The impact of language models and loss functions on repair disfluency detection

Simon Zwarts and Mark Johnson
Centre for Language Technology
Macquarie University
{simon.zwarts|mark.johnson|}@mq.edu.au

Abstract

Unrehearsed spoken language often contains disfluencies. In order to correctly interpret a spoken utterance, any such disfluencies must be identified and removed or otherwise dealt with. Operating on transcripts of speech which contain disfluencies, we study the effect of language model and loss function on the performance of a linear reranker that rescores the 25-best output of a noisy-channel model. We show that language models trained on large amounts of non-speech data improve performance more than a language model trained on a more modest amount of speech data, and that optimising f-score rather than log loss improves disfluency detection performance.

Our approach uses a log-linear reranker, operating on the top n analyses of a noisy channel model. We use large language models, introduce new features into this reranker and examine different optimisation strategies. We obtain a disfluency detection f-scores of 0.838 which improves upon the current state-of-the-art.

1 Introduction

Most spontaneous speech contains disfluencies such as partial words, filled pauses (e.g., "uh", "um", "huh"), explicit editing terms (e.g., "I mean"), parenthetical asides and repairs. Of these, repairs pose particularly difficult problems for parsing and related Natural Language Processing (NLP) tasks. This paper presents a model of disfluency detection based on the noisy channel framework, which specifically targets the repair disfluencies. By combining language models and using an appropriate loss function in a log-linear reranker we are able to achieve f-scores which are higher than previously reported.

Often in natural language processing algorithms, more data is more important than better algorithms (Brill and Banko, 2001). It is this insight that drives the first part of the work described in this paper. This paper investigates how we can use language models trained on large corpora to increase repair detection accuracy performance.

There are three main innovations in this paper. First, we investigate the use of a variety of language models trained from text or speech corpora of various genres and sizes. The largest available language models are based on written text: we investigate the effect of written text language models as opposed to language models based on speech transcripts. Second, we develop a new set of reranker features explicitly designed to capture important properties of speech repairs. Many of these features are lexically grounded and provide a large performance increase. Third, we utilise a loss function, approximate expected f-score, that explicitly targets the asymmetric evaluation metrics used in the disfluency detection task. We explain how to optimise this loss function, and show that this leads to a marked improvement in disfluency detection. This is consistent with Jansche (2005) and Smith and Eisner (2006), who observed similar improvements when using approximate f-score loss for other problems. Similarly we introduce a loss function based on the edit-f-score in our domain.

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 703–711,
Portland, Oregon, June 19-24, 2011.

Together, these three improvements are enough to boost detection performance to a higher f-score than previously reported in literature. Zhang et al. (2006) investigate the use of 'ultra large feature spaces' as an aid for disfluency detection. Using over 19 million features, they report a final f-score in this task of 0.820. Operating on the same body of text (Switchboard), our work leads to an f-score of 0.838, this is a 9% relative improvement in residual f-score.

The remainder of this paper is structured as follows. First in Section 2 we describe related work. Then in Section 3 we present some background on disfluencies and their structure. Section 4 describes appropriate evaluation techniques. In Section 5 we describe the noisy channel model we are using. The next three sections describe the new additions: Section 6 describe the corpora used for language models, Section 7 describes features used in the log-linear model employed by the reranker and Section 8 describes appropriate loss functions which are critical for our approach. We evaluate the new model in Section 9. Section 10 draws up a conclusion.

2 Related work

A number of different techniques have been proposed for automatic disfluency detection. Schuler et al. (2010) propose a Hierarchical Hidden Markov Model approach; this is a statistical approach which builds up a syntactic analysis of the sentence and marks those subtrees which it considers to be made up of disfluent material. Although they are interested not only in disfluency but also a syntactic analysis of the utterance, including the disfluencies being analysed, their model's final f-score for disfluency detection is lower than that of other models.

Snover et al. (2004) investigate the use of purely lexical features combined with part-of-speech tags to detect disfluencies. This approach is compared to approaches which use primarily prosodic cues, and appears to perform equally well. However, the authors note that this model finds it difficult to identify disfluencies which by themselves are very fluent. As we will see later, the individual components of a disfluency do not have to be disfluent by themselves. This can occur when a speaker edits her speech for meaning-related reasons, rather than errors that arise from performance. The edit repairs which are the focus of our work typically have this characteristic.

