A Lexically-Driven Algorithm for Disfluency Detection

Author 1
Address 1
`email address`

Author 2
Address 2
`email address`

Author 3
Address 3
`email address`

Abstract

This paper describes a transformation-based learning approach to disfluency detection in speech transcripts using primarily lexical features. Our method produces comparable results to two other systems that make heavy use of prosodic features, thus demonstrating that reasonable performance can be achieved without extensive prosodic cues. In addition, we show that it is possible to facilitate the identification of less frequently disfluent discourse markers by taking speaker style into account.

1 Introduction

Disfluencies in human speech are widespread and cause problems for both downstream processing and human readability of speech transcripts. Recent human studies (Jones et al., 2003) have examined the effect of disfluencies on the readability of speech transcripts. These results suggest that the "cleaning" of text by removing disfluent words can increase the speed at which readers can process text. Recent work on detecting edits for use in parsing of speech transcripts (Core and Schubert, 1999) (Charniak and Johnson, 2001) has shown an improvement in the parser error rate by modeling of disfluencies.

Many researchers investigating disfluency detection have focused on the use of prosodic cues, as opposed to lexical features (Nakatani and Hirschberg, 1994). There are different approaches to detecting disfluencies. In one approach, one can first try to locate evidence of a general disfluency, e.g., using prosodic features or language model discontinuations. These locations are called interruption points (IPs). Following this, it is generally sufficient to look in the nearby vicinity of the IP to find the disfluent words. The most successful approaches so far combine the detection of IPs using prosodic features and language modeling techniques (Liu et al., 2003) (Shriberg et al., 2001), (Stolcke et al., 1998).

Our work is based on the premise that the vast majority of disfluencies can be detected using primarily lexical features—specifically the words themselves and part-of-speech labels—without the use of extensive prosodic cues. Lexical modeling of disfluencies with only minimal acoustic cues has been shown to be successful in the past using strongly statistical techniques (Heeman and Allen, 1999). We shall discuss our algorithm and then compare it to two other algorithms that were designed to make extensive use of acoustic features. Our algorithm performs comparably on the tasks assigned and in some cases outperforms systems that used both prosodic and lexical features.

We present an approach to disfluency detection that makes use of Transformation-Based Learning (TBL) to identify simple lexical rules from training that are easily readable and understandable. We use a novel feature for our rules based upon the style of speaker speech to aid in the identification of less frequently disfluent discourse markers. We examine how frequently a speaker uses certain words, such as "like", in order to flag those words that a speaker uses more frequently than other speakers. The system can then learn rules conditioned on these high frequency words. The intuition is that extremely frequent usage of words such as "like" might be a possible clue that these words are being used in a disfluent manner.

We discuss the framework used for annotating disfluencies in Section 2. In Section 3 we shall describe our algorithm and its associated features. Section 4 presents results for our system and two other systems that make heavy use of prosodic features to detect disfluencies. We then discuss the errors made by our system, in Section 5, and discuss our conclusions and future work in Section 6.

2 EARS Disfluency Annotation

One of the major goals of the DARPA program for Effective, Affordable, Reusable Speech-to-Text (EARS) (Wayne, 2003) is to provide a rich transcription of speech recognition output, including speaker identification, sentence boundary detection and the annotation of disfluencies in the transcript (This collection of additional features is also known as Metadata). One of the results of this program has been production of an annotation specification for disfluencies in speech transcripts (Strassel, 2003b) and the transcription of sizable amounts of speech data, both from conversational telephone speech and broadcast news, according to this specification (Strassel, 2003a).

The task of disfluency detection is to distinguish fluent from disfluent words. The EARS MDE (MetaData Extraction) program addresses two types of disfluencies: (i) *edits*—words that were not intended to be said and that are normally replaced with the intended words, such as repeats, restarts, and revisions; and (ii) *fillers*—words with no meaning that are used as discourse markers and pauses, such as "um" and "you know".

Interruption points were also annotated in the data, but were defined as the boundary points of edits and fillers. However, a hesitation is defined as neither an IP nor a disfluency, although it may have similar prosodic properties.

