Hybrid Multi-Step Disfluency Detection

Sebastian Germesin, Tilman Becker, and Peter Poller

Deutsches **F**orschungszentrum für **K**ünstliche **I**ntelligenz GmbH
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany

sebastian.germesin@dfki.de
tilman.becker@dfki.de
peter.poller@dfki.de

Abstract. Previous research has shown that speech disfluencies - speech errors that occur in spoken language - affect NLP systems and hence need to be repaired or at least marked. This study presents a hybrid approach that uses different detection techniques for this task where each of these techniques is specialized within its own disfluency domain. A thorough investigation of the used disfluency scheme, which was developed by [1], led us to a detection design where basic rule-matching techniques are combined with machine learning approaches. The aim was both to reduce computational overhead and processing time and also to increase the detection performance. In fact, our system works with an accuracy of 92.9% and an F-Score of 90.6% while working faster than real-time.

1 Introduction

One major problem in natural language processing (NLP) systems is that they get confused when processing spoken language. This is because of **speech disfluencies** - speech phenomena that are based on the incrementality of human speech production [2]. In fact, 5% - 15% of our speech is disfluent in the form of corrections (1), filled pauses (2), disruptions (3) or even uncorrected sentences (4).

(1) I want to go **to Alex, no,** to Joe.
(2) **Uh,** I want to go to Joe.
(3) **I want to.**
(4) I want to **gone** to Joe.

These disfluencies decrease the performance of ASP systems. [3] quantified the influence of disfluencies on data-driven parsing of spoken language and his experiment showed that "the parsing performance is increased when disfluencies are removed prior to parsing". [4] observed the effect of uncleaned speech disfluencies on N-gram models and described an increased perplexity of the N-gram models which were built on the uncleaned speech material. For cost and time reasons, disfluency detection and correction could be done via an automatic system that is placed right behind a speech-to-text (STT) system (see figure 1).

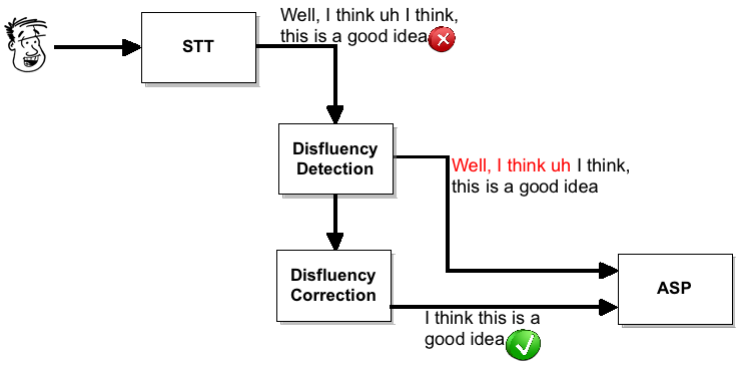

Fig. 1. Speech to ASP with Disfluency Detection

The scheme of the disfluency types this study is based on was developed by [1] as part of the AMI project and is explained in detail in section 2. AMI stands for *Augmented Multi-party Interaction* and is a multi-disciplinary research project to "develop technology to support human interaction in meetings and to provide better structure in the way meetings are run and documented" [5]. The meeting scenario is focused on business meetings with four participants whose task is to design a remote television control. All meetings are held in English and the uncontrolled and natural-like environment produce very good reflections of what happens in real meetings (including speech disfluencies). More details of the corpus possessed by the project and the available disfluency annotations is given in section 3.

1.1 Related Work

A number of researchers published different techniques to detect disfluencies. [6] developed a TAG-based approach (TAG - Tree Adjoining Grammar) combined with a noisy channel model and yielded results of 79.7% F-Score on the Penn 3 disfluency-tagged Switchboard corpus. Later on, [7] extended this approach with a maximum-entropy reranker and manually written deterministic rules and outperformed all state-of-the-art systems in the RT-04F evaluation task. The idea of writing lexical rules for the detection of disfluencies was also followed by [8] who gained competitive results. Additionally, many studies trained machine learning algorithms to recognize disfluencies on lexical [9] as well as on prosodic features [10] and gained equally good results. [10] claimed that combining lexical and prosodic features would result in a system that would outperform both.

