Reranking with Baseline System Scores and Ranks as Features

Kristy Hollingshead and **Brian Roark**
Center for Spoken Language Understanding
OGI School of Science & Engineering
Oregon Health & Science University
Beaverton, Oregon, 97006 USA
{hollingk,roark}@cslu.ogi.edu

Abstract

Reranking models have become ubiquitous in most NLP applications where suitable training data can be found, yet the utility of reranking has not been universal. This paper investigates two recent reranking "mysteries", and finds that the use of features derived from baseline system scores is responsible for the underperformance in both cases. Several alternative methods for encoding features derived from baseline scores are shown to repair these problems. Additional methods for deriving score features from the output of baseline systems are investigated, leading to a better understanding of the best practices for training reranking models under a variety of conditions.

1 Introduction

Reranking has proven to be a successful strategy for many NLP tasks, including parsing (Collins, 2000; Charniak and Johnson, 2005), speech recognition (Fiscus, 1997; Goel et al., 2000; Collins et al., 2005; Roark et al., 2007), and machine translation (Shen et al., 2004; Och and Ney, 2004). In a reranking scenario, a solution set produced by a baseline system is rescored, typically using richer, more complex models, in order to select a better solution from among those provided by the baseline system. The recent success of discriminative reranking approaches has made this technique a ubiquitous part of the standard toolbox in NLP. In tasks with appropriate training data, reranking can usually be counted on to provide a significant additional accuracy improvement.

Unfortunately, this is not always the case. Two recent syntactic parsing papers have included very promising techniques for use in a baseline system, which did not achieve typical improvements when a reranking model was trained to be applied to the new baseline systems output. In McClosky et al. (2006), the technique of "self-training" was used within the Charniak and Johnson (2005) parsing pipeline, whereby the output of the parser on large amounts of unlabeled data was used to re-train the baseline parsing model. This technique yielded over 1 percent absolute improvement over the already state-of-the-art parsing performance of the baseline Charniak and Johnson (2005) parsing pipeline, a very impressive result. However, this result was achieved without retraining the reranking model used in the pipeline, i.e., the reranker used in the final system configuration was trained on data that did not use self-training. This represents a significant mismatch between training and testing conditions, and perhaps it was just a lucky accident that the resulting model yielded system improvements when applied to these new n-best lists. In footnote 1 of McClosky et al. (2006), they note that they "attempted to retrain the reranker" but no improvement was achieved. This puzzling failure to train a useful reranking model for the new parsing conditions has remained a mystery.

In Hollingshead and Roark (2007), a similar lack of expected improvements with reranking was observed with the Charniak and Johnson (2005) parsing pipeline. In this paper, the parser was run twice, once with constraints imposed by a finite-state chunker, and once without constraints. The resulting n-best lists were unioned, and a reranker was trained on these unioned lists. Despite significant improvements in baseline parser performance, this particular scenario did not achieve the expected improve-

ments via reranking, much as with the McClosky et al. (2006) results. Again, this lack of improvement was noted as a mystery, requiring further investigation.

This paper has two objectives. First, we will demonstrate that the underperformance in both of these cases can be traced to the method of deriving a feature for reranking from the baseline system score of each candidate in the n-best list. Replicating the above training scenarios, we present several techniques that make better use of the baseline system score, and thus achieve the expected levels of reranking performance. Second, we will examine more generally the question of how to derive score features from the output of the baseline system. In particular, we investigate the use of features that go beyond the baseline system score to capture information about rank or centrality of the particular candidate within the set output by the baseline system. Overall, this paper advances our understanding of the best practices for training reranking models under varied baseline system conditions.

In the next section, we discuss a number of reasons why using the baseline system score directly as a feature in the model may be problematic. This will be followed by empirical results showing several techniques for making use of the baseline score applied in multiple reranking scenarios. Finally, we will propose a range of new score features derived from the baseline system output and compare them empirically, again using three different reranking scenarios.

2 Baseline System Score

Most reranking systems include the score from the baseline system within the reranking model. In parsing, many systems include the parser baseline score as a feature in the reranker, including the Collins (2000) parser and the Charniak and Johnson (2005) parsing pipeline. In fact, Charniak and Johnson (2005) explicitly demonstrated the difference in reranking performance with and without the parser baseline score as a feature: with the baseline score feature, reranking improved the parser F-score (on the Penn Wall St. Journal (WSJ) treebank (Marcus et al., 1993) section 24) from 88.9 to 90.2, whereas without the score feature the reranker provided a smaller F-score improvement, to 89.5. These results established what have become the standard practices of reranking.