Noisy channel models have done well on the disfluency detection task in the past; the work of Johnson and Charniak (2004) first explores such an approach. Johnson et al. (2004) adds some handwritten rules to the noisy channel model and use a maximum entropy approach, providing results comparable to Zhang et al. (2006), which are state-of-the art results.

Kahn et al. (2005) investigated the role of prosodic cues in disfluency detection, although the main focus of their work was accurately recovering and parsing a fluent version of the sentence. They report a 0.782 f-score for disfluency detection.

3 Speech Disfluencies

We follow the definitions of Shriberg (1994) regarding speech disfluencies. She identifies and defines three distinct parts of a speech disfluency, referred to as the *reparandum*, the *interregnum* and the *repair*. Consider the following utterance:

$$\overbrace{\underbrace{\textit{I want a flight}}_{}\ \textit{to Boston}}^{\text{reparandum}},\ \underbrace{\textit{uh, I mean}}_{\text{interregnum}}\ \underbrace{\textit{to Denver}}_{\text{repair}}\ \textit{on Friday} \tag{1}$$

The reparandum *to Boston* is the part of the utterance that is 'edited out'; the interregnum *uh, I mean* is a filled pause, which need not always be present; and the repair *to Denver* replaces the reparandum.

Shriberg and Stolcke (1998) studied the location and distribution of repairs in the Switchboard corpus (Godfrey and Holliman, 1997), the primary corpus for speech disfluency research, but did not propose an actual model of repairs. They found that the overall distribution of speech disfluencies in a large corpus can be fit well by a model that uses only information on a very local level. Our model, as explained in section 5, follows from this observation.

As our domain of interest we use the Switchboard corpus. This is a large corpus consisting of transcribed telephone conversations between two partners. In the Treebank III (Marcus et al., 1999) corpus there is annotation available for the Switchboard corpus, which annotates which parts of utterances

are in a reparandum, interregnum or repair.

4 Evaluation metrics for disfluency detection systems

Disfluency detection systems like the one described here identify a subset of the word tokens in each transcribed utterance as "edited" or disfluent. Perhaps the simplest way to evaluate such systems is to calculate the accuracy of labelling they produce, i.e., the fraction of words that are correctly labelled (i.e., either "edited" or "not edited"). However, as Charniak and Johnson (2001) observe, because only 5.9% of words in the Switchboard corpus are "edited", the trivial baseline classifier which assigns all words the "not edited" label achieves a labelling accuracy of 94.1%.

Because the labelling accuracy of the trivial baseline classifier is so high, it is standard to use a different evaluation metric that focuses more on the detection of "edited" words. We follow Charniak and Johnson (2001) and report the f-score of our disfluency detection system. The f-score f is:

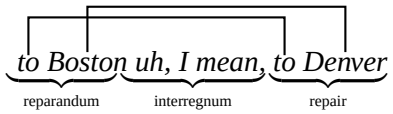
$$f = \frac{2c}{g+e} \quad (2)$$

where g is the number of "edited" words in the gold test corpus, e is the number of "edited" words proposed by the system on that corpus, and c is the number of the "edited" words proposed by the system that are in fact correct. A perfect classifier which correctly labels every word achieves an f-score of 1, while the trivial baseline classifiers which label every word as "edited" or "not edited" respectively achieve a very low f-score.

Informally, the f-score metric focuses more on the "edited" words than it does on the "not edited" words. As we will see in section 8, this has implications for the choice of loss function used to train the classifier.

5 Noisy Channel Model

Following Johnson and Charniak (2004), we use a noisy channel model to propose a 25-best list of possible speech disfluency analyses. The choice of this model is driven by the observation that the repairs frequently seem to be a "rough copy" of the reparandum, often incorporating the same or very similar words in roughly the same word order. That is, they seem to involve "crossed" dependencies between the reparandum and the repair. Example (3) shows the crossing dependencies. As this example also shows, the repair often contains many of the same words that appear in the reparandum. In fact, in our Switchboard training corpus we found that 62reparandum also appeared in the associated repair,



(3)

5.1 Informal Description

Given an observed sentence Y we wish to find the most likely source sentence $\hat{X}$, where

$$\hat{X} = \operatorname*{argmax}_X P(Y|X)P(X) \quad (4)$$

In our model the unobserved X is a substring of the complete utterance Y.

