AUTOMATIC DISFLUENCY REMOVAL ON RECOGNIZED SPONTANEOUS SPEECH - RAPID ADAPTATION TO SPEAKER-DEPENDENT DISFLUENCIES

Matthias Honal, Tanja Schultz

Interactive Systems Laboratories
University of Karlsruhe Germany, Carnegie Mellon University USA
76128 Karlsruhe, 15212 Pittsburgh

ABSTRACT

In this paper we investigate methods to adapt a system for disfluency removal to different data properties. A gradient descent algorithm for parameter optimization is presented which achieves 85.1% recall and 93.1% precision on the English Verbmobil corpus and 53.0% recall and 79.0% precision on the Mandarin Chinese CallHome corpus. This compares to the results produced with hand-optimization on the test set. Furthermore we investigated the impact of cross-validation and training set selection on recognizer output. Finally we examined speaker dependent disfluency production behavior and clustered training data accordingly in order to improve the overall system.

1. INTRODUCTION

Spontaneous speech is disfluent. Disfluencies are those parts of spontaneous speech, which can be removed in order to retrieve the originally intended fluent utterance. Disfluency removal makes sentences shorter, less ill-formed and thus facilitates the downstream processing by natural language understanding components such as machine translation or summarization. Initial results on integrating our disfluency remover into a speech-to-speech translation system show very promising results.

1.1. Disfluencies

We distinguish three disfluency categories:

- Repetitions/Corrections: A word sequence is repeated or modified without changing the original train of thought.
- False starts: An utterance is aborted, then restarted with a new idea or train of thought.
- Discourse markers: Filler words which do not contribute to the semantic content of the discourse but indicate the speakers intend to keep or take a turn. Furthermore, words are considered as discourse markers which occur within repetitions or false starts, indicating that previously uttered words will be corrected. Words of the latter category are sometimes called editing terms.

The goal of disfluency removal is to delete disfluent words. In the case of discourse markers this results simply in deleting the above defined filler words and editing terms. More complex disfluencies (repetitions and false starts) can be divided in a reparandum (words that will be abandoned, repeated or corrected), followed by an optional interregnum (an editing term, as described above) and a repair, where the actual repetition, correction, or restart takes place. Disfluency removal for these complex disfluencies is thus the removal of the reparandum. Table 1 shows examples for disfluent sentences before and after disfluency removal.

Repetition	"I think **you should**, no, we should leave early in the morning" becomes "I think, we should leave early in the morning."
False Start	"**We'll never find a day**, what about next month?" becomes "What about next month?"
Discourse marker	"**Well**, **let's see**, we can leave Thursday, **no**, Friday." becomes "We can leave Friday."

Table 1. *Examples of disfluency types and their removal; disfluent words of the different types appear in bold face*

1.2. Related work

Several systems for disfluency correction or detection have been proposed in the past. In [1] Stolcke et al. propose a system for sentence boundary and disfluency detection in speech recognizer output. A prosodic model and a language model are combined to identify the interruption point of disfluencies (it marks the offset of the reparandum). On the Switchboard corpus the system identifies 93.0% of the considered events correctly; on recognizer output the performance achieves 76.6%.

The system presented by Liu, Stolcke and Shriberg in [2] combines a word-based and a POS-based language model with a language model accounting for repetition patterns and with acoustic and prosodic cues in order to identify interruption points of disfluencies. Rule based knowledge is then applied to find the reparandum onset in order to remove the disfluencies. On the Switchboard corpus a recall of 61.45% and a precision of 68.64% is reported.

Spilker et al. developed a system [3] where interruption points are identified using acoustic cues. The extent of repair and reparandum is determined using methods of statistical machine translation to translate one into the other. On the German Verbmobil corpus they achieve a recall of 64% and a precision of 84% on manually transcribed speech. These results can be compared to the results of our previous system discussed in [4]: Applying statistical machine translation methods to identify and remove disfluencies we could achieve a recall of 77.2% and a precision of 90.2% on manually transcribed data of the English Verbmobil corpus.