1.2 Hybrid Approach

The present work continues these studies in the way that our invented system copes with a broader set of disfluency types. The observed heterogeneity of the

disfluencies led us to the assumption that such a system should be designed in a hybrid way which means that each disfluency should be detected by a special detection technique that was fine-tuned for this disfluency domain. These individual techniques should be combined, resulting in a system that is able to cope with more disfluency types than the individual systems. Additionally, this design leads to a reduced computational overhead and an improved detection performance. Both implementations of the individual techniques as well as of the hybrid system are presented in section 4. The results of the detection systems are contrasted and discussed in the sections 5 and 6.

2 Disfluency Scheme

Disfluencies are "syntactical and grammatical [speech] errors" [1] that occur in spoken language and are present in nearly every conversation. This is based on the incremental speech building process in human articulation [11] and the impossibility to take back already uttered words [12]. [10] found out that between 5% - 10% of our speech is disfluent and, in fact, our corpus contains about 15% erroneous speech material which can be justified by our broader annotation scheme, invented by [1].

The common structure of a disfluency consists of three regions: The **Reparandum** which contains the erroneous speech material, an optional medial region which is called **Interregnum** and the repairing part called the **Reparans**. [1] states that not all disfluencies fit into that scheme and hence splits up her classification scheme into what she calls **simple** and **complex** disfluencies. Simple disfluencies consist only of erroneous speech material while complex disfluencies fit into the common structure. Furthermore, she considers types of disfluencies where the annotator (or the system) has to insert new speech material to gain the speakers intended utterance. She calls them *Uncorrected* disfluencies as they are grammatical errors which were left uncorrected by the speaker.

Finally, she created a finely granulated classification scheme including 15 different classes which are listed in table 1. It shows the abbreviations of these classes and examples that help in the understanding of the particular meaning of the disfluency types. The disfluencies whose name is written in italics are of the simple structure and the rest are complex disfluencies. The disfluency types *Mistake*, *Order* and *Omission* are in fact the previously mentioned *Uncorrected* disfluencies.

3 AMI Corpus

The aims of the AMI (Augmented Multi-party Interaction) project are to develop technology to support human interaction in meetings and to provide better structure in the way meetings are run and documented [5]. To fulfill these, the project possesses a speech corpus with more than 100 hours of four person project meetings. These meetings are all held in English and the task of the

class	abbrev.	example
Hesitation	hesit	This **uh** is an example.
Stuttering	stutter	This is an **exa** example.
Disruption	disrupt	This is an example **and I**
Slip Of the Tongue	sot	This is an **y** example.
Discourse Marker	dm	**Well,** this is an example.
Explicit Editing Term	eet	This is **uh** this is an example.
Deletion	delete	**This really is** this is an example.
Insertion	insert	**This an** this is an example.
Repetition	repeat	**This is** this is an example.
Replacement	replace	**This was** this is an example.
Restart	restart	**We should,** this is an example.
Mistake	mistake	This **be** an example.
Order	order	This **an is** example.
Omission	omiss	This is [ ] example.
Other	other	

Table 1. Overview of all Disfluencies used in this study

particular participants is to design a television remote control. Next to the transcribed speech of the participants, the corpus offers different *annotation layers* that contain a variety of information (e.g., dialog acts, extractive summaries, ASR output, topics, ...).

