A Progressive Feature Selection Algorithm for Ultra Large Feature Spaces

Qi Zhang
Computer Science Department
Fudan University
Shanghai 200433, P.R. China
qi_zhang@fudan.edu.cn

Fuliang Weng
Research and Technology Center
Robert Bosch Corp.
Palo Alto, CA 94304, USA
fuliang.weng@rtc.bosch.com

Zhe Feng
Research and Technology Center
Robert Bosch Corp.
Palo Alto, CA 94304, USA
zhe.feng@rtc.bosch.com

Abstract

Recent developments in statistical modeling of various linguistic phenomena have shown that additional features give consistent performance improvements. Quite often, improvements are limited by the number of features a system is able to explore. This paper describes a novel progressive training algorithm that selects features from virtually unlimited feature spaces for conditional maximum entropy (CME) modeling. Experimental results in edit region identification demonstrate the benefits of the progressive feature selection (PFS) algorithm: the PFS algorithm maintains the same accuracy performance as previous CME feature selection algorithms (e.g., Zhou et al., 2003) when the same feature spaces are used. When additional features and their combinations are used, the PFS gives 17.66% relative improvement over the previously reported best result in edit region identification on Switchboard corpus (Kahn et al., 2005), which leads to a 20% relative error reduction in parsing the Switchboard corpus when gold edits are used as the upper bound.

1 Introduction

Conditional Maximum Entropy (CME) modeling has received a great amount of attention within natural language processing community for the past decade (e.g., Berger et al., 1996; Reynar and Ratnaparkhi, 1997; Koeling, 2000; Malouf, 2002; Zhou et al., 2003; Riezler and Vasserman, 2004). One of the main advantages of CME modeling is the ability to incorporate a variety of features in a uniform framework with a sound mathematical foundation. Recent improvements on the original incremental feature selection (IFS) algorithm, such as Malouf (2002) and Zhou et al. (2003), greatly speed up the feature selection process. However, like many other statistical modeling algorithms, such as boosting (Schapire and Singer, 1999) and support vector machine (Vapnik 1995), the algorithm is limited by the size of the defined feature space. Past results show that larger feature spaces tend to give better results. However, finding a way to include an unlimited amount of features is still an open research problem.

In this paper, we propose a novel progressive feature selection (PFS) algorithm that addresses the feature space size limitation. The algorithm is implemented on top of the Selective Gain Computation (SGC) algorithm (Zhou et al., 2003), which offers fast training and high quality models. Theoretically, the new algorithm is able to explore an unlimited amount of features. Because of the improved capability of the CME algorithm, we are able to consider many new features and feature combinations during model construction.

To demonstrate the effectiveness of our new algorithm, we conducted a number of experiments on the task of identifying edit regions, a practical task in spoken language processing. Based on the convention from Shriberg (1994) and Charniak and Johnson (2001), a disfluent spoken utterance is divided into three parts: the *reparandum*, the part that is repaired; the *inter-*

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 561–568,
Sydney, July 2006.

regnum, which can be filler words or empty; and the *repair/repeat*, the part that replaces or repeats the reparandum. The first two parts combined are called an *edit* or *edit region*. An example is shown below:

It is,	you know,	this is a tough problem.
reparandum	interregnum	repair

In section 2, we briefly review the CME modeling and SGC algorithm. Then, section 3 gives a detailed description of the PFS algorithm. In section 4, we describe the Switchboard corpus, features used in the experiments, and the effectiveness of the PFS with different feature spaces. Section 5 concludes the paper.

2 Background

Before presenting the PFS algorithm, we first give a brief review of the conditional maximum entropy modeling, its training process, and the SGC algorithm. This is to provide the background and motivation for our PFS algorithm.

2.1 Conditional Maximum Entropy Model

The goal of CME is to find the most uniform conditional distribution of y given observation x, $p(y|x)$, subject to constraints specified by a set of features $f_i(x,y)$, where features typically take the value of either 0 or 1 (Berger et al., 1996). More precisely, we want to maximize

$$H(p) = -\sum_{x,y} \tilde{p}(x)p(y|x)\log(p(y|x)) \qquad (1)$$

given the constraints:

$$E(f_i) = \tilde{E}(f_i) \qquad (2)$$

where

$$\tilde{E}(f_i) = \sum_{x,y} \tilde{p}(x,y)f_i(x,y)$$

is the empirical expected feature count from the training data and

$$E(f_i) = \sum_{x,y} \tilde{p}(x)p(y|x)f_i(x,y)$$

is the feature expectation from the conditional model $p(y|x)$.