Noisy-channel models are used in a similar way in statistical speech recognition and machine translation. The language model assigns a probability $P(X)$ to the string X, which is a substring of the observed utterance Y. The channel model $P(Y|X)$ generates the utterance Y, which is a potentially disfluent version of the source sentence X. A repair can potentially begin before any word of X. When a repair has begun, the channel model incrementally processes the succeeding words from the start of the repair. Before each succeeding word either the repair can end or else a sequence of words can be inserted in the reparandum. At the end of each repair, a (possibly null) interregnum is appended to the reparandum.

We will look at these two components in the next two Sections in more detail.

5.2 Language Model

Informally, the task of language model component of the noisy channel model is to assess fluency of the sentence with disfluency removed. Ideally we would like to have a model which assigns a very high probability to disfluency-free utterances and a lower probability to utterances still containing disfluencies. For computational complexity reasons, as described in the next section, inside the noisy channel model we use a bigram language model. This

bigram language model is trained on the fluent version of the Switchboard corpus (training section).

We realise that a bigram model might not be able to capture more complex language behaviour. This motivates our investigation of a range of additional language models, which are used to define features used in the log-linear reranker as described below.

5.3 Channel Model

The intuition motivating the channel model design is that the words inserted into the reparandum are very closely related to those in the repair. Indeed, in our training data we find that 62% of the words in the reparandum are exact copies of words in the repair; this identity is strong evidence of a repair. The channel model is designed so that exact copy reparandum words will have high probability.

Because these repair structures can involve an unbounded number of crossed dependencies, they cannot be described by a context-free or finite-state grammar. This motivates the use of a more expressive formalism to describe these repair structures.

We assume that X is a substring of Y, i.e., that the source sentence can be obtained by deleting words from Y, so for a fixed observed utterance Y there are only a finite number of possible source sentences. However, the number of possible source sentences, X, grows exponentially with the length of Y, so exhaustive search is infeasible. Tree Adjoining Grammars (TAG) provide a systematic way of formalising the channel model, and their polynomial-time dynamic programming parsing algorithms can be used to search for likely repairs, at least when used with simple language models like a bigram language model. In this paper we first identify the 25 most likely analyses of each sentence using the TAG channel model together with a bigram language model.

Further details of the noisy channel model can be found in Johnson and Charniak (2004).

5.4 Reranker

To improve performance over the standard noisy channel model we use a reranker, as previously suggest by Johnson and Charniak (2004). We rerank a 25-best list of analyses. This choice is motivated by an oracle experiment we performed, probing for the location of the best analysis in a 100-best list. This experiment shows that in 99.5% of the cases the best analysis is located within the first 25, and indicates that an f-score of 0.958 should be achievable as the upper bound on a model using the first 25 best analyses. We therefore use the top 25 analyses from the noisy channel model in the remainder of this paper and use a reranker to choose the most suitable candidate among these.

6 Corpora for language modelling

We would like to use additional data to model the fluent part of spoken language. However, the Switchboard corpus is one of the largest widely-available disfluency-annotated speech corpora. It is reasonable to believe that for effective disfluency detection Switchboard is not large enough and more text can provide better analyses. Schwartz et al. (1994), although not focusing on disfluency detection, show that using written language data for modelling spoken language can improve performance. We turn to three other bodies of text and investigate the use of these corpora for our task, disfluency detection. We will describe these corpora in detail here.

The predictions made by several language models are likely to be strongly correlated, even if the language models are trained on different corpora. This motivates the choice for log-linear learners, which are built to handle features which are not necessarily independent. We incorporate information from the external language models by defining a reranker feature for each external language model. The value of this feature is the log probability assigned by the language model to the candidate underlying fluent substring X

For each of our corpora (including Switchboard) we built a 4-gram language model with Kneser-Ney smoothing (Kneser and Ney, 1995). For each analysis we calculate the probability under that language model for the candidate underlying fluent substring X. We use this log probability as a feature in the reranker. We use the SRILM toolkit (Stolcke, 2002) both for estimating the model from the training corpus as well as for computing the probabilities of the underlying fluent sentences X of the different analysis.

As previously described, **Switchboard** is our pri-

mary corpus for our model. The language model part of the noisy channel model already uses a bigram language model based on Switchboard, but in the reranker we would like to also use 4-grams for reranking. Directly using Switchboard to build a 4-gram language model is slightly problematic. When we use the training data of Switchboard both for language fluency prediction and the same training data also for the loss function, the reranker will overestimate the weight associated with the feature derived from the Switchboard language model, since the fluent sentence itself is part of the language model training data. We solve this by dividing the Switchboard training data into 20 folds. For each fold we use the 19 other folds to construct a language model and then score the utterance in this fold with that language model.