28 out of the 135 meetings are annotated with our disfluency scheme. The detailed distribution of all classes and the separation in training set and evaluation set are shown in table 2. It is noticeable that the top five disfluencies cover 75% of all appearing disfluencies, which makes them important in the detection task. We split the disfluency corpus into an 80% training set and 20% evaluation set which corresponds to an amount of 10.19 and 2.79 hours meeting time. Despite this, our investigation of the disfluency annotated corpus showed that nearly 15% of all words are disfluent and 40.5% of all dialog acts contain at least one disfluency. Taking into account that these disfluencies would confuse an NLP system, this is quite a huge amount and we will see that our system is able to decrease this. The structure of the disfluencies allow the embedding of other disfluencies but we found out that most of them have either no parent disfluency or just one. The deepest layered disfluency has five parents. Furthermore, we analyzed the length of the disfluencies and about 95% of all simple disfluencies consist of one or two words and the most complex disfluencies have an average length of two to ten words. The longest disfluency - a *Disruption* - contains 24 words.

Additionally, some features for the machine learning approach need N-grams which had to be build on fluent speech material. Therefore, we calculated these N-grams out of the disfluency annotated - and cleaned - training part of the corpus which contains 3760 unique words. As this is a relatively small corpus for the estimation of statistical word probabilities, we were not able to gain the best

	TRAIN	EVAL	SUM	%	kumul. %
hesit	3472	1038	4510	28.67	28.67
repeat	2038	360	2398	15.24	43.92
dm	1981	256	2237	14.22	58.14
disrupt	1389	203	1601	10.18	68.32
sot	928	214	1142	7.26	75.58
omiss	871	82	953	6.06	81.63
mistake	703	61	764	4.86	86.49
stutter	537	123	660	4.20	90.69
restart	502	97	599	3.81	94.49
replace	319	50	369	2.35	96.84
eet	141	19	160	1.02	97.86
insert	117	25	142	0.90	98.76
other	107	6	113	0.72	99.48
order	67	7	74	0.47	99.95
delete	6	2	8	0.05	100.0

Table 2. Distribution of disfluency classes in the corpus.

out-of-vocabulary and perplexity results (see table 3). Therefore, a corpus with more material is definitely preferable and would lead to better performances.

N	OOV	PP
1	3.47%	1181.85
2	27.13%	2674.26
3	80.17%	33310.79
4	95.35%	86872.62

Table 3. N-gram Corpus Statistics

4 Hybrid Detection System

The disfluency detection system is composed of different individual classification methods. Each method is responsible for a subset of disfluency classes and is fine-tuned based on this. In fact, we have two different subsets and hence two different detection approaches.

4.1 Rule-matching Approach

The rule-matching based approach detects disfluencies of the types *Hesitation, Stuttering* and *Repetitions* and uses regular expressions as well as simple word

matching techniques. We will see that these techniques are very strong and lead to no performance loss while transferring them from the development environment into the hybrid system.

4.2 Machine Learning Approach

The machine learning approach is implemented with the help of the freely available WEKA toolkit [13] which contains many state-of-the-art machine learning algorithms and a variety of evaluation metrics. Furthermore, it allows for the adaption of other algorithms due to its simple interface. Disfluencies of the classes *Discourse Marker, Slip of the Tongue, Explicit Editing Term, Restart, Replacement, Insertion, Deletion* and *Other* are detected by a machine learning approach that performs a word-by-word detection. In general, we have four different types of features: **lexical, prosodic, speaker-related** and **dynamic**.

Lexical features are estimated on the word-layer and consider also the POS tags of the particular words. Next to the absolute words, we use some relative lexical features that describe the lexical parallelism between the current word and its neighbors. As [10] describes, prosodic features are well suited for the disfluency detection task and hence we use them as well. The term **prosodic** means, in this context, features that describe the *duration*, *energy*, *pitch pauses* and *velocity* of the words. The *duration* and *pauses* features are calculated directly from the word-boundaries as annotated in the corpus. The *velocity* of a single word is defined by its duration divided by the number of syllables. For the *energy*- and *pitch*-based features, we have 10 ms frames available for each channel of a meeting. As these raw values cannot be used directly, we first had to normalize them globally, based on each channel to eliminate the influence of the microphones. After that, we computed the features on a word and a subword level. The **speaker-related** features describe the speakers *role, gender, age* and *native-language* as we found a correlation between these characteristics and the rate of disfluent words. The last type of features are the **dynamic** features that are generated during the process of the classification and describe the relationship between the disfluency type of the ongoing word to its neighbors.