This results in the following exponential model:

$$p(y|x) = \frac{1}{Z(x)}\exp\left(\sum_j \lambda_j f_j(x,y)\right) \qquad (3)$$

where λ_j is the weight corresponding to the feature f_j, and $Z(x)$ is a normalization factor.

A variety of different phenomena, including lexical, structural, and semantic aspects, in natural language processing tasks can be expressed in terms of features. For example, a feature can be whether the word in the current position is a verb, or the word is a particular lexical item. A feature can also be about a particular syntactic subtree, or a dependency relation (e.g., Charniak and Johnson, 2005).

2.2 Selective Gain Computation Algorithm

In real world applications, the number of possible features can be in the millions or beyond. Including all the features in a model may lead to data over-fitting, as well as poor efficiency and memory overflow. Good feature selection algorithms are required to produce efficient and high quality models. This leads to a good amount of work in this area (Ratnaparkhi et al., 1994; Berger et al., 1996; Pietra et al, 1997; Zhou et al., 2003; Riezler and Vasserman, 2004)

In the most basic approach, such as Ratnaparkhi et al. (1994) and Berger et al. (1996), training starts with a uniform distribution over all values of y and an empty feature set. For each candidate feature in a predefined feature space, it computes the likelihood gain achieved by including the feature in the model. The feature that maximizes the gain is selected and added to the current model. This process is repeated until the gain from the best candidate feature only gives marginal improvement. The process is very slow, because it has to re-compute the gain for every feature at each selection stage, and the computation of a parameter using Newton's method becomes expensive, considering that it has to be repeated many times.

The idea behind the SGC algorithm (Zhou et al., 2003) is to use the gains computed in the previous step as approximate upper bounds for the subsequent steps. The gain for a feature needs to be re-computed only when the feature reaches the top of a priority queue ordered by gain. In other words, this happens when the feature is the top candidate for inclusion in the model. If the re-computed gain is smaller than that of the next candidate in the list, the feature is re-ranked according to its newly computed gain, and the feature now at the top of the list goes through the same gain re-computing process.

This heuristics comes from evidences that the gains become smaller and smaller as more and more good features are added to the model. This can be explained as follows: assume that the Maximum Likelihood (ML) estimation lead to the best model that reaches a ML value. The ML value is the upper bound. Since the gains need to be positive to proceed the process, the difference

between the Likelihood of the current and the ML value becomes smaller and smaller. In other words, the possible gain each feature may add to the model gets smaller. Experiments in Zhou et al. (2003) also confirm the prediction that the gains become smaller when more and more features are added to the model, and the gains do not get unexpectively bigger or smaller as the model grows. Furthermore, the experiments in Zhou et al. (2003) show no significant advantage for looking ahead beyond the first element in the feature list. The SGC algorithm runs hundreds to thousands of times faster than the original IFS algorithm without degrading classification performance. We used this algorithm for it enables us to find high quality CME models quickly.

The original SGC algorithm uses a technique proposed by Darroch and Ratcliff (1972) and elaborated by Goodman (2002): when considering a feature f_i, the algorithm only modifies those un-normalized conditional probabilities:

$$\exp\left(\sum_j \lambda_j f_j(x, y)\right)$$

for (x, y) that satisfy $f_i(x, y)=1$, and subsequently adjusts the corresponding normalizing factors $Z(x)$ in (3). An implementation often uses a mapping table, which maps features to the training instance pairs (x, y).

3 Progressive Feature Selection Algorithm

In general, the more contextual information is used, the better a system performs. However, richer context can lead to combinatorial explosion of the feature space. When the feature space is huge (e.g., in the order of tens of millions of features or even more), the SGC algorithm exceeds the memory limitation on commonly available computing platforms with gigabytes of memory.