The largest widely-available corpus for language modelling is the **Web 1T 5-gram** corpus (Brants and Franz, 2006). This data set, collected by Google Inc., contains English word n-grams and their observed frequency counts. Frequency counts are produced from this billion-token corpus of web text. Because of the noise[1] present in this corpus there is an ongoing debate in the scientific community of the use of this corpus for serious language modelling.

The **Gigaword** Corpus (Graff and Cieri, 2003) is a large body of newswire text. The corpus contains $1.6 \cdot 10^9$ tokens, however fluent newswire text is not necessarily of the same domain as disfluency removed speech.

The **Fisher** corpora Part I (David et al., 2004) and Part II (David et al., 2005) are large bodies of transcribed text. Unlike Switchboard there is no disfluency annotation available for Fisher. Together the two Fisher corpora consist of $2.2 \cdot 10^7$ tokens.

7 Features

The log-linear reranker, which rescores the 25-best lists produced by the noisy-channel model, can also include additional features besides the noisy-channel log probabilities. As we show below, these additional features can make a substantial improvement to disfluency detection performance. Our reranker incorporates two kinds of features. The first are log-probabilities of various scores computed by the noisy-channel model and the external language models. We only include features which occur at least 5 times in our training data.

The noisy channel and language model features consist of:

1. LMP: 4 features indicating the probabilities of the underlying fluent sentences under the language models, as discussed in the previous section.

2. NCLogP: The Log Probability of the entire noisy channel model. Since by itself the noisy channel model is already doing a very good job, we do not want this information to be lost.

3. LogFom: This feature is the log of the "figure of merit" used to guide search in the noisy channel model when it is producing the 25-best list for the reranker. The log figure of merit is the sum of the log language model probability and the log channel model probability plus 1.5 times the number of edits in the sentence. This feature is redundant, i.e., it is a linear combination of other features available to the reranker model: we include it here so the reranker has direct access to all of the features used by the noisy channel model.

4. NCTransOdd: We include as a feature parts of the noisy channel model itself, i.e. the channel model probability. We do this so that the task to choosing appropriate weights of the channel model and language model can be moved from the noisy channel model to the log-linear optimisation algorithm.

The boolean indicator features consist of the following 3 groups of features operating on words and their edit status; the latter indicated by one of three possible flags: _ when the word is not part of a disfluency or E when it is part of the reparandum or I when it is part of the interregnum.

1. CopyFlags_X_Y: When there is an exact copy in the input text of length X ($1 \leq X \leq 3$) and the gap between the copies is Y ($0 \leq Y \leq 3$) this feature is the sequence of flags covering the two copies. Example: `CopyFlags_1_0 (E`

[1] We do not mean speech disfluencies here, but noise in web-text; web-text is often poorly written and unedited text.

_) records a feature when two identical words are present, directly consecutive and the first one is part of a disfluency (**E**dited) while the second one is not. There are 745 different instances of these features.

2. WordsFlags_L_n_R: This feature records the immediate area around an n-gram ($n \leq 3$). L denotes how many flags to the left and R ($0 \leq R \leq 1$) how many to the right are includes in this feature (Both L and R range over 0 and 1). Example: `WordsFlags_1_1_0 (need _)` is a feature that fires when a fluent word is followed by the word 'need' (one flag to the left, none to the right). There are 256808 of these features present.

3. SentenceEdgeFlags_B_L: This feature indicates the location of a disfluency in an utterance. The Boolean B indicates whether this features records sentence initial or sentence final behaviour, L ($1 \leq L \leq 3$) records the length of the flags. Example `SentenceEdgeFlags_1_1 (I)` is a feature recording whether a sentence ends on an interregnum. There are 22 of these features present.