Based on our system as introduced in [4] we investigate in this paper how system parameters and the data for training and cross-validation can be selected in order to optimize the overall system performance. The paper is organized as follows: After the

description of the data set and a short introduction into the basic concepts of our system, we present a gradient descent method for automatic parameter optimization using a cross-validation set. Next we compare the impact of manually transcribed text versus recognized speech used as training and cross-validation material for disfluency removal in recognizer output. Finally we present our approach to adapt the disfluency removal system to speaker clusters differing in the amount of produced disfluencies.

2. DATA

Experiments were conducted based on two different corpora: The English Verbmobil Corpus (EVM) for American English and the Mandarin Chinese CallHome corpus (MCC) for Mandarin. Both corpora consist of spontaneously spoken dialogs spoken by native speakers. Table 2 shows important corpus characteristics.

	EVM	MCC
Dialogs	127	100
Speakers	60	200
Sentences	16583	24275
Words	118356	202099
Vocabulary	2290	7503

Table 2. *Statistics from the EVM and the MCC corpus*

Both corpora are divided in a training set, a test set, and a development test set. The latter is used for cross-validation. Table 3 shows the size of the different data sets. For speech of the EVM corpus we conducted experiments on both, the manual transcriptions and the output of the recognizer. Since only little data is available, the entire test set of the recognizer was used for testing and tuning the disfluency removal system. The recognizer achieves 76.1% word accuracy on this test set. For training the disfluency removal system we used the recognizer output of the recognizer's training set. The recognizer performance on this training set is 88.1% word accuracy.

Disfluencies were annotated manually for the MCC corpus. For the EVM corpus, repetitions and false starts were annotated manually, while discourse markers were annotated automatically using Zechner's system as described in [5]. For more details on disfluencies in both corpora please refer to [4].

3. APPROACH

For the removal of disfluencies we use the noisy-channel approach, a concept which is borrowed from statistical machine translation [6]. The basic idea is that a fluent string C is passed through a channel that adds noise (in form of disfluencies) to this string. We can only observe the noisy, i.e. the disfluent string N. The goal of disfluency removal is to recover the string $\hat{C}$ that is most likely to be the fluent input string given the noisy output string. In statistical machine translation the fluent string is associated with the target language, the disfluent string with the source language. Thus, our problem can be reformulated as the translation of the disfluent string into a fluent one. In equation 1 the problem is expressed in mathematical terms and reformulated using Bayes rule. C denotes the fluent string, N the disfluent string.

	Training	Test	Devtest
EVM	91.1K	12.4K	9.1K
MCC	156.6K	21.1K	41.3K

Table 3. *Data size in terms of words (EVM and MCC corpus)*

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

We model the probability $P(C)$ with an n-gram language model trained on fluent speech. The probability $P(N|C)$ (the "translation model") can be decomposed as follows[1]:

$$P(N|C) = P_{I,J}(m) \cdot \prod_{j=1}^{J} P_w(n_j) \quad (2)$$

The probability $P_{I,J}(m)$ models the number m of contiguous word sequences which can be deleted in N to obtain C. $P_w(n_j)$ is the probability that word n_j of the string N is disfluent. I denotes the length of the fluent sentence C and J the length of the disfluent sentence N.

Each of the probabilities $P_w(n_j)$ is finally composed of a weighted sum over the following six models: (M1) models 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 word fragment at the end of the reparandum, (M4) the context of a potentially disfluent word, (M5) uses information about the deletions of the last two words preceding a potentially disfluent word and (M6) takes into account whether a potentially disfluent word is part of a repeated word sequence. Thus we have

$$P_w(n_j) = \frac{\sum_{k=1}^{6} \lambda_k P_{Mk}(n_j)}{\sum_{k=1}^{6} \lambda_k} \quad (3)$$

where $P_{Mk}(n_j)$ is the contribution of model Mk and λ_k is the weighting factor for model Mk. The probability $P_{M_k}(\cdot)$ is derived from the occurrences of disfluency characteristic k in the training data. For a more detailed description of the models and their impact on the system's performance, please refer to [4].