To address the limitation of the SGC algorithm, we propose a progressive feature selection algorithm that selects features in multiple rounds. The main idea of the PFS algorithm is to split the feature space into tractable disjoint sub-spaces such that the SGC algorithm can be performed on each one of them. In the merge step, the features that SGC selects from different sub-spaces are merged into groups. Instead of re-generating the feature-to-instance mapping table for each sub-space during the time of splitting and merging, we create the new mapping table from the previous round's tables by collecting those entries that correspond to the selected features. Then, the SGC algorithm is performed on each of the feature groups and new features are selected from each of them. In other words, the feature space splitting and subspace merging are performed mainly on the feature-to-instance mapping tables. This is a key step that leads to this very efficient PFS algorithm.

At the beginning of each round for feature selection, a uniform prior distribution is always assumed for the new CME model. A more precise description of the PFS algorithm is given in Table 1, and it is also graphically illustrated in Figure 1.

Given:
Feature space $\mathbf{F^{(0)}} = \{f_1^{(0)}, f_2^{(0)}, \ldots, f_N^{(0)}\}$,
step_num = m, select_factor = s

1. Split the feature space into N_1 parts
$\{\mathbf{F_1^{(1)}}, \mathbf{F_2^{(1)}}, \ldots, \mathbf{F_{N1}^{(1)}}\}$ = split($\mathbf{F^{(0)}}$)

2. for k=1 to m-1 **do**
//2.1 Feature selection
for each feature space $\mathbf{F_i^{(k)}}$ **do**
$\mathbf{FS_i^{(k)}}$ = SGC($\mathbf{F_i^{(k)}}$, s)
//2.2 Combine selected features
$\{\mathbf{F_1^{(k+1)}}, \ldots, \mathbf{F_{Nk+1}^{(k+1)}}\}$ =
merge($\mathbf{FS_1^{(k)}}, \ldots, \mathbf{FS_{Nk}^{(k)}}$)

3. Final feature selection & optimization
$\mathbf{F^{(m)}}$ = merge($\mathbf{FS_1^{(m-1)}}, \ldots, \mathbf{FS_{Nm-1}^{(m-1)}}$)
$\mathbf{FS^{(m)}}$ = SGC($\mathbf{F^{(m)}}$, s)
$\mathbf{M_{final}}$ = Opt($\mathbf{FS^{(m)}}$)

Table 1. The PFS algorithm.

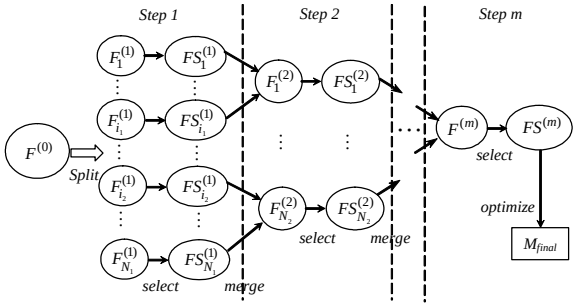

Figure 1. Graphic illustration of PFS algorithm.

In Table 1, SGC() invokes the SGC algorithm, and Opt() optimizes feature weights. The functions split() and merge() are used to split and merge the feature space respectively.

Two variations of the split() function are investigated in the paper and they are described below:

1. **random-split**: randomly split a feature space into n- disjoint subspaces, and select an equal amount of features for each feature subspace.
2. **dimension-based-split**: split a feature space into disjoint subspaces based on fea-

ture dimensions/variables, and select the number of features for each feature subspace with a certain distribution.

We use a simple method for merge() in the experiments reported here, i.e., adding together the features from a set of selected feature subspaces.

One may image other variations of the split() function, such as allowing overlapping subspaces. Other alternatives for merge() are also possible, such as randomly grouping the selected feature subspaces in the dimension-based split. Due to the limitation of the space, they are not discussed here.

This approach can in principle be applied to other machine learning algorithms as well.

4 Experiments with PFS for Edit Region Identification

In this section, we will demonstrate the benefits of the PFS algorithm for identifying edit regions. The main reason that we use this task is that the edit region detection task uses features from several levels, including prosodic, lexical, and syntactic ones. It presents a big challenge to find a set of good features from a huge feature space.

First we will present the additional features that the PFS algorithm allows us to include. Then, we will briefly introduce the variant of the Switchboard corpus used in the experiments. Finally, we will compare results from two variants of the PFS algorithm.