However, there are several reasons why using the score from the baseline system directly as a feature may not be the best practice. First, in order to improve upon the baseline output, a reranking model must select something other than the baseline-best candidate solution at least once; otherwise the one-best output of the reranker would be unchanged from the baseline. Thus, even though the baseline scoring provides a fairly reliable estimate of high-quality candidate solutions, the reranking model must "down-weight" the baseline score in order to move away from the baseline-best candidate solution.

Second, the baseline scores are typically real-valued numbers, rather than integer-valued or indicator-valued numbers as is the case for most other features in a reranking model. It is common in machine learning to optimize continuous-valued features by quantization, i.e., grouping the feature values into a pre-defined number of *bins*, then treating each bin as an individual (indicator) feature, such as the approach to modeling exon length in Bernal et al. (2007). A secondary approach, and one that is taken by Charniak and Johnson (2005), is to essentially learn a *scaling factor*, where a single weight is learned for all values of the feature. One possible problem with learning such a scaling factor is that it represents a linear weighting function, which, in conjunction with the requirement that the reranking model down-weight the baseline score, can result in over-weighting candidates low in the n-best list.

Third, as previously stated, the baseline score is often fairly accurate in indicating high-quality solutions, and thus provides a particularly reliable feature. In fact, the baseline system score was shown to be the single most *influential* feature in the Charniak and Johnson (2005) parsing and reranking pipeline. Training with such a strong feature can result in weight under-training for the other features in the set (Sutton et al., 2006). One method for dealing with this is to remove the baseline-score feature from the reranking model during training, then empirically optimizing a mixing weight for combining the baseline score with the reranker score.

Finally, the baseline score may not be the most reliable metric derivable from the baseline system in terms of assessing the quality of a candidate. Furthermore, while such a score may reliably indicate a high-quality one-best candidate, it may not provide a reliable ranking of the remaining candidates in the list. Two methods for deriving more reliable features will be investigated in the last sections of the paper.

3 Pairwise Comparison Scores

In this paper, we conduct pairwise comparisons of the candidates in the list and use the values derived from the comparison function to re-score and re-order the list.

One reason for using characteristics of the list itself to score candidates in the list is that the relative ranking of candidates within the list may not be an accurate reflection of the similarity of the candidates. Recall that the baseline system is optimized to find the maximum likelihood candidate; similarly the reranking model optimizes with respect to the conditional likelihood of the data. There is no reward (or penalty) for determining an accurate ranking of the full list of candidates. Intuitively it seems reasonable to think that providing a more accurately-ranked list of candidates as input to the reranker would improve the accuracy of the model. However, it may be the case that the MaxEnt reranking model that we use in this paper is robust to changes in the relative ordering of the candidates. In this section we investigate the effects of re-ordering the parse candidates input to the reranking model.

3.1 Pairwise F-score

In parsing, parse candidates are evaluated according to the F_1 metric, which compares each parse to the gold-standard parse as follows:[1]

$$F_1 = \frac{2 * |A \cap B|}{|A| + |B|}. \quad (1)$$

A correct parse, i.e. one which is identical[2] to the gold-standard parse, has an F-score of 1.0. We can use the F-score to calculate the pairwise similarity of two parse candidates; two highly-similar parses will have a high pairwise F-score near 1.0, versus two very different parses which will have a much lower pairwise F-score.

3.2 Pairwise Distance-Score

F-score provides a *similarity* metric, indicating a level of shared information between two parse trees. It would be convenient if we could straightforwardly use $1-F$ to calculate the *distance* between two trees. Unfortunately, such a calculation is invalid as a distance metric because it does not meet the triangle inequality condition (Li et al., 2004), where the length of the path between two points must be less than or equal to the path created by adding an intermediary point:

$$\triangle AC \leq \triangle AB + \triangle BC.$$

To see that this condition is not met with the $1-F$ metric, consider the case where A=$\{x, y\}$, B=$\{x, y, z\}$, and C=$\{y, z\}$. Using $1-F$ to calculate the distance ($\triangle$) between the three sets gives us $\triangle AC$=$1-\frac{1}{4}$, $\triangle AB$=$1-\frac{4}{5}$, and $\triangle BC$=$1-\frac{4}{5}$. Since $\frac{3}{4}>\frac{1}{5}+\frac{1}{5}$, the "distance" calculated by $1-F$ violates the necessary condition of triangle inequality.