4. GRADIENT DESCENT

Using equations (1), (2) and (3) and transforming the result into negative log space, our search criterion becomes (we write $S(x)$ for $-\log P(x)$):

$$\hat{C} = \arg\min_C S(C|N) = \arg\min_C \gamma S(C) + (1-\gamma) S(N|C) \quad (4)$$

with

$$S(N|C) = S_{J,I}(m) - \sum_{j=1}^{J} \log \left(\sum_{k=1}^{6} \frac{\lambda_k P_{Mk}(n_j)}{\sum_{k=1}^{6} \lambda_k} \right) \quad (5)$$

The factor γ represents a weighting factor that controls the influence of the language model over the translation model. While all probabilities can be learned from the training data, the weighting factors γ and λ_k have to be determined separately. In our work we use an iterative gradient descent procedure which maximizes the average probability $P(C|N)$ for given pairs (C, N) of

[1] This is a slightly simplified presentation that omits detailed information of the probabilities. For more details please refer to [4].

the training data. (Maximization of $P(C|N)$ corresponds to minimization of $S(C|N)$ in negative log space.)

Starting with a set of initial parameters $(\gamma^{(0)}, \lambda_k^{(0)})$, for each pair (C, N) new parameter values are calculated using the following update rules:

$$\gamma^{(l+1)} = \gamma^{(l)} + \Delta\gamma^{(l)} \quad \lambda_k^{(l+1)} = \lambda_k^{(l)} + \Delta\lambda_k^{(l)} \tag{6}$$

The update quantities $\Delta\gamma^{(l)}$ and $\Delta\lambda_k^{(l)}$ are calculated as follows:

$$\Delta\gamma^{(l)} = -\eta\frac{\partial S^{(l)}(C|N)}{\partial\gamma} + \alpha\Delta\gamma^{(l-1)} \tag{7}$$

$$\Delta\lambda_k^{(l)} = -\eta\frac{\partial S^{(l)}(C|N)}{\partial\lambda_k} + \alpha\Delta\lambda_k^{(l-1)} \tag{8}$$

The factor η is the learning rate, α is a momentum term, which includes previous update directions to the current update direction and thus influences the effective learning rate. This procedure is continued until the difference between the average value of $S(C|N)$ of the previous and the current epoch (one epoch is one complete iteration through the training set) falls below a given threshold. The final parameter set is taken from the epoch, in which the average value of $S(C|N)$ over all the pairs (N_{xv}, C_{xv}) of development test set is minimal.

Table 4 compares the results obtained from using the gradient descent procedure to results from hand-optimizing the parameters on the test data. As can be seen, the parameter set resulting from the gradient descent procedure achieves the same F_1 as the hand tuned parameter set. These results are very encouraging since the hand optimization was performed on the test data and therefore defines a kind of golden standard. These results hold for both languages, and indicate that the gradient descent procedure generalizes well, even across languages. The performance difference between EVM and MCC may result from the much larger vocabulary size of the MCC corpus which can not be compensated by the larger corpus size. We conclude that the gradient descent procedure is an appropriate method for rapid system development that makes hand tuning obsolete. According to our experience, hand tuning requires tremendous manual and computational effort and expertise, since the whole system has to be run for several parameter combinations and the results have to be evaluated carefully, in order to find a good parameter set.

Setup	Recall	Precision	F_1
Hand optimized (EVM)	86.2%	91.5%	0.444
Gradient descent (EVM)	85.1%	93.1%	0.445
Hand optimized (MCC)	53.4%	77.8%	0.317
Gradient descent (MCC)	53.0%	79.0%	0.317

Table 4. *Results for gradient descent compared to hand tuning on the EVM and the MCC corpus*

5. DISFLUENCY REMOVAL ON RECOGNIZED SPEECH

Results of the previous sections were all performed on manual transcriptions. In this section we describe our experiments on applying the disfluency removal system to recognized speech. All experiments were conducted on the EVM corpus. We investigated the following training and cross-validation setups: (S1) Training on manual transcriptions, cross-validation on recognizer output, (S2) training and cross-validation on recognizer output, (S3) training and cross-validation on manual transcriptions, and (S4) training and cross-validation on a combination of manual transcriptions and recognizer output. Parameters for each setup were optimized separately using the gradient descent procedure. The disfluencies in recognized speech were annotated by aligning the recognizer output with the annotated manual transcriptions using the minimal editing distance. Results of the experiments for the different setups are given in table 5.