4.3 Hybrid Design

Figure 2 shows the schematic drawing of the architecture that has been developed. There we can see that both the rule matching subsystem and the machine learning approach work on the data for itself instead of a combined solution where both approaches process the data in parallel. In the first step, the rule-matching system processes the speech material until no more disfluencies can be found. After that, the system's state advances to the machine learning approach where the remaining types of the disfluencies are marked. If this subsystem found any disfluencies, the speech material gets directly re-inserted into the rule-matching system. If not, the labeled stream or the cleaned speech material is made available for a possible subsequent NLP system.

When developing the hybrid system, the first step after the implementation of the particular subsystems was to decide how to arrange both approaches. The presented architecture emerged from a set of different design ideas that were all evaluated on the evaluation part of the corpus. The particular ideas differed in the way both subsystems were placed and in the way the speech was carried through them. In all design steps, we focussed our attention on keeping the precision as high as possible, because wrongly disfluent marked words have more of a bad influence on the meaning of the sentence than wrongly fluent marked ones.

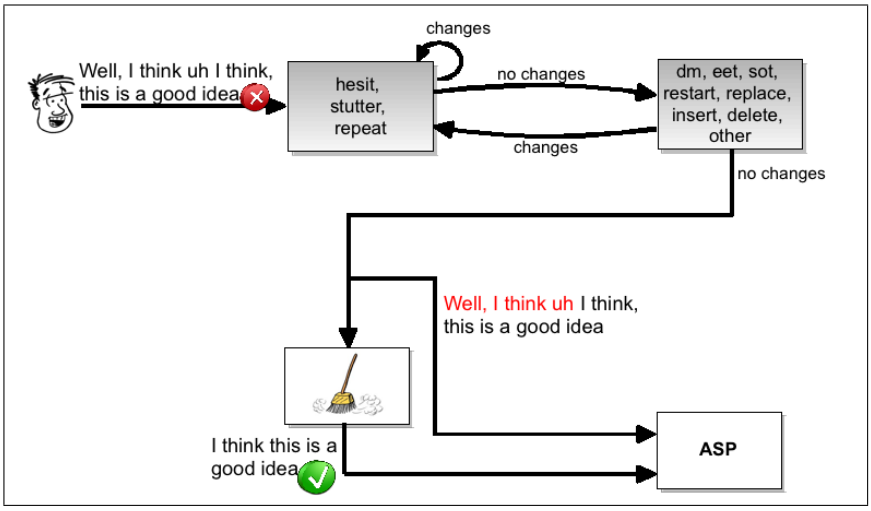

Fig. 2. Design of the hybrid System

5 Experimental Results

This section first describes the evaluation of our experimental results for the individual classification systems with the results that were gained in the development setting and after that the results for the hybrid system where all individual classification methods are combined to create the final architecture as shown in figure 2. The metrics are estimated on a per-word matching of the classified and the corpus-based labeling. As we do not want to present Precision, Recall and F-Scores for all different classes, we combined the results for all classes by a weighted mean for each metric where the weight is the probability of the particular disfluency type. This has the advantage that numerous disfluencies, which are hence more important for the detection approach, get a higher rank than the less frequent ones.

5.1 Individual Results

The evaluation results of the rule-matching based approach is shown in table 4 where both the baseline and the particular results are presented. We can see that the system works very fast (6 seconds detection time for about 10,000 seconds of speech material) and yields a very good outcome with an accuracy of 98.75% and an F-Score of 98.78%.