3 The Algorithm

We set out to solve the task of disfluency detection using primarily lexical features in a system we call System A. This section describes the algorithm embodied by the system, including the set of features we use to identify disfluencies.

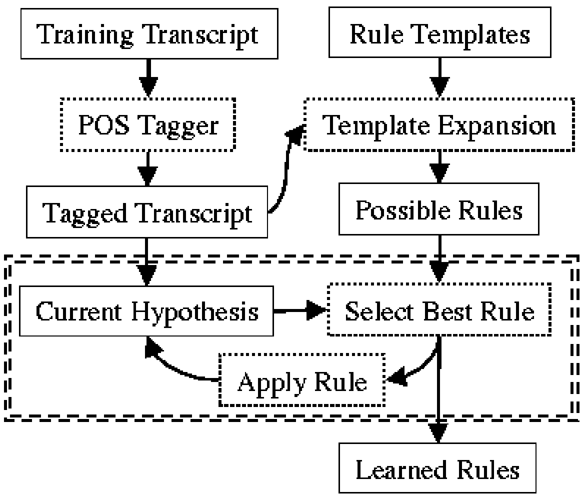

Figure 1: Flow Diagram

The training data for the system are time aligned reference speech transcripts, with speaker identification, sentence boundaries, edits, fillers and interruption points annotated. The input for evaluation is a transcript, either a reference transcript or a speech recognizer output transcript. Some of the evaluation data may be marked with sentence boundaries and speaker identification. The task is to identify which words in the transcript are filler, edits, or fluent. The evaluation data was held out, and not available for tuning system parameters..

The input to System A is a time-aligned transcript of either conversational telephone speech (CTS) or broadcast news speech (BNEWS). In all of the experiments described, the system was trained on reference transcripts, but was tested on both reference and speech output transcripts. The input is processed by the system as shown in Figure 1.

We use a Transformation-Based Learning (Brill, 1995) algorithm to induce rules from the training data. Transformation-Based Learning (TBL) is a technique for learning a set of rules that transform an initial hypothesis for the purpose of reducing the error rate of the hypothesis. The set of possible rules is found by expanding rule templates, which are given as an input. The algorithm greedily selects the rule that reduces the error rate the most, applies it to the data, and then searches for the next rule. To prevent overtraining and speed up the system we used a threshold, which requires a minimum number of error corrections for a rule to be chosen. A threshold of 8 corrections was chosen for CTS training, and a threshold of 5 corrections was chosen for BNEWS training. The algorithm halts when there are no more rules that can reduce the error rate by more than the threshold. The output of the system is an ordered set of rules, which can then be applied to the test data to annotate it for disfluencies.

We allow one of three tags to be assigned to each word: edit, filler or fluent. Since only 15% of the words in conversational speech are disfluent, we begin with the initial hypothesis that all the words in the corpus are fluent. The system then learns rules to relabel words as edits or fillers in order to reduce the number of errors. The rules are iteratively applied to the data.

Table 1 shows the pseudocode for the basic learning algorithm. The system is given the rule templates and the training data as input, and has a training threshold to prevent over training. When the algorithm finishes it returns an ordered set R of rules which can then be used to annotate the test data. The algorithm shown can be sped up dramatically by a number of techniques, which preserve the optimal solution.

```
Let T = Set of Rule Templates
Let D = Training Data
Let R = Ordered Set of Rules. (Empty)
Let P = Set of Possible Rules from T and D
Apply Initial Hypothesis to D
While P is not empty
  r is best rule this iteration.
  r = first element of P
  For every rule p in P
      Calculate corrections of p on D
      if corrections(p) > corrections(r)
         r = p
  If corrections (r) < Training-Threshold
     Halt and Return R
  Apply r to D
  Remove r from P
  Add r to R
Return R
```

Table 1: Pseudo-Code for Basic Learning Algorithm

The algorithm needs access to a set of features, which the rules the system learns can be conditioned on. The set of possible rules and the features each rule uses are specified by the rule templates which are given as an input to the system.