4.1 Edit Region Identification Task

In spoken utterances, disfluencies, such as self-editing, pauses and repairs, are common phenomena. Charniak and Johnson (2001) and Kahn et al. (2005) have shown that improved edit region identification leads to better parsing accuracy – they observe a relative reduction in parsing f-score error of 14% (2% absolute) between automatic and oracle edit removal.

The focus of our work is to show that our new PFS algorithm enables the exploration of much larger feature spaces for edit identification – including prosodic features, their confidence scores, and various feature combinations – and consequently, it further improves edit region identification. Memory limitation prevents us from including all of these features in experiments using the boosting method described in Johnson and Charniak (2004) and Zhang and Weng (2005). We couldn't use the new features with the SGC algorithm either for the same reason.

The features used here are grouped according to variables, which define feature sub-spaces as in Charniak and Johnson (2001) and Zhang and Weng (2005). In this work, we use a total of 62 variables, which include 16 [1] variables from Charniak and Johnson (2001) and Johnson and Charniak (2004), an additional 29 variables from Zhang and Weng (2005), 11 hierarchical POS tag variables, and 8 prosody variables (labels and their confidence scores). Furthermore, we explore 377 combinations of these 62 variables, which include 40 combinations from Zhang and Weng (2005). The complete list of the variables is given in Table 2, and the combinations used in the experiments are given in Table 3. One additional note is that some features are obtained after the rough copy procedure is performed, where we used the same procedure as the one by Zhang and Weng (2005). For a fair comparison with the work by Kahn et al. (2005), word fragment information is retained.

4.2 The Re-segmented Switchboard Data

In order to include prosodic features and be able to compare with the state-oft-art, we use the University of Washington re-segmented Switchboard corpus, described in Kahn et al. (2005). In this corpus, the Switchboard sentences were segmented into V5-style sentence-like units (SUs) (LDC, 2004). The resulting sentences fit more closely with the boundaries that can be detected through automatic procedures (e.g., Liu et al., 2005). Because the edit region identification results on the original Switchboard are not directly comparable with the results on the newly segmented data, the state-of-art results reported by Charniak and Johnson (2001) and Johnson and Charniak (2004) are repeated on this new corpus by Kahn et al. (2005).

The re-segmented UW Switchboard corpus is labeled with a simplified subset of the ToBI prosodic system (Ostendorf et al., 2001). The three simplified labels in the subset are *p*, 1 and 4, where *p* refers to a general class of disfluent boundaries (e.g., word fragments, abruptly shortened words, and hesitation); 4 refers to break level 4, which describes a boundary that has a boundary tone and phrase-final lengthening;

[1] Among the original 18 variables, two variables, P_f and T_f are not used in our experiments, because they are mostly covered by the other variables. Partial word flags only contribute to 3 features in the final selected feature list.

Categories		Variable Name	Short Description
Words	Orthographic Words	W_{-5}, … , W_{+5}	Words at the current position and the left and right 5 positions.
	Partial Word Flags	P_{-3}, …, P_{+3}	Partial word flags at the current position and the left and right 3 positions
	Distance	$D_{INTJ,}$ $D_{W,}$ D_{Bigram}, $D_{Trigram}$	Distance features
Tags	POS Tags	T_{-5}, …, T_{+5}	POS tags at the current position and the left and right 5 positions.
	Hierarchical POS Tags (HTag)	HT_{-5}, …, HT_{+5}	Hierarchical POS tags at the current position and the left and right 5 positions.
Rough Copy	HTag Rough Copy	N_m, N_n, N_i, N_l, N_r, T_i	Hierarchical POS rough copy features.
	Word Rough Copy	WN_m, WN_i, WN_l, WN_r	Word rough copy features.
Prosody	Prosody Labels	PL_0, …, PL_3	Prosody label with largest post possibility at the current position and the right 3 positions.
	Prosody Scores	PC_0, …, PC_3	Prosody confidence at the current position and the right 3 positions.

Table 2. A complete list of variables used in the experiments.