	Baseline	**RuleMatcher**
Evalinstances	25449	
Evaltime	–	6 sec
Accuracy [%]	93.29	98.75
avg. Precision [%]	87.09	98.78
avg. Recall [%]	93.29	98.75
avg. F-Score [%]	90.08	98.76

Table 4. Results of Rule Matching

For the machine learning approach, which detects disfluencies of type *Discourse Marker, Slip of the Tongue, Explicit Editing Term, Restart, Replacement, Insertion, Deletion, Disruption* and *Other*, we tried several machine learning algorithms to gain the best suited one for this task and, in fact, the **Decision Tree** implementation from the WEKA toolkit outperformed all others. We can see in table 5 that it also works very fast with good performance but needs an immense amount of training time.

	Baseline	**DecisionTree**
Traininstances	98562	
Evalinstances	24728	
Traintime	–	20:41 h
Evaltime	–	7 sec
Accuracy [%]	96.09	97.34
avg. Precision [%]	92.34	97.15
avg. Recall [%]	96.09	97.34
avg. F-Score [%]	94.17	97.24

Table 5. Results for the Machine Learning Approach

5.2 Hybrid Approach

Since we combined the two previously mentioned individual approaches, the hybrid approach is able to detect all their disfluency types. In addition to the

word-based evaluation metrics, we decided to calculate the amount of disfluent dialog acts (see table 6) where a disfluent dialog act is a dialog act that contains at least one disfluency. In our evaluation set, 64% of the dialog acts contained disfluencies with 12.5% of disfluent words. These are the baselines for the particular evaluation metrics which are listed in table 6. There, you can see that our hybrid approach is able to label 92.9% of all words correct which means that it detects more than 56% of all disfluencies. Furthermore, after cleaning the dialog acts of the found disfluencies, the amount of fluent dialog acts increased to more than 77%.

The processing time increased to 1:10 h which is justified by the multi-step design where the transformation of the words to the instances requires a huge amount of time for each of the steps. This could be reduced by omitting the POS tagging by accepting a little degradation of the performance. Nevertheless, the current system still works faster than real-time.

Word Level			**DA Level**		
	Baseline [%]	Result [%]		uncleaned [%]	cleaned [%]
Accuracy	87.5	92.9	fluent	64.3	77.3
Precision	78.4	90.6	disfluent	35.9	22.7
Recall	89.6	90.5			
F-Score	83.6	90.6			
Real Time	2:47 h				
Processing Time	1:10 h				

Table 6. Performance of hybrid Detection and Correction System

5.3 Detection of the Remaining Disfluency Types

So far, we excluded the detection of disfluencies with type *Disruptions*, *Mistake*, *Order* and *Omission* from the hybrid design. Nevertheless, we tried to implement two systems that could deal with the detection of these disfluencies and we want to present them here as well.

The approach that should detect *Disruptions* was also implemented with machine learning based techniques and - for the development setting - produced very good results. The **Decision Tree** outperformed all other algorithms and the particular results with the corresponding baselines are presented in table 7. Unfortunately, by transferring this approach from the development setting into the final system, it's performance crashed and did not yield any detection improvements.

The detection of the remaining disfluencies, which are *Omission, Mistake* and *Order*, was the most difficult task because the speaker did not produce explicit editing terms or any other information about his/her error. A statistical approach like the N-gram technique seemed to be a good way to gain information

	Baseline	Decision Tree
Traininstances	76666	
Evalinstances	19653	
Traintime	–	10:14 h
Testtime	–	4 sec
Accuracy [%]	98.99	99.23
avg. Precision [%]	98.64	99.13
avg. Recall [%]	98.99	99.23
avg. F-Score [%]	98.81	99.18

Table 7. Results for the Disruption Detection

about the correctness of a word-order or a possible missing/superuous word. Unfortunately, the N-gram approach did not yield any detection improvements which is most likely due to the small size of the available corpus. The N-gram statistics have to be estimated on a huge text that must be fluent and from the same context as the evaluation text. Both properties are fulfilled by the training set but it was too small to gain useful N-gram probabilities as seen in the perplexity and out-of-vocabulary values presented in table 3.