3.1 Feature Set

The rules learned by the system are conditioned on several features of each of the words including the lexeme (the word itself), a part-of-speech tag for the word, whether the word is followed by a silence and whether the word is a *high frequency word*. That is, whether the word is more frequent for this speaker than in the rest of the corpus. The last feature (high frequency of the word) is useful for identifying when words that are usually fluent—but are sometimes disfluent (such as “like”)—are more likely to be disfluencies, with the intuition being that if a speaker is using the word “like” very frequently, then it is likely that the word is being used as a filler. The word “like” for example was only a disfluency 22% of the time it occurred. So a rule that always tags “like” as a disfluency would hurt rather than help the system.

The part-of-speech of the words is assigned using a maximum entropy tagger (Ratnaparkhi, 1996) trained on switchboard data with the additional tags of FP (filled pause) and FRAG (fragment) to represent filled pauses and word fragments. In addition, since contractions such as *he’s* are single words in the rich transcriptions, the two parts of speech assigned to the word, PRP for *he* and VBP for *’s*, were combined for the contraction. Sentence boundary tokens with a sentence boundary part-of-speech were also inserted at the start and end of each sentence in the input.

3.2 Rule Templates

The system was given a set of 33 rule templates, which were used to generate the set of possible rules. Not all rule templates generated rules that were chosen by the system. The set of rule templates that did generate rules chosen by the system are:

1. Change the label of a word **X** from L_1 to L_2.

2. Change the label of word with POS tag **X** from L_1 to L_2.

3. Assign a label of L_1 to the word sequence **X Y**.

4. Assign label L_1 to left side of simple repeat.

5. Change the label of a word with POS **X** from L_1 to L_2 if followed by a word with POS **Y**.
6. Change the label of a word **X** from L_1 to L_2 if followed by the word **Y**.
7. Change the label of a word from L_1 to L_2 if followed by words **X Y**.
8. Change the label of a word from L_1 to L_2 if preceded by words **X Y**.
9. Change the label of a word **X** with POS **Y** from L_1 to L_2.
10. If the words match the pattern A^* POS **X** B^* A^* then assign the first A^* label L_1. A^* and B^* can be any words and are not instantiated by the system.
11. Assign label L_1 to the left side of repeat, separated by a word with POS **X**.
12. Assign label L_1 to the left side of a partial repeat, followed (or not followed) by a pause. A partial repeat is a repeat of length 4 or more with at most one difference in the words.
13. Assign a label L_1 to words **X Y** if followed by the word **Z**.
14. Assign a label L_1 to words **X Y** if preceded by the word **Z**.
15. Change the label of a word with POS **X** from L_1 to L_2 if preceded by a word with POS **Y**.
16. Change the label of a word **X** from L_1 to L_2 if preceded by the word **Y**.
17. Change the label of a word **X** from L_1 to L_2 if followed by words **Y Z**.
18. Change the label of a word with POS **X** from L_1 to L_2 if followed by words with POS tags **Y Z**.
19. Change the label of a word **X** from L_1 to L_2 if preceded by the word **Y** and followed by the word **Z**.
20. Change the label of a word with POS **X** from L_1 to L_2 if preceded by a word with POS **Y** and followed by a word with POS **Z**.
21. Change the label of a word **X** from L_1 to L_2 if followed (or not followed) by a silence.
22. Change the label of a word with POS **X** from L_1 to L_2 if followed (or not followed) by a silence.
23. Change the label of a word **X** from L_1 to L_2 if followed (or not followed) by a silence and followed by a word **Y**.
24. Change the label of a word with POS **X** from L_1 to L_2 if followed (or not followed) by a silence and followed by a word with POS **Y**.
25. Change the label of a word **X** that is (or is not) a high frequency word for the speaker from L_1 to L_2.

3.3 Sample Rules

Some sample rules that would be derived by the templates (and are actually learned by the system) are listed below.

- Change the label of a word with POS **FP** (filled pause) from **fluent** to **filler**. (Instantiated from template 2.)
- Assign label **filler** to the words **you know**. (Instantiated from template 3.)
- Assign label **edit** to left side of simple repeat. (Instantiated from template 4.)
- If the words match the pattern A^* POS **FP** B^* A^*. then assign the first A^* label **edit**. (Instantiated from template 10.) For example this would apply to the word sequence “car uh red car”, assuming “uh” is tagged as a filled pause.