We give the following analysis as an example:

but E *but _ that _ does _ n't _ work _*

The language model features are the probability calculated over the fluent part. NCLogP, LogFom and NCTransOdd are present with their associated value. The following binary flags are present:
`CopyFlags_1_0 (E _)`
`WordsFlags:0:1:0 (but E)`
`WordsFlags:0:1:0 (but _)`
`WordsFlags:1:1:0 (E but _)`
`WordsFlags:1:1:0 (_ that _)`
`WordsFlags:0:2:0 (but E but _)` etc.[2]
`SentenceEdgeFlags:0:1 (E)`
`SentenceEdgeFlags:0:2 (E _)`
`SentenceEdgeFlags:0:3 (E _ _)`

These three kinds of boolean indicator features together constitute the *extended feature set*.

[2] An exhaustive list here would be too verbose.

8 Loss functions for reranker training

We formalise the reranker training procedure as follows. We are given a training corpus T containing information about n possibly disfluent sentences. For the ith sentence T specifies the sequence of words x_i, a set $\mathcal{Y}_i$ of 25-best candidate "edited" labellings produced by the noisy channel model, as well as the correct "edited" labelling $y_i^\star \in \mathcal{Y}_i$.[3]

We are also given a vector $\mathbf{f} = (f_1, \ldots, f_m)$ of *feature functions*, where each f_j maps a word sequence x and an "edit" labelling y for x to a real value $f_j(x, y)$. Abusing notation somewhat, we write $\mathbf{f}(x, y) = (f_1(x, y), \ldots, f_m(x, y))$. We interpret a vector $\mathbf{w} = (w_1, \ldots, w_m)$ of *feature weights* as defining a conditional probability distribution over a candidate set $\mathcal{Y}$ of "edited" labellings for a string x as follows:

$$\mathrm{P}_{\mathbf{w}}(y \mid x, \mathcal{Y}) = \frac{\exp(\mathbf{w} \cdot \mathbf{f}(x, y))}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w} \cdot \mathbf{f}(x, y'))}$$

We estimate the feature weights $\mathbf{w}$ from the training data T by finding a feature weight vector $\widehat{\mathbf{w}}$ that optimises a regularised objective function:

$$\widehat{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}} L_T(\mathbf{w}) + \alpha \sum_{j=1}^{m} w_j^2$$

Here α is the *regulariser weight* and L_T is a *loss function*. We investigate two different loss functions in this paper. $LogLoss$ is the negative log conditional likelihood of the training data:

$$LogLoss_T(\mathbf{w}) = \sum_{i=1}^{m} -\log \mathrm{P}(y_i^\star \mid x_i, \mathcal{Y}_i)$$

Optimising $LogLoss$ finds the $\widehat{\mathbf{w}}$ that define (regularised) conditional Maximum Entropy models.

It turns out that optimising $LogLoss$ yields suboptimal weight vectors $\widehat{\mathbf{w}}$ here. $LogLoss$ is a symmetric loss function (i.e., each mistake is equally weighted), while our f-score evaluation metric weights "edited" labels more highly, as explained in section 4. Because our data is so skewed (i.e., "edited" words are comparatively infrequent), we

[3] In the situation where the true "edited" labelling does not appear in the 25-best list $\mathcal{Y}_i$ produced by the noisy-channel model, we choose $y_i^\star$ to be a labelling in $\mathcal{Y}_i$ closest to the true labelling.

can improve performance by using an asymmetric loss function.

Inspired by our evaluation metric, we devised an *approximate expected f-score loss function* $FLoss$.

$$FLoss_T(\mathbf{w}) \;=\; 1 - \frac{2\mathrm{E}_{\mathbf{w}}[c]}{g + \mathrm{E}_{\mathbf{w}}[e]}$$

This approximation assumes that the expectations approximately distribute over the division: see Jansche (2005) and Smith and Eisner (2006) for other approximations to expected f-score and methods for optimising them. We experimented with other asymmetric loss functions (e.g., the expected error rate) and found that they gave very similar results.

An advantage of $FLoss$ is that it and its derivatives with respect to $\mathbf{w}$ (which are required for numerical optimisation) are easy to calculate exactly. For example, the expected number of correct "edited" words is:

$$\begin{aligned}\mathrm{E}_{\mathbf{w}}[c] &= \sum_{i=1}^{n} \mathrm{E}_{\mathbf{w}}[c_{y_i^\star} \mid \mathcal{Y}_i], \text{ where:} \\ \mathrm{E}_{\mathbf{w}}[c_{y_i^\star} \mid \mathcal{Y}_i] &= \sum_{y \in \mathcal{Y}_i} c_{y_i^\star}(y)\, \mathrm{P}_{\mathbf{w}}(y \mid x_i, \mathcal{Y}_i)\end{aligned}$$

