Domain-specific Classification Methods for Disfluency Detection

Sebastian Germesin, Tilman Becker, Peter Poller

Deutsches **F**orschungszentrum für **K**ünstliche **I**ntelligenz GmbH
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
{germesin, becker, poller}@dfki.de

Abstract

Speech disfluencies are very common in our everyday life and considerably affect NLP systems, which makes systems that can detect or even repair them highly desirable. Previous research achieved good results in the field of disfluency detection but only in subsets of the disfluency types. The aim of this study was to develop a technology that is able to cope with a broad field of disfluency types. A thorough investigation of our corpus led us to a detection design where basic rule-matching techniques are complemented with machine learning and N-gram based approaches. In this paper, we describe the different detection techniques, each specialized on its own disfluency domain and the results we gained.

Index Terms: machine learning, disfluencies, classification, hybrid, lexical rules

1. Introduction

In the last decades, the need for intelligent automatic systems that use natural language processing (NLP) increased enormously. The progressive capacity and power of today's computer systems allow the satisfaction of people's needs towards task oriented speech applications such as meeting supporting systems [1] or online translations [2].

The approach of transferring written language systems to spoken language often results in decreased performance. This is due to the fact that "speech differs from written language" [3]. The main aspect that we consider in this work, are **speech disfluencies** - "syntactical and grammatical [speech] errors" [4] which are based on the incrementality of human speech production [5]. In fact, 5% - 15% of spontaneous 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.

At best, the disfluencies should be repaired or at least marked before the speech material is processed. This could be done via an automatic system that is placed right behind a speech-to-text (STT) system, as shown in Figure 1.

The scheme of the disfluency types this study is based on was developed by [4] 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" [1]. A corpus with over 100 hours of meetings was recorded in the project, focussing on business meetings

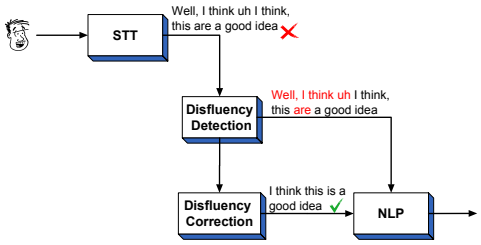

Figure 1: Speech to NLP with Disfluency Detection

(see section 3). All meetings are held in English and participants speak freely in a typical meeting room, yielding a very good reflection of what happens in real meetings (e.g., speech disfluencies).

A number of researchers published different techniques to detect different types of 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.

The difference to the present work is that each of these studies were only focused on a subset of the disfluency types. Our goal was to develop a system that is able to cope with all types of disfluencies and the observed heterogeneity of the disfluencies led us to the assumption that such a system should be designed in a hybrid way, meaning that each disfluency should be detected by a special detection technique that was fine-tuned on this disfluency domain. The effect is a system with reduced computational overhead and also an improved detection performance. The particular techniques that are invented are explained in detail in section 4 and their detection results are presented and discussed in section 5.

2. Disfluency scheme

[10] found that 5% - 10% of our speech is disfluent and, in fact, our corpus even contains about 15% erroneous speech material which can be justified by our available annotation scheme,

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

which has a broad set of disfluency types. We based our work on this disfluency set as a comparison with other schemes showed that this scheme was the best fitting one for our approach with respect to the range of disfluency types and availability.

The common structure of our disfluencies consists of three regions: The **Reparandum** which contains the erroneous speech material, an optional medial region - called the **Interregnum** - containing, e.g., *Hesitations* or *Explicit Editing Terms* and the repairing part called the **Reparans**. [4] states that not all disfluencies fit into that scheme and hence splits up her classification scheme to 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 speaker's 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. Table 1 shows the abbreviations of these classes and examples that help in the understanding of the particular meaning of each disfluency type. The italic written disfluencies are simple disfluencies and the rest are complex disfluencies.