4 Results

All of the results in this section are from training and evaluation on data produced by the Linguistic Data Consortium (LDC) for the EARS Metadata community. There were 75,182 utterences, containing a total of 491,543 tokens in the CTS training set. The BNEWS training set consisted of 21,551 utterences containing 189,766 tokens. There were 4820 utterences, containing a total of 33,670 tokens in the CTS evaluation set. The BNEWS evaluation set was much smaller and consisted of only 984 utterences with 14,544 tokens.

In the following results, we compare our System A to two other systems that were designed for the same task, System B and System C. System C was only applied to conversational speech, so there are no results for it on broadcast news transcripts. Our system was also given the same speech recognition output as System C for the conversational speech condition, whereas System B used transcripts produced by a different speech recognition system.

System B used both prosodic cues and lexical information to detect disfluencies. The prosodic cues were modeled by a decision tree classifier, whereas the lexical information was modeled using a hidden language model. The hidden language model used was a 4-gram language model, separately trained for both CTS and BNEWS.

System C also used transformation-based learning to model disfluencies. The initial step of the system does not use TBL but rather relies on the detection of interruption points using a decision-tree classifier. The decision tree uses both prosodic and lexical features. The IPs detected were then inserted into the text as additional words, so that the transformation-based learning system could use them to identify disfluencies. System C's feature set included the identity of the word, the POS label for a word, whether it was commonly used as any of the subtypes of fillers or edits, and whether it was commonly a back-channel word. To model repeated words, each word had a flag indicating whether the word or part-of-speech matches the word 1, 2, or 3 to its right. Turn and segment boundary flags were also used by the system. Whereas System A only attempted to learn three labels, filler, edit and fluent, System C attempted to learn many subtypes of disfluencies, which were not distinguished in the evaluation.

4.1 Lexeme Error Rate

We use *Lexeme Error Rate* (LER) as a measure of effectiveness. This measure is the same as the traditional word-error rate used in speech recognition, except that filled pauses and fragments are not optionally deletable in lexeme error rate. The LERs of the speech transcripts used by the three systems were all fairly similar and are shown in Table 2.

System	CTS	BNEWS
System A	25.3%	11.7%
System B	24.9%	12.7%
System C	25.2%	-

Table 2: Lexeme Error Rates

4.2 Top Rules Learned

A total of 106 rules were learned by the system for conversational telephone speech. The top 20 rules learned by the system for CTS are listed below.

1. Label all fluent filled pauses as fillers.
2. Label the left side of a simple repeat as an edit.
3. Label "you know" as a filler.
4. Label fluent "well"s with a UH part-of-speech as a filler.
5. Label fluent fragments as edits.
6. Label "I mean" as a filler.
7. Label the left side of a simple repeat separated by a filled pause as an edit.
8. Label the left side of a simple repeat separated by a fragment as an edit.
9. Label edit filled pauses as fillers.
10. Label edit fragments at end of sentence as fluent.
11. If the words match the pattern A* pronoun B* A*, label the first A* as an edit.
12. Label fluent pronoun followed by a contracted pronoun (such as "he's", PRP+VBP) as an edit.
13. Label fluent pronoun followed by a contracted pronoun (PRP+BES) as an edit.
14. Label fluent "see" with a "UH" part-of-speech as a filler.
15. Label fluent pronoun followed by a RB and PRP part-of-speech as an edit.
16. Label fluent "now" at start of sentence as a filler.

17. Label fluent contracted pronoun (PRP+VBP) followed by a pronoun as an edit.

18. Label fluent contracted pronoun (PRP+BES) followed by a pronoun as an edit.

19. Label edit filled pause followed by a pronoun as a filler.

20. If the words match the pattern A* fragment B* A*, label the first A* as an edit.

The majority of the error reduction is done by the top rules. On the CTS training data the top 5 rules were responsible for correcting 86% of the errors that system was able to fix. The top ten rules were responsible for 94% of the system's performance, and the top twenty rules were responsible for 96% of the system's performance.

