length of their deletion regions (DR) for the EVM corpus*

string $\hat{C}$ such that $P(\hat{C}|N)$ becomes maximal. Using Bayes Rule, this problem can be expressed as follows:

$$\hat{C} = \arg\max_{C} P(C|N) = \arg\max_{C} P(N|C) \cdot P(C), \quad (1)$$

where the probability $P(C)$ denotes the language model probability for the fluent string; $P(N|C)$ is the probability that the noisy channel generates N as output given C as input. In terms of SMT the latter probability is referred to as the translation model. In SMT alignments are used to establish correspondences between the positions of the source and the target sentences. In our approach we also make use of alignments but they differ from SMT in the way that disfluency correction only requires deletions of words rather than deletions, insertions and reorderings. Assuming that each target sentence is generated from left to right, the alignment a_j defines whether the word n_j in the source sentence is deleted or appended to the target sentence. Let J be the length and n_j the words of the source sentence N, I the length and c_i the words of the target sentence C and m the number of deletions (of contiguous word sequences) which are made during generation of the target sentence. Then we can introduce an alignment a_j for each word n_j and rewrite $P(N|C)$ as follows:

$$P(N|C) = P(J|I) \cdot P(m|J,C) \cdot \sum_{a_1^J} P(n_1^J, a_1^J | c_1^I, I, J, m) \quad (2)$$

The probability $P(n_1^J, a_1^J | c_1^I, I, J, m)$ can be decomposed into a product of probabilities over all source words n_j. In our system we use five different models which contribute to these probabilities. They are combined by a weighted sum. Each model assigns a translation probability to a word and is named according to the features it uses: (M1) The length of the deletion region of a disfluency, (M2) the position of a disfluency, (M3) the length of the deletion region of a disfluency with a fragment at the end of the reparandum, (M4) the context of a potentially disfluent word, (M5) the information about the deletions of the last two words preceeding a potentially disfluent word. The models (M1), (M2) and (M3) reflect important properties for disfluency identification as outlined in section 3. Models (M4) and (M5) take into account that the local context is often helpful to determine the deletion region of a disfluency.

The probability distributions for the models encoding the features enumerated above are obtained from the training data using relative frequencies. In order to perform an efficient search, we do not store the actual hypotheses, but only sequences of 0 and 1. Each sequence represents uniquely a hypothesis, when we define that the number at position j has to be 0, iff the word n_j is deleted in this hypothesis and 1 otherwise. A straight forward representation of our search space would be a binary search tree as we consider all 2^J possible hypotheses for a source sentence of length J. However, as the number of leafs of this tree grows exponentially with J, we defined merge criteria for 0-1-sequences in order to construct a lattice which can be searched efficiently with a dynamic programming approach. According to these criteria two partial sequences of equal length can be merged, iff the number of performed deletions (i.e. contiguous 0-subsequences) is equal and either the length of the current deletion (i.e. the current 0-subsequence) is equal or there is no deletion at the current position. Each edge of the lattice is assigned a translation probability. The language model probability for a hypothesis is calculated over all those words, which are not deleted in this hypothesis.

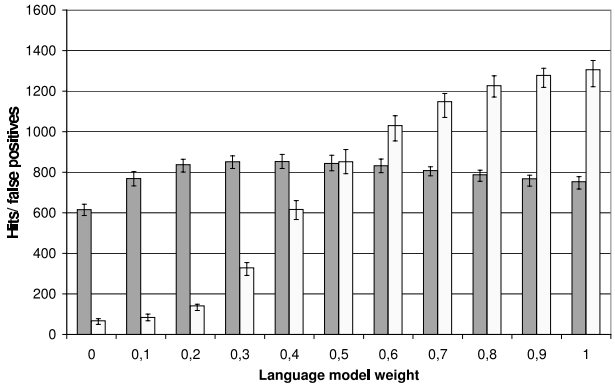

Figure 2: *Hits (grey bars) and false positives (white bars) for different language model weights for the EVM corpus; The error bars indicate the deviation among the test sets*

5. Experimental Results

All experiments are conducted on the EVM and on the MCC corpus. The EVM corpus was split into 10 disjoint test sets of 10% corpus size, the corresponding 90% remainder of the corpus were used for training. The presented results on EVM are averaged over the 10 test sets. For the MCC corpus we used the predefined splitting into a training, development, and evaluation set. The results on the MCC are reported on the evaluation test set.

As described above, different weights can be assigned to the language model probability and the five models composing the translation probability. The major goal of the following experiments was to optimize the weighting parameters and to investigate the impact of these free parameters on the overall system performance. The implementation of an automatic optimization algorithm is left for future work.

5.1. Language Model Weight

Firstly, we investigated the influence of the language model weight parameter. Figure 2 illustrates the changes of hits (complete corrections of disfluencies) and false positives (deletions of only non-disfluent words) for the EVM corpus. While the number of hits is increasing up to a language model weight of 0.4 and then slightly decreasing, the number of false positives is growing rapidly with larger language model weight. This could be a result of the fact that a high language model weight causes the system to delete events which were not seen during training. Good results are obtained for a language model weight of 0.2. Here the number of hits is comparatively high and the number of false positives is still low. For the MCC corpus we observed similar trends. However the best language model weight turned out to be 0.1 in the case of MCC. The lower language model weight might be a consequence of the larger vocabulary and resulting data sparseness with an even larger number of unseen events.