Categories			Short Description	Number of Combinations
Tags	HTagComb		Combinations among *Hierarchical POS Tags*	55
Words	WTComb	OrthWordComb	Combinations among *Orthographic Words*	55
Tags		WTTComb	Combinations of *Orthographic Words* and *POS Tags*; Combination among *POS Tags*	176
Rough Copy	RCComb		Combinations of *HTag Rough Copy* and *Word Rough Copy*	55
Prosody	PComb		Combinations among *Prosody*, and with *Words*	36

Table 3. All the variable combinations used in the experiments.

and 1 is used to include the break index levels BL 0, 1, 2, and 3. Since the majority of the corpus is labeled via automatic methods, the f-scores for the prosodic labels are not high. In particular, 4 and *p* have f-scores of about 70% and 60% respectively (Wong et al., 2005). Therefore, in our experiments, we also take prosody confidence scores into consideration.

Besides the symbolic prosody labels, the corpus preserves the majority of the previously annotated syntactic information as well as edit region labels.

In following experiments, to make the results comparable, the same data subsets described in Kahn et al. (2005) are used for training, developing and testing.

4.3 Experiments

The best result on the UW Switchboard for edit region identification uses a TAG-based approach (Kahn et al., 2005). On the original Switchboard corpus, Zhang and Weng (2005) reported nearly 20% better results using the boosting method with a much larger feature space[2]. To allow comparison with the best past results, we create a new CME baseline with the same set of features as that used in Zhang and Weng (2005).

We design a number of experiments to test the following hypotheses:

1. PFS can include a huge number of new features, which leads to an overall performance improvement.
2. Richer context, represented by the combinations of different variables, has a positive impact on performance.
3. When the same feature space is used, PFS performs equally well as the original SGC algorithm.

The new models from the PFS algorithm are trained on the training data and tuned on the development data. The results of our experiments on the test data are summarized in Table 4. The first three lines show that the TAG-based approach is outperformed by the new CME baseline (line 3) using all the features in Zhang and Weng (2005). However, the improvement from

[2] PFS is not applied to the boosting algorithm at this time because it would require significant changes to the available algorithm.

Feature Space Codes	number of features	Results on test data		
		Precision	Recall	F-Value
TAG-based result on UW-SWBD reported in Kahn et al. (2005)				**78.20**
CME with all the variables from Zhang and Weng (2005)	2412382	89.42	71.22	79.29
CME with all the variables from Zhang and Weng (2005) + post	2412382	87.15	73.78	**79.91**
+HTag +HTagComb +WTComb +RCComb	17116957	90.44	72.53	80.50
+HTag +HTagComb +WTComb +RCComb +PL_0 … PL_3	17116981	88.69	74.01	80.69
+HTag +HTagComb +WTComb +RCComb +PComb: without cut	20445375	89.43	73.78	80.86
+HTag +HTagComb +WTComb +RCComb +PComb: cut2	19294583	88.95	74.66	**81.18**
+HTag +HTagComb +WTComb +RCComb +PComb: cut2 +Gau	19294583	90.37	74.40	81.61
+HTag +HTagComb +WTComb +RCComb +PComb: cut2 +post	19294583	86.88	77.29	81.80
+HTag +HTagComb +WTComb +RCComb +PComb: cut2 +Gau +post	19294583	87.79	77.02	**82.05**

Table 4. Summary of experimental results with PFS.

CME is significantly smaller than the reported results using the boosting method. In other words, using CME instead of boosting incurs a performance hit.

The next four lines in Table 4 show that additional combinations of the feature variables used in Zhang and Weng (2005) give an absolute improvement of more than 1%. This improvement is realized through increasing the search space to more than 20 million features, 8 times the maximum size that the original boosting and CME algorithms are able to handle.

Table 4 shows that prosody labels alone make no difference in performance. Instead, for each position in the sentence, we compute the entropy of the distribution of the labels' confidence scores. We normalize the entropy to the range [0, 1], according to the formula below:

$$score = 1 - H(p)/H(Uniform) \qquad (4)$$

Including this feature does result in a good improvement. In the table, *cut2* means that we equally divide the feature scores into 10 buckets and any number below 0.2 is ignored. The total contribution from the combined feature variables leads to a 1.9% absolute improvement. This confirms the first two hypotheses.

When Gaussian smoothing (Chen and Rosenfeld, 1999), labeled as *+Gau*, and post-processing (Zhang and Weng, 2005), labeled as *+post*, are added, we observe 17.66% relative improvement (or 3.85% absolute) over the previous best f-score of 78.2 from Kahn et al. (2005).