All systems were evaluated using rteval (Rich Transcription Evaluation) version 2.3 (Kubala and Srivastava, 2003). Rteval aligns the system output to the annotated reference transcripts in such a way as to minimize the lexeme error rate. In the event that there are multiple alignments with the same lexeme error rate, then the alignment that minimizes the disfluency error rate is chosen. The error rate is the number of disfluency errors (insertions and deletions) divided by the number of disfluent tokens in the reference transcript. Edit and filler errors are calculated separately. The results of the evaluation are shown in Table 3.

Data	System	Edit Err	Filler Err
CTS Reference	**A**	68.0%	18.1%
	B	59.0%	18.2%
	C	75.1%	23.2%
CTS Speech	**A**	87.9%	48.8%
	B	87.5%	46.9%
	C	88.5%	51.0%
BNews Reference	**A**	45.3%	6.5%
	B	44.2%	7.9%
BNews Speech	**A**	93.9%	57.2%
	B	96.1%	50.4%

Table 3: Disfluency Detection Results

Many of the small differences in the CTS results were not found to be significantly different. The broadcast news data were not large enough to use as the basis of significance tests.

5 Error Analysis

It is clear from the discrepancies between the reference and speech condition that a large portion of the errors (a majority except in the case of edit detection for CTS) are due to errors in the STT (Speech-To-Text). This is most notable for fillers in broadcast news where the error rate for our systemincreases from 6.5% to 57.2%. Such a trend can be seen for the other systems, indicating that the prosodic models in the other systems were not more robust to the lexical errors, despite the fact that errors in the STT should not affect the prosodic models.

All three systems produced comparable results on all of the conditions, with the only large exception being edit detection for CTS Reference, where System B had an error rate of 59% compared to our system's error rate of 68%. This is possibly due to the prosodic model employed by System B, though no significant gain was shown for the other conditions.

The speech output condition suffers from several types of errors due to errors in the transcript produced by the speech transcription system. First, the system can output the wrong word causing it to be misannotated. 27% of our edit errors in CTS and 19% of our filler errors occurred when the STT system misrecognized the word. The recognizer can also hallucinate words which do not correspond to words in the reference transcripts. If a filled pause is hallucinated in this manner, the disfluency detection system will always annotate it as filler, which results in a false insertion error. Another form of error is caused when the recognizer fails to output any token corresponding to a reference token. Since no word was output it is impossible to annotate anything, and a deletion is caused. 19% of our edit error and 12% of our filler error occured when the word was not output by the system. Finally errors in the context words surrounding disfluencies can affect disfluency detection as well.

One possible method to correct for the STT errors would be to train our system on speech output from the recognizer rather than on reference transcripts. This would allow our system to learn to correct for the errors the recognizer makes. Another option would be to use a word recognition confidence score from the recognizer as a feature in the TBL system; such scores were not output by system and thus could not be incorporated at this time. A more systematic analysis of the errors caused by the recognizer and their effect on disfluencies also needs to be performed.

System A has a much higher error for edits than fillers, due, in large part, to the presence of long, difficult to detect edits. Consider the following word sequence: "*[and whenever they come out with a warning] you know they were coming out with a warning about trains* ". The portion within square brackets is the edit to be detected. The difficulty in finding such regions is that the edit itself appears very fluent. One can identify these regions by examining what comes after the edit and finding that is highly similar in content to the edit region. Prosodic features can be useful in identifying the interruption point at which the edit ends, but the degree to which the edit extends backwards from this point still needs to be identified. Long distance dependencies should reveal the edit region, and it is possible that parsing or semantic analysis of the text would be a useful technique to employ. In addition there are other cues such as the filler "you know" after the edit which can be used to locate these edit regions. Long edit regions (of length four or more) are responsible for 48% of the edit errors in the CTS reference condition for our system.

Another source of error is the misidentification of sentence boundaries in the transcript. Since these boundaries are identified by other algorithms before disfluencies are detected, errors in those steps can cause additional errors by our system. Incorrect boundaries can cause the context of possible disfluencies to change, causing them to be mislabeled. Consider rule 10 learned by our system for conversational speech, which states that fragments at the end of a sentence should be labeled as fluent. If the sentence boundary is wrong after a fragment then the fragment would be mislabeled, either as fluent if a boundary was inserted or as an edit if the boundary was deleted.