Fortunately, Yianilos (19912002) demonstrated that a metric meeting each of the conditions necessary for a true distance metric, including the triangle inequality condition, can be derived from the same calculations as used for similarity metrics such as F-score. (See that paper for a proof in the general case.) We will refer to this metric as the D-score, for distance metric, which is defined as follows:

$$D = 1 - \frac{|A \cap B|}{|A \cup B|}. \quad (2)$$

Note that since $|A \cup B| = |A| + |B| - |A \cap B|$, calculating this distance metric requires just the same measurements as calculating the F-score metric. Thus in addition to calculating pairwise F-scores to determine the average similarity amongst candidates in an n-best list, we will also calculate pairwise d-scores to determine the average distances between candidates in the list[3].

[1] $F_1 = \frac{2*R*P}{R+P}$, where $R = \frac{|A \cap B|}{|A|}$ and $P = \frac{|A \cap B|}{|B|}$.

[2] Standard PARSEEVAL parameterizations ignore the following differences for evaluation: different part-of-speech tags, ADVP/PRT substitutions, and disagreement on bracket boundaries that span punctuation marks.

[3] In the interest of time and space savings, we averaged F-scores across candidates rather than storing bracket counts as is the standard operating procedure.

Note the similarity between these methods of pairwise comparison and minimum Bayes-risk inference (Shafran and Byrne, 2004). The value resulting from the full pairwise comparison represents the expected loss of the hypothesized candidate with respect to the remainder of the n-best list.

4 Functions to Derive Features from Scores

In this section we will discuss several different functions for deriving and utilizing candidate scores as features in a reranking model. The default function, which we will term "Scaling Factor" simply uses the score in the model, scaled with whatever parameter the model training assigns to that feature. To contrast with this de-facto default, we present three others, one based on removing the score from the reranker training and empirically optimizing a mixing factor, and two based on quantizing the score.

4.1 Empirical Optimization

For empirical optimization, the score from the baseline model is not included in the reranking model. The resulting reranking model is then combined with the candidate score with weights empirically optimized on held aside data. This approach has multiple possible benefits, which we do not tease apart here. First, removing the baseline score from reranker training may improve performance due to weight under-training when the baseline score is left in the model. Second, any improvements could be due to optimizing the mixing with respect to parse accuracy achieved, rather than with respect to conditional likelihood, as the reranking training does. It is most likely that any improvements are due to a combination of these factors; future work will include further analysis.

4.2 Quantization

In order to address the problem that a linear function may not be well-suited for weighting a real-valued feature such as candidate scores, we will explore several options for quantizing the score. Quantization consists of grouping the possible values of a variable from a continuous set (such as real numbers) to a discrete set (such as integers). With a discrete set of values and by learning a separate weight for each possible values, in essence we can learn a non-linear function for the baseline system–score feature. We note that distributing the baseline score into bins is not a novel concept, and was in fact explored as an option in the original experimental setup of (Charniak and Johnson, 2005), where it was found to make little difference. We replicated these findings, but will demonstrate that under different conditions it does prove beneficial to bin the baseline-score feature.

4.2.1 Conditional Log-Probability Bins

The first method that we explore for quantizing the baseline-score is to simply define a set of bins[4] and assign each candidate to a bin according to the baseline-score for that candidate. In our case, this straightforward quantization is complicated by the fact that the log probability scores output by the Charniak baseline parser are not normalized, and thus are not comparable across different sentences. The probability scores are also affected by sentence length, such that the one-best candidates for short sentences have greater probability according to the parsing model than the one-best candidates for long sentences. Thus, we conditionally normalize the log probabilities output by the parser before quantizing into bins.

4.2.2 Rank Bins

Another method for quantizing the baseline-score is to use the rank of a candidate as its score. The benefit of using the rank as a feature is that rank is comparable across sentences, thus there is no need for normalization. For this paper, we treat the rank of a candidate as a binned feature, i.e. with a separate feature defined for each bin. There are several options for defining the boundaries of the bins for such a feature, including linearly dividing the ranks into equally-sized bins, and exponentially dividing the ranks such that the lower ranks are placed into larger bins and higher ranks are spread more sparsely across the bins. However, in this paper we simply investigate the straightforward method of defining a separate bin for each rank value.