Setup	Recall (in %)	Precision (in %)	F_1
(S1)	70.8	80.2	0.376
(S2)	68.2	81.5	0.371
(S3)	70.6	80.1	0.375
(S4)	69.3	80.2	0.372

Table 5. *Results on recognizer output (EVM) for different setups*

As expected, the disfluency removal system suffers a significant degradation when tested on recognition output rather than manual transcriptions. The lower recall may result from the fact that words tagged as disfluent are not deleted by the system, since they are falsely recognized and thus perceived as fluent by the system in the given context. Furthermore, due to recognition errors, sequences which are tagged as repetitions no longer exist as sequences of repeated words. The lower precision can be explained by the fact that wrongly recognized words appear to be disfluent in their context although the original word is fluent.

The setup (S1) achieves the best results with respect to the F_1 score. This indicates that introducing noise in form of recognition errors during training does not help to improve performance on noisy test data. The decrease of recall in the other setups is mostly due to the performance in deletions of short discourse markers (such as "well", "you know"). Using (S2) for training, due to recognition errors these discourse markers occur less frequently in contexts in which they are deleted. Precision increases for experiments with setup (S2) because a system trained on recognized speech can cope better with the problem of ill-formed and ungrammatical sentences in a test set that is based on recognized speech as well. Therefore, a smaller number of false positives are produced, since some ill-formed constructions are tolerated. The results produced with setup (S3) are almost as good as the results produced with setup (S1). This indicates that training on manually transcribed speech produces better results overall, however for tuning the model weights a cross-validation set based on recognized speech seems more appropriate. The combination of manually transcribed and recognized speech in (S4) does not improve the results. This means that a simple combination does not profit from the gains seen in (S2) and (S3).

6. SPEAKER DEPENDENT DISFLUENCY REMOVAL

The production of disfluencies differs across speakers. Some speakers tend to speak more disfluent than others. Figure 1 shows the number of utterances and disfluencies over all test speakers. It can be observed that about 50% of the test speakers produce more than one disfluency per utterance, and the other 50% of speakers produce less than one disfluency per utterance.

When we calculated the performance of our system per speaker we found that both recall and precision vary significantly. The re-

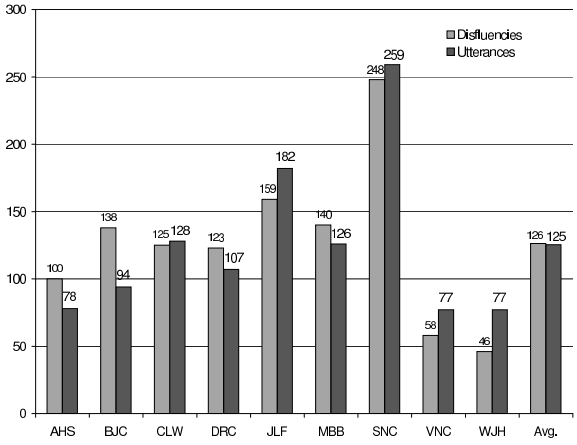

Fig. 1. *Number of disfluencies and utterances over test speakers*

call ranges from 75.2% to 94.0%, the precision ranges from 82.4% to 99.1% across speakers. In order to compensate for this effect we divided the training and the cross-validation set into speaker clusters in the hope that this gives a better model of the speaker dependent disfluency production.