6 Conclusions

We have described the implementation and evaluation of a hybrid multi-step system for the detection and correction of disfluencies. We used machine learning techniques as well as rule-based approaches. For the machine learning approach, we estimated a variety of lexical, prosodic, speaker-related and dynamic features. Unfortunately, we had to be aware that the detection of disfluencies with type *Disruption*, *Mistake*, *Order* and *Omission* was not successful and therefore not included in the final system. Despite this, we have shown that the system works and detects and corrects the remaining disfluencies. We reached an Accuracy of 92.9% with an F-Score of 90.6%. Evaluating on the dialog act level, we were able to clean more than 56% of all disfluent dialog acts which resulted in 77.3% clean dialog acts.

6.1 Future Work

The next planned steps are to increase the stability of the machine learning based approaches to ensure their performance in the multi-step hybrid environment. Additionally, we will use a larger text source for the calculation of the N-gram statistics to give the approach for the detection of disfluencies of class *Uncorrected* a better basis for the probability calculation of the correctness of an ongoing utterance. Furthermore, another design of the hybrid approach is thinkable where the different classification methods are used in parallel with a particular weighting to detect whether a word is disfluent or not.

7 Acknowledgment

This work is supported by the European IST Programme Project FP6-0033812 (AMIDA), Publication ID - AMIDA-26. This paper only reflects the authors views and funding agencies are not liable for any use that may be made of the information contained herein. The used POS tags were estimated with the help of the freely available Stanford POS-Tagger and, hence, we want to thank the persons from the Stanford NLP group.
(http://nlp.stanford.edu/software/tagger.shtml)

References

1. Besser, Jana: "A Corpus-Based Approach to the Classification and Correction of Disfluencies in Spontaneous Speech", Master's thesis, Saarbrücken, 2006
2. Ferreira, F., Lau, E., Bailey, K.: "Disfluencies, Language Comprehension and Tree Adjoining Grammars", In: Cognitive Science, Bd. 28, p. 721–749, 2004
3. Jorgensen, Frederik: "The Effects of Disfluency Detection in Parsing Spoken Language", In: NODALIDA-2007, p. 240–244, 2007
4. Stolcke, Andreas, Shriberg, Elizabeth: "Statistical Language Modeling for Speech Disfluencies", In: Proc. ICASSP '96, p. 405–408, 1996
5. Carletta, Jean, Ashby, S., Bourban, S., Flynn, M., et al: "The AMI Meeting Corpus", In: Proceedings of the Measuring Behavior 2005 symposium on "Annotating and measuring Meeting Behaviour", 2005
6. Charniak, E., Johnson, M.: "A TAG-based noisy channel model of speech repairs", In: Annual Meeting of the Association for Computational Linguistics, 2004
7. Lease, M., Johnson, M., Charniak, E.: "Recognizing disfluencies in conversational speech", In: IEEE Transactions on Audio, Speech and Language Processing, p.1566–1573, September 2006
8. Snover, M., Dorr B., Schwartz, R.: "A lexically-driven algorithm for disfluency detection", In: Human Language Technology Conference, 2004
9. Moreno, I., Pineda, L.: "Speech Repairs in the DIME Corpus", 2006
10. Shriberg, E., Bates, R., Stolcke A.: "A Prosody-Only Decision-Tree Model for Disfluency Detection", In: Proc. Eurospeech '97, p. 2383–2386, 1997
11. Finkler, Wolfgang: "Automatische Selbstkorrektur bei der inkrementellen Generierung gesprochener Sprache unter Realzeitbedinungen", Dissertation, Universität des Saarlandes, 165, DISKI, 1997
12. Eklund, Robert: "Disfluency in Swedish human-human and human-machine travel booking dialogues", 2004
13. Witten, I., Frank, E.: "Data Mining: Practical Machine Learning Tools and Techniques" 2nd volume, San Francisco: Morgan Kaufmann, 2005