[4]The pre-determined number of bins and the boundaries of each bin may affect performance, though such details are outside the focus of this paper. In the following experiments we arbitrarily selected the values of these parameters.

Score	Parser Output	Baseline	Scaling Factor	Empirical Optim	Binned CLP	Binned Rank
Baseline System Score	CJ05	88.9	90.2	90.2	90.4	90.2
	HR07	89.4	90.3	90.7	90.6	90.5
	MCJ06	90.2	90.9	91.0	**91.1**	90.8

Table 1: F-scores on WSJ section 24 of reranker-best parses where the weight for the baseline-score feature was either learned using the de-facto standard method (Scaling Factor), empirically optimized on held aside data (Empirical Optim), or quantized into bins by Conditional Log-Probability (CLP), or Rank.

We could also create a scalar feature to represent the rank values, i.e. by defining a single feature for the rank score, and multiplying the learned weight of the feature by the rank value of the candidate. We did explore this option briefly, but found similar problems to using the baseline-score as a scalar feature, namely that a linear function is inadequate for weighting such a feature.

5 Experimental Setup

Throughout this paper we will be using the well-known Charniak and Johnson (2005) parser and reranker, with the feature set described in that paper. We replicate three different experimental conditions. First, we replicate the experimental setup described in Charniak and Johnson (2005), following what has become the de-facto standard of producing 50-best parse candidates from the parser. Second, we replicate the self-training experiments in McClosky et al. (2006). We report on just the best-performing setup from that paper, namely retraining the parser on the reranker-best parses on the North American News Text Corpus (NANC), mixed with five copies of sections 02-21 of the WSJ Treebank (reported as "WSJx5+1,750k" in that paper). Finally, we reproduce one of the experimental conditions in Hollingshead and Roark (2007) by constraining the Charniak context-free parser with base phrases output by a finite-state parser (Hollingshead et al., 2005), then taking the union of the two n-best lists produced by the parser under the unconstrained and constrained conditions (reported as "Unconstrained∪FS-Constrained" in that paper). As mentioned in Section 1, these final two experimental conditions were chosen for this paper because under these conditions, using the de-facto standard reranking procedures, the output of the reranker underperforms expectations despite significant improvements in the baseline-parser candidates.

These three experimental conditions each represent a method of producing n-best parses, which we will refer to in experimental results as CJ05 (Charniak and Johnson, 2005), MCJ06 (McClosky et al., 2006), and HR07 (Hollingshead and Roark, 2007), respectively. We then train three MaxEnt reranker models, under a crossfold validation scenario, on the n-best lists produced under each of these conditions. Crossfold validation (20-fold with 2,000 sentences per fold) was used to train the reranking models. In all cases, WSJ section 00 was used as heldout data and section 24 as our development set. Unless stated otherwise, all reported results will be the F-score of WSJ section 24. Evaluation was performed using `evalb` under standard parameterizations. All statistical significance tests were conducted using the stratified shuffling test (Yeh, 2000).

6 Results: Baseline System Score

Table 1 presents the results of using our four functions to derive features for learning a reranking model from the baseline system score. The Baseline column shows the F-score accuracy of the baseline-best parse candidate, and the Scaling Factor column shows the F-score accuracy of the reranker-best parse candidate using the de-facto standard method of learning a feature weight for the baseline score within the reranking model.

Empirical optimization, where we withheld the baseline-score from the learned reranking model and empirically optimized the feature weight on our heldout set (WSJ 00) after training, provides the largest improvement on the HR07 parser output with an absolute increase of 0.4 F-score from the default method of learning a linearly-scaled weight for the baseline-score, which is a statistically significant improvement ($p < 0.05$). There is no change in the

Scoring Metric	Parser Output	Baseline	Scaling Factor	Empirical Optim	Binned CLP	Binned Rank
Pairwise F-score	CJ05	89.4	90.1	90.2	89.6	**90.6**
	HR07	89.4	90.1	90.7	89.8	90.3
	MCJ06	90.4	90.5	91.0	90.1	90.8
Pairwise D-score	CJ05	89.4	90.5	90.2	89.9	**90.6**
	HR07	89.5	90.4	**90.8**	89.9	90.4
	MCJ06	90.4	90.9	90.9	90.6	90.9

Table 2: F-scores on WSJ section 24 of reranker-best parses where candidates were re-scored and re-ordered by pairwise F-score or D-score comparison values. The feature weight for these scores was either learned using the default method (Scaling Factor), where the feature weight was empirically optimized on held aside data (Empirical Optim), or using one of two methods to quantize the score (Binned Conditional Log-Probability (CLP), and Binned Rank).