and $c_{y^\star}(y)$ is the number of correct "edited" labels in y given the gold labelling $y^\star$. The derivatives of $FLoss$ are:

$$\frac{\partial FLoss_T}{\partial w_j}(\mathbf{w}) \;=\; \frac{1}{g + \mathrm{E}_{\mathbf{w}}[e]} \left(FLoss_T(\mathbf{w}) \frac{\partial \mathrm{E}_{\mathbf{w}}[e]}{\partial w_j} - 2 \frac{\partial \mathrm{E}_{\mathbf{w}}[c]}{\partial w_j} \right)$$

where:

$$\frac{\partial \mathrm{E}_{\mathbf{w}}[c]}{\partial w_j} \;=\; \sum_{i=1}^{n} \frac{\partial \mathrm{E}_{\mathbf{w}}[c_{y_i^\star} \mid x_i, \mathcal{Y}_i]}{\partial w_j}$$

$$\frac{\partial \mathrm{E}_{\mathbf{w}}[c_{y^\star} \mid x, \mathcal{Y}]}{\partial w_j} \;=\; \mathrm{E}_{\mathbf{w}}[f_j c_{y^\star} \mid x, \mathcal{Y}] - \mathrm{E}_{\mathbf{w}}[f_j \mid x, \mathcal{Y}]\, \mathrm{E}_{\mathbf{w}}[c_{y^\star} \mid x, \mathcal{Y}].$$

$\partial \mathrm{E}[e] / \partial w_j$ is given by a similar formula.

9 Results

We follow Charniak and Johnson (2001) and split the corpus into main training data, held-out training data and test data as follows: main training consisted of all sw[23]$*$.dps files, held-out training consisted of all sw4[5-9]$*$.dps files and test consisted of all sw4[0-1]$*$.dps files. However, we follow (Johnson and Charniak, 2004) in deleting all partial words and punctuation from the training and test data (they argued that this is more realistic in a speech processing application).

Table 1 shows the results for the different models on held-out data. To avoid over-fitting on the test data, we present the f-scores over held-out training data instead of test data. We used the held-out data to select the best-performing set of reranker features, which consisted of features for all of the language models plus the extended (i.e., indicator) features, and used this model to analyse the test data. The f-score of this model on test data was 0.838. In this table, the set of *Extended Features* is defined as all the boolean features as described in Section 7.

We first observe that adding different external language models does increase the final score. The difference between the external language models is relatively small, although the differences in choice are several orders of magnitude. Despite the putative noise in the corpus, a language model built on Google's Web1T data seems to perform very well. Only the model where Switchboard 4-grams are used scores slightly lower, we explain this because the internal bigram model of the noisy channel model is already trained on Switchboard and so this model adds less new information to the reranker than the other models do.

Including additional features to describe the problem space is very productive. Indeed the best performing model is the model which has all extended features and all language model features. The differences among the different language models when extended features are present are relatively small. We assume that much of the information expressed in the language models overlaps with the lexical features.

We find that using a loss function related to our evaluation metric, rather than optimising $LogLoss$, consistently improves edit-word f-score. The standard $LogLoss$ function, which estimates the "maximum entropy" model, consistently performs worse than the loss function minimising expected errors.

The best performing model (Base + Ext. Feat. + All LM, using expected f-score loss) scores an f-score of **0.838** on **test data**. The results as indicated by the f-score outperform state-of-the-art models re-

Model	F-score	
Base (noisy channel, no reranking)	0.756	
Model	**log loss**	**expected f-score loss**
Base + Switchboard	0.776	0.791
Base + Fisher	0.771	0.797
Base + Gigaword	0.777	0.797
Base + Web1T	0.781	0.798
Base + Ext. Feat.	0.824	0.827
Base + Ext. Feat. + Switchboard	0.827	0.828
Base + Ext. Feat. + Fisher	0.841	0.856
Base + Ext. Feat. + Gigaword	0.843	0.852
Base + Ext. Feat. + Web1T	0.843	0.850
Base + Ext. Feat. + All LM	0.841	**0.857**

Table 1: Edited word detection f-score on held-out data for a variety of language models and loss functions

ported in literature operating on identical data, even though we use vastly less features than other do.