3. Corpus

The AMI Meeting Corpus [1] contains business meetings which are focused on the design of a television remote control. The corpus consists of 135 meetings and 28 of them are annotated with the disfluency scheme described in the last section. We split the disfluency corpus in 80% training set and 20% evaluation set, resulting in an amount of 10.19 and 2.79 hours meeting time. An analysis 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. 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. 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. Althought the system is designed to work in an online environment we just work here on manual transcript data as disfluency annotations on speech recognition ouput is not yet available. This will be the next step.

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

Table 2: N-gram Corpus Statistics

We had to calculate N-grams out of the disfluency annotated training part of the corpus to use them as features in the machine learning approaches and in the statistical N-gram based approach (shown in section 4.3). As the corpus only contains 3760 unique word, this is a relatively small corpus for the estimation of statistical word probabilities. Hence, we obtained limited out-of-vocabulary and perplexity results (see Table 2). Therefore, a corpus with more material is definitely preferable and would lead to better performances.

4. Domain-specific disfluency detection

A thorough investigation of our corpus and the used disfluency scheme showed a heterogeneity with respect to how the different disfluencies can be detected. This led us to the following design: Easily detectable disfluencies should be identified by a simple rule-based approach while the remaining disfluencies need a more sophisticated machine learning approach. Additionally, the disfluencies of the *Uncorrected* group cannot be detected via usual classification approaches as most of their material is missing in the speech and hence we decided to use a statistical N-gram based approach to cope with them. Furthermore, the usage of different detection techniques, each specialized and fine-tuned on its own disfluency domain, yields the advantage of an improved performance in conjunction with a reduced computational overhead at the same time.

4.1. Rule-based approach

The development of rules for the detection of disfluencies starts by separating the easily detectable disfluencies from the ones that need a more complex machine learning approach. Our study showed that *Hesitations*, *Stutterings* and *Repetitions* are the only classes that are well suited for being recognized by rules. This is based in their strict structure which allows a transformation into lexical rules.

The detection of *Hesitations* is easy in the way that there exists only a finite set of 16 words (in our corpus) that count as a *Hesitation* and the top five of them cover more than 98% of all. This means that detecting *Hesitations* is just a word-based matching of these identified words which is shown in Figure 2.

if $word \in [uh, um, mm, hmm, \ldots]$ **then**
  return $''hesit''$;
end if

Figure 2: Algorithm to detect Hesitations

Stutterings are "syllables, speech sounds or single consonants, which are similar to the beginning of the next fully articulated word ... [and] they may neither be equal to the whole next word" [4]. This definition leads to Figure 3 which checks if the current word is "similar" to the next word. This is done by counting the number of equal characters of the current and the next word divided by the length of the current word. If the resulting value exceeds the empirically measured threshold of 0.84 and both words are not equal, the algorithm iden-

tifies the current word as *Stuttering*. Additionally, the pseudocode shows a check for "false-friends" which are words that fit into the described scheme even though they are fluent. To avoid matching them, these often appearing false-friends (in our study: $[we, no, on, so, it]$) are explicitly excluded from the detection.

if $1.0 > getSimilarity(word, nextWord) \geq 0.84$ **then**
  if $!isFalseFriend(word)$ **then**
    return $''stutter''$;
  end if
end if

Figure 3: Algorithm to detect Stutterings

[4] states that *Repetitions* are "expressions that occur several times consecutively ... this denotes both single words and whole phrases, but no word fragments". Transforming this definition directly into a regular expression for the rule-matching yields in `((?: \w+)+)\1 )` which would detect all *Repetitions* but with an enormous computation time and a huge number of false positives. To avoid this, we trimmed the expression in the way that we restrict the number of words we look for and it turns out that a length of 1 to 6 words for the *Reparandum* (2 to 12 words for the whole disfluency) is the best trade-off that we could find. In Figure 4 we can see the corresponding regular expression where `((?: \w+){1,6})` matches up to 6 consecutive words and the `(\1)` detects their repetition. Again, we explicitly exclude some words from the detection algorithm as they are common repetitions that are assumed correct: $[very, okay, hmm, no, right, yes]$.

if $word.matches(''((?:\w+)\{1,6\})(\1\ )'')$ **then**
  if $word.isInReparandum() \wedge !isFalseFriend(word)$ **then**
    return $''repeat''$;
  end if
end if

Figure 4: Algorithm to detect Repetitions

4.2. Machine learning approach

The machine learning approach is implemented with the help of the freely available WEKA toolkit [11] which contains many state-of-the-art machine learning algorithms and a variety of evaluation metrics. Furthermore, it allows to adapt other algorithms due to its simple interface.