To test hypothesis 3, we are constrained to the feature spaces that both PFS and SGC algorithms can process. Therefore, we take all the variables from Zhang and Weng (2005) as the feature space for the experiments. The results are listed in Table 5. We observed no f-score degradation with PFS. Surprisingly, the total amount of time PFS spends on selecting its best features is smaller than the time SGC uses in selecting its best features. This confirms our hypothesis 3.

Split / Non-split	Results on test data		
	Precision	Recall	F-Value
non-split	89.42	71.22	79.29
split by 4 parts	89.67	71.68	79.67
split by 10 parts	89.65	71.29	79.42

Table 5. Comparison between PFS and SGC with all the variables from Zhang and Weng (2005).

The last set of experiments for edit identification is designed to find out what split strategies PFS algorithm should adopt in order to obtain good results. Two different split strategies are tested here. In all the experiments reported so far, we use 10 random splits, i.e., all the features are randomly assigned to 10 subsets of equal size. We may also envision a split strategy that divides the features based on feature variables (or *dimensions*), such as word-based, tag-based, etc. The four dimensions used in the experiments are listed as the top categories in Tables 2 and 3, and the results are given in Table 6.

Split Criteria	Allocation Criteria	Results on test data		
		Precision	Recall	F-Value
Random	Uniform	88.95	74.66	81.18
Dimension	Uniform	89.78	73.42	80.78
Dimension	Prior	89.78	74.01	81.14

Table 6. Comparison of split strategies using feature space +HTag+HTagComb+WTComb+RCComb+PComb: cut2

In Table 6, the first two columns show criteria for splitting feature spaces and the number of features to be allocated for each group. *Random* and *Dimension* mean random-split and dimension-based-split, respectively. When the criterion

is *Random*, the features are allocated to different groups randomly, and each group gets the same number of features. In the case of dimension-based split, we determine the number of features allocated for each dimension in two ways. When the split is *Uniform*, the same number of features is allocated for each dimension. When the split is *Prior*, the number of features to be allocated in each dimension is determined in proportion to the importance of each dimension. To determine the importance, we use the distribution of the selected features from each dimension in the model "+ HTag + HTagComb + WTComb + RCComb + PComb: cut2", namely: Word-based 15%, Tag-based 70%, RoughCopy-based 7.5% and Prosody-based 7.5%[3]. From the results, we can see no significant difference between the random-split and the dimension-based-split.

To see whether the improvements are translated into parsing results, we have conducted one more set of experiments on the UW Switchboard corpus. We apply the latest version of Charniak's parser (2005-08-16) and the same procedure as Charniak and Johnson (2001) and Kahn et al. (2005) to the output from our best edit detector in this paper. To make it more comparable with the results in Kahn et al. (2005), we repeat the same experiment with the gold edits, using the latest parser. Both results are listed in Table 7. The difference between our best detector and the gold edits in parsing (1.51%) is smaller than the difference between the TAG-based detector and the gold edits (1.9%). In other words, if we use the gold edits as the upper bound, we see a relative error reduction of 20.5%.

Methods	Edit *F-score*	Parsing *F-score*		
		Reported in Kahn et al. (2005)	Latest Charniak Parser	Diff. with Oracle
Oracle	100	86.9	87.92	--
Kahn et al. (2005)	78.2	85.0	--	**1.90**
PFS best results	82.05	--	86.41	**1.51**

Table 7. Parsing F-score various different edit region identification results.

[3] It is a bit of cheating to use the distribution from the selected model. However, even with this distribution, we do not see any improvement over the version with random-split.

5 Conclusion

This paper presents our progressive feature selection algorithm that greatly extends the feature space for conditional maximum entropy modeling. The new algorithm is able to select features from feature space in the order of tens of millions in practice, i.e., 8 times the maximal size previous algorithms are able to process, and unlimited space size in theory. Experiments on edit region identification task have shown that the increased feature space leads to 17.66% relative improvement (or 3.85% absolute) over the best result reported by Kahn et al. (2005), and 10.65% relative improvement (or 2.14% absolute) over the new baseline SGC algorithm with all the variables from Zhang and Weng (2005). We also show that symbolic prosody labels together with confidence scores are useful in edit region identification task.