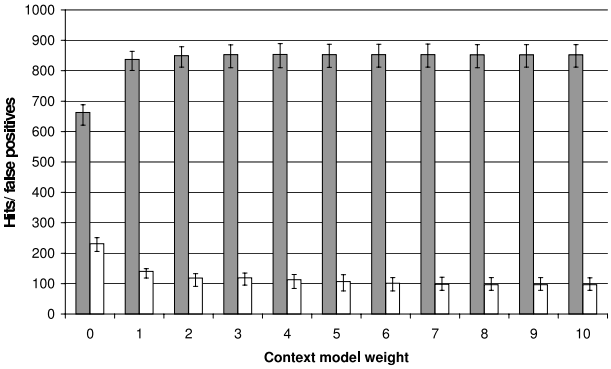

Figure 3: *Hits (grey bars) and false positives (white bars) for different context model weights (language model weight set to 0.2) for the EVM corpus; The error bars indicate the deviation among the test sets*

5.2. Context Model Weight

The most remarkable effect on the overall performance gain results from the context model (M4). This model considers the context of a potentially disfluent word. The positive effect of the context model can be easily explained for filler words, since it allows to discriminate between the deletion of the word "well" in the context "Well done!" and "Alright, well, this is a good idea.". We also found it to be helpful for short repetitions/corrections, such as "the the" or "he she". Figure 3 illustrates the impact of the context model weight. In these experiments we set the language model weight to 0.2. With increasing context model weight the number of hits improves only slightly, but the number of false positives decreases significantly. For the EVM corpus a weight of 10 gave the best results, for the MCC corpus the best weight turned out to be 5.

Model	Hits	False Positives
(M1)	-7.9	-5.9
(M2)	+10.1	+21.2
(M3)	+2.6	+8.8
(M4)	+174.1	-90.3
(M5)	+15.9	+264.6

Table 4: *Contribution of all the five models to the baseline system (for the EVM corpus with language model weight set to 0.2); The model numbers correspond to the enumeration in section 4*

The effect of all the five models is summarized in table 4. It shows that the impact is a slight increase of the number of hits at the cost of a slight increase of the number of false positives or a slight decrease of both figures for the models (M1), (M2), and (M3). The numbers for (M4) refer to the comparison of weight 0 to 1. Model (M5) causes a huge number of false positives and is therefore disregarded for the best system.

5.3. Best System

Table 5 shows the final results for the best parameter combinations for the EVM and the MCC corpus. The total number of disfluencies in the test sets are given in parentheses. The results confirm our assumption that disfluency correction on the MCC corpus is much more difficult than on the EVM corpus.

Our system was first developed for English using the EVM corpus and then ported for the MCC corpus. Almost no effort is needed for the adaptation to Mandarin Chinese. The same algorithms and the same statistical models can be used. Only the weighting parameters for the models are different for the MCC corpus. We adjusted them empirically using the experience from our work with the EVM corpus.

	Hits	False Positives	Recall	Precision
EVM	853.3 (1102.7)	92.4	77.2%	90.2%
MCC	1486 (3008)	448	49.4%	78.8%

Table 5: *Hits, false positives, recall and precision for the best parameter combinations for the EVM and the MCC corpus (total number of disfluencies in the test sets in parentheses)*

6. Conclusions

In this paper we proposed a statistical approach for disfluency correction using a noisy-channel approach. The resulting system performs well for disfluency corrections achieving up to 77.2% recall and 90.2% precision. Our approach allows for rapid adaptation to new languages which has been demonstrated on the portation from English to Mandarin Chinese. The noisy-channel approach does not require any linguistic knowledge, is easy to train and very flexible. It allows to incorporate additional models, and supports a high granularity since it allows for case-to-case decisions. In the next steps we will investigate the impact of using automatically transcribed input data rather than manual transcriptions. Furthermore we will study the effect of acoustic features such as the duration of pauses and words or intonational boundaries. Finally, an algorithm will be implemented that optimizes the model weight parameters.

7. References

[1] Bear, J., Dowding, J. and Shriberg, E., "Integrating Multiple Knowledge Sources for Detection and Correction of Repairs in Human-Computer Dialog", Proc. of the 30th Annual Meeting of the ACL, 1992

[2] Heeman, P. A., "Speech Repairs, Intonational Boundaries and Discourse Markers: Modeling Speakers' Utterances in Spoken Dialog", PhD-Thesis, University of Rochester, 1997

[3] Hindle, D., "Deterministic Parsing of Syntactic Nonfluencies", Proc. of the 21th Annual ACL Meeting, 1983

[4] Shriberg, E., "Preliminaries to a Theory of Speech Disfluencies", PhD-Thesis, University of California at Berkeley, 1994

[5] Spilker, J., Klarner, M., Görz, G., "Processing Self-Corrections in a Speech-to-Speech System", Verbmobil: Foundations of Speech-to-Speech Translation, Springer Verlag Berlin, 2000

[6] Wang, Y., Waibel, A., "Decoding Algorithm in Statistical Machine Translation", Proc. of the 35th Annual Meeting of the ACL, 1997

[7] Zechner, K., "Automatic Summarization of Spoken Dialogues in Unrestricted Domains", PhD-Thesis, Carnegie Mellon University, 2001