We used machine learning based techniques to detect the following disfluency types: *Discourse Marker, Slip of the Tongue, Explicit Editing Term, Restart, Replacement, Insertion, Deletion, Disruption* and *Other*. We had to separate the detection of the *Disruption* class from the remaining ones as it needs a completely different feature set. The *Disruptions* are, according to their definition, classified as a complete segment while the remaining classes are detected word-by-word.

For both machine learning tasks, we trained and evaluated several algorithms to find the most suitable one for the task of the disfluency detection. In fact, the **Decision Tree** implementation of the WEKA toolkit outperformed all other algorithms in accuracy, F-Score and detection time but needs a lot of computation time for the training process. F-Score is a well used metric that is calculated by the harmonic mean of recall and precision. Henceforth, there exists an F-Score value for each class and as presenting all these values would be too much, we decided to combine all single F-Score values by a weighted mean to one average value. All results are presented in section 5. We used four different types of features: **lexical, prosodic, speaker-related** and **dynamic**.

Lexical features are estimated on the word-layer and consider also the Part-of-Speech (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 to its neighbors.

As [10] describes, **prosodic** features are well suited for the disfluency detection task and hence, we use them too. The term prosodic in this context means features that describe the *duration*, *energy*, *pitch* and *velocity* of the words. The *energy* and *pitch* values were normalized with a mean variance normalization per channel to reduce the influence of the microphones. Afterwards, we used these values to compute features like mean, variance and mode of the current word or segment and additionally, contextual features that described the prosodic parallelism of the surrounding elements.

The **speaker-related** features describe the speaker's role, gender, age and native-language as they appear in the corpus. A speaker's role can either be the Industrial Designer (ID), the Project Manager (PM), the Marketing Expert (ME) or the User Interface Designer (UI). These were used because we found a correlation between these characteristics and the rate of disfluent words.

The last type of features are **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. Detection by n-gram-based approach

The detection of the *Uncorrected* disfluencies (*Omission, Mistake* and *Order*) was the most difficult task of this study because the speaker usually does not produce any 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 about the correctness of a word-order or a possible missing or superfluous word.

Using the N-grams directly did not yield any information about the correctness of the ongoing sentence. Therefore, we had to define a more global probability of the correctness of the sentence. Equation 1 shows how we combined the probability of normal word-based N-grams (see Equation 2) and POS N-grams (see Equation 3). Using this definition, we were able to calculate the difference between the "probability of the current sentence" to the particular alterations where, for example, two words get swapped or a word gets inserted.

$$P(s = w_0^{n-1}) = \alpha * P_{POS}^N(s) + (1 - \alpha) * P_W^N(s) \quad (1)$$

$$P_{POS}^N(s) = \sum_{i=N-1}^{n-1} P(POS(w_i)|POS(w_{i-N+1}^{i-1})) \quad (2)$$

$$P_W^N(s) = \sum_{i=N-1}^{n-1} P(w_i|w_{i-N+1}^{i-1}) \quad (3)$$

5. Experimental results

This section describes the results that were gathered on the particular disfluency domains with the respective detection technique. For lack of space, we do only present the confusion matrices of the particular systems and explain evaluation values like the amount of used training instances or the accuracy and

F-Score in the text. The confusion matrices have to be read in the way that the predicted classes are placed in the columns and the actual classes are placed in the rows. The F-Score values are calculated by a weighted mean over all classes where each weight is the rate of the particular class.