6 Conclusions and Future Work

We have presented a transformation-based learning approach to detecting disfluencies that uses primarily lexical features. Our system performed comparably with other systems that relied on both prosodic and lexical features. Our speaker style (high frequency word) feature enabled us to detect rarer disfluencies, although this was not a large factor in our performance. It does appear to be a promising technique for future research however.

The technique described here shows promise for extension to disfluency detection in other languages, such as Chinese and Arabic. Since Transformation Based Learning is a weakly statistical technique, it does not require a large amount of training data and could be more rapidly applied to new languages. Assuming the basic forms of disfluencies in other languages is similar to that in English, very few modifications would be required of the system for the adaption since the system has few language specific assumptions built-in.

The longer edits that the system currently misses may be detectable using parsing, with the intuition that a parser trained on fluent speech may perform poorly in the presence of longer edits. Techniques using parse trees to identify disfluencies have shown success in the past (Hindle, 1983). The system could use portions of the parse structure as features and could relabel entire subtrees of the parse tree. Repeated words are another feature of the longer edits, which we might leverage off of by performing a weighted alignment of the edit and the repair. Eventually it may prove that more elaborate acoustic cues will be needed to identify these edits, at which point a model of interruption points could be included as a feature in the rules learned by the system.

References

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

Eugene Charniak and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of the NAACL*.

Mark G. Core and Lenhart K. Schubert. 1999. A model of speech repairs and other disruptions. In Susan E.

Brennan, Alain Giboin, and David Traum, editors, *Working Papers of the AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems*, pages 48–53, Menlo Park, California. AAAI.

Peter Heeman and James Allen. 1999. Speech repairs, intonational phrases, and discourse marker: Modeling speakers' utterances in spoken dialogue. *Computational Linguistics*, 25(4).

Donald Hindle. 1983. Deterministic parsing of syntactic non-fluencies. In *Proceedings of ACL*, pages 123–128.

Douglas Jones, Florian Wolf, Edward Gibson, Elliott Williams, Evelina Fedorenko, Douglas Reynolds, and Marc Zissman. 2003. Measuring the readability of automatic speech-to-text transcripts. In *Proceedings of Eurospeech*, Geneva.

Francis Kubala and Amit Srivastava. 2003. *A Framework for Evaluating Rich Transcription Technology*. BBN Ears Website. http://www.speech.bbn.com/ears.

Yang Liu, Elizabeth Shriberg, and Andreas Stolcke. 2003. Automatic disfluency identification in coversational speech using multiple knowledge sources. In *Proceedings of Eurospeech*, Geneva.

Christine Nakatani and Julia Hirschberg. 1994. A corpus-based study of repair cue in spontaneous speech. *Journal of the Acoustical Society of America*, 95(3):160–1616.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *ACL-SIGDAT Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142, Philadelphia, PA.

Elizabeth Shriberg, Andreas Stolcke, and Dan Baron. 2001. Can prosody aid the automatic processing of multi-party meetings? evidence from predicting punctuation, disfluencies, and overlapping speech. In *Proceedings of ISCA Tutorial and Research Workshop on Prosody in Speech Recognition and Understanding*, pages 139–146, Red Bank, NJ.

Andreas Stolcke, Elizabeth Shriberg, Rebecca Bates, Mari Ostendorf, Dilek Hakkani, Madelaine Plauche, Gokhan Tur, and Yu Lu. 1998. Automatic detection of sentence boundaries and disfluencies based on recognized words. In *Proceedings of the ICSLP*, volume 5, pages 2247–2250, Sydney, Australia.

Stephanie Strassel. 2003a. *Guidelines for RT-03 Transcription – Version 2.2*. Linguistic Data Consortium, Universitry of Pennsylvannia.

Stephanie Strassel. 2003b. *Simple Metadata Annotation Specification – Version 5.0*. Linguistic Data Consortium, Universitry of Pennsylvannia.

Charles Wayne. 2003. *Effective, Affordable, Reusable Speech-to-Text (EARS)*. Official web site for DARPA/EARS Program. http://www.darpa.muk/iao/EARS.htm.