In addition, the improvements in the edit identification lead to a relative 20% error reduction in parsing disfluent sentences when gold edits are used as the upper bound.

Acknowledgement

This work is partly sponsored by a NIST ATP funding. The authors would like to express their many thanks to Mari Ostendorf and Jeremy Kahn for providing us with the re-segmented UW Switchboard Treebank and the corresponding prosodic labels. Our thanks also go to Jeff Russell for his careful proof reading, and the anonymous reviewers for their useful comments. All the remaining errors are ours.

References

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22 (1): 39-71.

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*, 118-126, Pittsburgh, PA, USA.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of the 43rd Annual Meeting of Association for Computational Linguistics*, 173-180, Ann Arbor, MI, USA.

Stanley Chen and Ronald Rosenfeld. 1999. A Gaussian Prior for Smoothing Maximum Entropy Mod-

els. *Technical Report CMUCS-99-108*, Carnegie Mellon University.

John N. Darroch and D. Ratcliff. 1972. Generalized Iterative Scaling for Log-Linear Models. In *Annals of Mathematical Statistics*, 43(5): 1470-1480.

Stephen A. Della Pietra, Vincent J. Della Pietra, and John Lafferty. 1997. Inducing Features of Random Fields. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4): 380-393.

Joshua Goodman. 2002. Sequential Conditional Generalized Iterative Scaling. In *Proceedings of the 40th Annual Meeting of Association for Computational Linguistics*, 9-16, Philadelphia, PA, USA.

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*, 33-39, Barcelona, Spain.

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

Rob Koeling. 2000. Chunking with Maximum Entropy Models. In *Proceedings of the CoNLL-2000 and LLL-2000*, 139-141, Lisbon, Portugal.

LDC. 2004. Simple MetaData Annotation Specification. *Technical Report of Linguistic Data Consortium*. (http://www.ldc.upenn.edu/Projects/MDE).

Yang Liu, Elizabeth Shriberg, Andreas Stolcke, Barbara Peskin, Jeremy Ang, Dustin Hillard, Mari Ostendorf, Marcus Tomalin, Phil Woodland and Mary Harper. 2005. Structural Metadata Research in the EARS Program. In *Proceedings of the 30th ICASSP*, volume V, 957-960, Philadelphia, PA, USA.

Robert Malouf. 2002. A Comparison of Algorithms for Maximum Entropy Parameter Estimation. In *Proceedings of the 6th Conference on Natural Language Learning (CoNLL-2002)*, 49-55, Taibei, Taiwan.

Mari Ostendorf, Izhak Shafran, Stefanie Shattuck-Hufnagel, Leslie Charmichael, and William Byrne. 2001. A Prosodically Labeled Database of Spontaneous Speech. In *Proceedings of the ISCA Workshop of Prosody in Speech Recognition and Understanding*, 119-121, Red Bank, NJ, USA.

Adwait Ratnaparkhi, Jeff Reynar and Salim Roukos. 1994. A Maximum Entropy Model for Prepositional Phrase Attachment. In *Proceedings of the ARPA Workshop on Human Language Technology*, 250-255, Plainsboro, NJ, USA.

Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A Maximum Entropy Approach to Identifying Sentence Boundaries. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, 16-19, Washington D.C., USA.

Stefan Riezler and Alexander Vasserman. 2004. Incremental Feature Selection and L1 Regularization for Relaxed Maximum-entropy Modeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 174-181, Barcelona, Spain.

Robert E. Schapire and Yoram Singer, 1999. Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, 37(3): 297-336.

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

Vladimir Vapnik. 1995. T*he Nature of Statistical Learning Theory*. Springer, New York, NY, USA.

Darby Wong, Mari Ostendorf, Jeremy G. Kahn. 2005. Using Weakly Supervised Learning to Improve Prosody Labeling. *Technical Report UWEETR-2005-0003*, University of Washington.

Qi Zhang and Fuliang Weng. 2005. Exploring Features for Identifying Edited Regions in Disfluent Sentences. In *Proc. of the 9th International Workshop on Parsing Technologies*, 179-185, Vancouver, Canada.

Yaqian Zhou, Fuliang Weng, Lide Wu, and Hauke Schmidt. 2003. A Fast Algorithm for Feature Selection in Conditional Maximum Entropy Modeling. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, 153-159, Sapporo, Japan.