The analysis in figure 1 indicates that the number of disfluencies per utterance could serve as a good criterion to group the speakers into two clusters. We divided the training set, the cross-validation set, and the test set into two clusters M and L: Cluster M contains the speakers producing *more* than one disfluency per utterance, cluster L contains the speakers producing *less* than one disfluency per utterance. In the following we trained the system for both clusters separately, and tested it on the test sets of both clusters. Experiments were conducted on both, manual transcriptions and recognized speech. For experiments on recognized speech the setup (S1) as described above was used, since it achieved the best results on recognized speech. We trimed the training material to assure an equal size of the training sets. This results in 3414 utterances for cluster M (TrM34) and cluster L (TrL34). Furthermore we randomly selected 3414 utterances from the original training set (Tr34). To investigate the impact of this data reduction we compared the results to the system trained on the complete training set (Tr85) consisting of 8573 utterances. Table 6 shows the performance of the resulting systems on the test sets for cluster M (TeM) and cluster L (TeL).

	TRL		ASR	
	TeM	TeL	TeM	TeL
TrM34	79.2/88.9	90.0/89.3	64.6/76.4	78.4/79.8
TrL34	74.0/94.7	87.3/94.3	60.6/82.5	77.1/81.7
Tr34	75.7/92.7	87.9/92.2	62.2/79.3	76.5/80.8
Tr85	79.9/94.0	90.2/91.7	65.5/80.7	77.5/79.8

Table 6. *Recall/precision (in%) for speaker clusters on manual transcriptions (TRL) and recognized speech (ASR)*

Training on the larger training set Tr85 produces in almost all cases better results than training on the smaller training sets. When comparing the results of the training sets built on the two speaker clusters to the reduced training set (Tr34), we observed an improvement of recall at the costs of precision when using TrM34, and vice versa when using TrL34. One reason might be that the system tends to delete more disfluencies when the training data contains many disfluencies. In case the training data contains less disfluencies, the system tends to delete less disfluencies but also produces less false positives.

Table 6 shows that we could achieve improvements on either recall or precision by training on the matching training set (TeM on TrM34; TeL on TrL34) but not managed to improve both at the same time. In terms of F_1 scores, we got small, but significant gains compared to Tr34 when the matching training set is used. This, however, requires that the correct speaker cluster is known or can be derived accurately. Furthermore, we conclude from the comparison between the reduced and the full training set that currently the effectiveness of speaker dependent modeling is hampered by the resulting reduction of the training data.

7. CONCLUSIONS

In this paper we presented three approaches to improve our disfluency removal system. First, we implemented a gradient descent method to automatically optimizing the parameter weights. The resulting system is as good as the golden standard which was set by hand optimizing the parameters on the test data. These results are very encouraging since they allow for a rapid deployment of the disfluency removal system in new domains or languages. Second, we extended our experiments to recognizer output. We achieved best results when we trained the models on manually transcribed data and optimized the model weights on recognizer output data. Finally, we found that the amount of produced disfluencies varies across speakers and investigated a very simple straight-forward criterion to cluster the training data in order to adapt our system to this speaker dependent disfluency production behavior.

8. REFERENCES

[1] A. Stolcke, E. Shriberg, R. Bates, M. Ostendorf, D. Hakkani, M. Plauche, G. Tür, and Y. Lu, "Automatic Detection of Sentence Boundaries and Disfluencies Based on Recognized Words," in *Proceedings of the ICSLP*, 1998, vol. 5, pp. 2247–2250.

[2] Y. Liu, E. Shriberg, and A. Stolcke, "Automatic Disfluency Identification in Conversational Speech Using Multiple Knowlege Sources," in *Proceedings of the 8th Eurospeech Conference*, Geneva, 2003.

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

[4] M. Honal and T. Schultz, "Correction of Disfluencies in Spontaneous Speech using a Noisy-Channel Approach," in *Proceedings of the 8th Eurospeech Conference*, Geneva, 2003.

[5] K. Zechner, *Automatic Summarization of Spoken Dialogues in Unrestricted Domains*, Ph.D. thesis, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, 2001.

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