5.1. Rule matching

The results for the rule-based approach are shown in Table 3. The approach has a real-time factor of 1:0.0006 which means that it needs on average 0.6 milliseconds for processing one second of speech. Furthermore, the presented results corresponds to an accuracy of 98.8% and an F-Score of about 98.7%. Compared to the baselines of 93.3% and 90.1%, this is a very good result.

fluent	hesit	stutter	repeat	
23601	80	11	49	**fluent**
112	998	0	0	**hesit**
10	0	112	0	**stutter**
49	0	7	410	**repeat**

Table 3: Confusion Matrix of Rule Matching

5.2. Machine learning

We split the machine learning part in two different approaches: One for the detection of *Disruptions* and one for the remaining disfluencies. In both following tables, we present the results from the particular Decision Tree algorithm. For the Disruption Detection approach, we gained an accuracy of 98.99% with an F-Score of 99.23% and from the confusion matrix (see Table 4) we can see that almost half of all disruptions were detected with a false-positives rate of just 0,22%. The results for the machine learning approach that detects the remaining classes show that there exists almost no inter-class confusion. We achieved an accuracy of 97.4% and an F-Score of 97.2% with the corresponding baseline values of 96.1% and 94.2%. The processing times of both approaches are competitive to the one from the rule-matching approach.

fluent	disrupt	
19407	43	**fluent**
108	95	**disrupt**

Table 4: Confusion Matrix for the Disruption Detection Task

5.3. N-gram based approach

Unfortunately, the N-gram approach did not yield any detection improvements. This is most likely due to the small corpus that was available. N-gram statistics have to be estimated on a huge text that has to be fluent and from the same context as the evaluation text. Although both properties are fulfilled by the training set, it was too small to gain useful N-gram probabilities.

fluent	sot	eet	dm	restart	replace	insert	delete	other	
23500	57	3	176	15	7	2	0	3	**fluent**
126	94	0	0	0	2	0	0	0	**sot**
6	0	22	0	0	0	0	0	0	**eet**
81	0	2	329	0	0	0	0	0	**dm**
98	4	0	0	12	0	0	0	0	**restart**
39	0	0	0	0	93	0	0	0	**replace**
28	0	0	0	0	0	9	0	0	**insert**
2	0	0	0	0	0	0	12	0	**delete**
6	0	0	0	0	0	0	0	0	**other**

Table 5: Confusion Matrix of Machine Learning Approach

6. Conclusions

We have described a domain-specific approach to detect speech disfluencies. We first designed a segmentation of the disfluencies in disjoint subsets and after that, used different classification techniques, each fine-tuned on its own subset for the detection. We used rule-based approaches, as well as machine learning techniques and N-gram based algorithms and combined different types of features. Except for the N-gram based approach, our approaches resulted in very competitive performances.

7. Future work

The next step is to combine the presented techniques in one system that can detect and correct the complete set of disfluencies. Additionally, we will use a larger text source for the calculation of the N-gram statistics to give this approach a better basis for the probability calculation of the correctness of an utterance under consideration.

8. 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.

9. References

[1] 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

[2] Wahlster, Wolfgang: Verbmobil: Foundations of Speech-To-Speech Translation, Springer, Berlin - New York, 2000

[3] Eklund, Robert: "Disfluency in Swedish human-human and human-machine travel booking dialogues", PhD thesis, Diss.No. 882, Department of Computer and Information Science, Linkping University, Sweden

[4] Besser, Jana: "A Corpus-Based Approach to the Classification and Correction of Disfluencies in Spontaneous Speech", Bachelor's thesis, Saarbrücken, 2006

[5] Ferreira, F., Lau, E., Bailey, K.: "Disfluencies, Language Comprehension and Tree Adjoining Grammars", In: Cognitive Science, Vol. 28, p. 721–749, 2004

[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", In: Research in Computing Science, Vol. 20, pp. 63, 74, 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] Witten, I., Frank, E.: "Data Mining: Practical Machine Learning Tools and Techniques", 2nd volume, San Francisco, Morgan Kaufmann, 2005