10 Conclusion and Future work

We have described a disfluency detection algorithm which we believe improves upon current state-of-the-art competitors. This model is based on a noisy channel model which scores putative analyses with a language model; its channel model is inspired by the observation that reparandum and repair are often very similar. As Johnson and Charniak (2004) noted, although this model performs well, a log-linear reranker can be used to increase performance.

We built language models from a variety of speech and non-speech corpora, and examine the effect they have on disfluency detection. We use language models derived from different larger corpora effectively in a maximum reranker setting. We show that the actual choice for a language model seems to be less relevant and newswire text can be used equally well for modelling fluent speech.

We describe different features to improve disfluency detection even further. Especially these features seem to boost performance significantly.

Finally we investigate the effect of different loss functions. We observe that using a loss function directly optimising our interest yields a performance increase which is at least at large as the effect of using very large language models.

We obtained an f-score which outperforms other models reported in literature operating on identical data, even though we use vastly fewer features than others do.

Acknowledgements

This work was supported was supported under Australian Research Council's Discovery Projects funding scheme (project number DP110102593) and by the Australian Research Council as part of the Thinking Head Project the Thinking Head Project, ARC/NHMRC Special Research Initiative Grant # TS0669874. We thank the anonymous reviewers for their helpful comments.

References

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram Version 1. Published by *Linguistic Data Consortium*, Philadelphia.

Erik Brill and Michele Banko. 2001. Mitigating the Paucity-of-Data Problem: Exploring the Effect of Training Corpus Size on Classifier Performance for Natural Language Processing. In *Proceedings of the First International Conference on Human Language Technology Research*.

Eugene Charniak and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 118–126.

Christopher Cieri David, David Miller, and Kevin Walker. 2004. Fisher English Training Speech Part 1 Transcripts. Published by *Linguistic Data Consortium*, Philadelphia.

Christopher Cieri David, David Miller, and Kevin Walker. 2005. Fisher English Training Speech Part 2 Transcripts. Published by *Linguistic Data Consortium*, Philadelphia.

John J. Godfrey and Edward Holliman. 1997. Switchboard-1 Release 2. Published by *Linguistic Data Consortium, Philadelphia*.

David Graff and Christopher Cieri. 2003. English gigaword. Published by *Linguistic Data Consortium*, Philadelphia.

Martin Jansche. 2005. Maximum Expected F-Measure Training of Logistic Regression Models. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 692–699, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Mark Johnson and Eugene Charniak. 2004. A TAG-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 33–39.

Mark Johnson, Eugene Charniak, and Matthew Lease. 2004. An Improved Model for Recognizing Disfluencies in Conversational Speech. In *Proceedings of the Rich Transcription Fall Workshop*.

Jeremy G. Kahn, Matthew Lease, Eugene Charniak, Mark Johnson, and Mari Ostendorf. 2005. Effective Use of Prosody in Parsing Conversational Speech. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 233–240, Vancouver, British Columbia, Canada.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 181–184.

Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. Treebank-3. Published by *Linguistic Data Consortium, Philadelphia*.

William Schuler, Samir AbdelRahman, Tim Miller, and Lane Schwartz. 2010. Broad-Coverage Parsing using Human-Like Memory Constraints. *Computational Linguistics*, 36(1):1–30.

Richard Schwartz, Long Nguyen, Francis Kubala, George Chou, George Zavaliagkos, and John Makhoul. 1994. On Using Written Language Training Data for Spoken Language Modeling. In *Proceedings of the Human Language Technology Workshop*, pages 94–98.

Elizabeth Shriberg and Andreas Stolcke. 1998. How far do speakers back up in repairs? A quantitative model. In *Proceedings of the International Conference on Spoken Language Processing*, pages 2183–2186.

Elizabeth Shriberg. 1994. *Preliminaries to a Theory of Speech Disuencies*. Ph.D. thesis, University of California, Berkeley.

David A. Smith and Jason Eisner. 2006. Minimum Risk Annealing for Training Log-Linear Models. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 787–794.

Matthew Snover, Bonnie Dorr, and Richard Schwartz. 2004. A Lexically-Driven Algorithm for Disfluency Detection. In *Proceedings of Human Language Technologies and North American Association for Computational Linguistics*, pages 157–160.

Andreas Stolcke. 2002. SRILM - An Extensible Language Modeling Toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 901–904.

Qi Zhang, Fuliang Weng, and Zhe Feng. 2006. A progressive feature selection algorithm for ultra large feature spaces. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 561–568.