F-score for the CJ05 parser output, though at least the empirical optimization does not harm accuracy of the learned reranking model. The F-score on the MCJ06 parser output improves slightly from 90.9 for the default method to 91.0 under empirical optimization.

Our two methods for quantizing the baseline-score feature improve over the scaling factor method under some circumstances. By normalizing the log-probability scores and binning, the F-score of the reranker output improves for all three of our baseline parsers: CJ05 and MCJ06 improve very slightly, with a 0.2 increase in F-score above the default method; HR07 shows a slightly larger improvement with a 0.3 absolute increase in F-score. Using the rank of each candidate as a binned feature is less effective, providing no increase in F-score for CJ05, a smaller improvement of 0.2 F-score for HR07, and a marginal decrease in F-score for MCJ06, from 90.9 to 90.8.

These results show that the function chosen to derive a reranking feature from the baseline-score can noticeably impact reranker performance.

7 Results: Pairwise Comparison Scores

The experimental setup for these results is identical to the previous results, as described in Section 5. The pairwise comparison scores may exhibit similar behavior to the baseline system score in terms of down-weighting, quantizing, and under-training other features in the model, thus we again apply our four function for deriving features from the scores to these two pairwise scoring metrics. The score features are either treated as scalar features, empirically optimized on a heldout set, or binned by normalized-value or absolute rank.

Table 2 shows the results of using each of our two pairwise comparison metrics as a feature in a reranking model. One interesting result can be seen by comparing the Baseline scores here to the Baseline scores in Table 1, which we discuss further in Section 8

The different effects of the two pairwise metrics on the reranking model can be seen most clearly under the default method (Scaling Factor) of training a linear weight for the score feature. Using the pairwise F-score as the candidate score results in decreased accuracy across the board in comparison to learning a scaling factor on the baseline system score (Table 1): CJ05 drops by 0.1 F-score, HR07 by 0.2, and MCJ06 by 0.4. In contrast, using the pairwise D-score as the candidate score results in nearly-universally increased accuracy: CJ05 increases by 0.4 F-score and HR07 by 0.1, although MCJ06 remains the same as using the baseline system score.

Surprisingly, the method of normalizing the score then quantizing into bins is actually detrimental when using either of the pairwise comparison metrics in comparison to the baseline system score. Furthermore, the reranker-best F-score on the MCJ06 parser output is actually lower than the parser-best F-score under this condition. We suspect that the pairwise scores were not equally spread across the defined bins; candidates in an n-best list tend to be highly similar and thus will tend to score closer to one than zero, which was not taken into account

when we defined the bin boundaries.

Unsurprisingly, given the improved rank-order accuracy of the pairwise metrics, the reranking models trained with the binned rank of the candidate as a feature performed well. Both the CJ05 and MCJ06 parser outputs showed an increase in reranker-best F-score when using either of the two pairwise metrics to rank the candidates in comparison to using the candidate rank derived from the baseline system score. HR07 provides an small exception, showing a slight decrease in F-score when using the pairwise metrics rather than the baseline system score.

We also experimented with learning a reranking model with all three scoring metrics, along with the different methods for learning the feature-weights for the scores, but ultimately did not see a noticeable difference from using just one of the scores as a feature, perhaps indicating that the different scoring metrics do not provide complementary information in a reranking model.

Interestingly, there was no clear winning function and scoring-metric. Instead, the "best" choice of function and score differed according to the dataset being reranked. The bolded entries in Tables 1 and 2 indicate the highest-accuracy results found for each of the three datasets: CJ05 performed best using the Binned Rank function on either the pairwise-F score or the pairwise-D score; HR07 performed best with empirical optimization on the pairwise-D score; and MCJ06 performed best using the Binned Conditional Log-Probability on the baseline system score. Thus, our conclusion from these results is that, since the effectiveness of using any function to derive reranking features from any baseline score will vary across different datasets, best-practices for reranking might do well to consider a range of functions and scores rather than arbitrarily using the current de-facto standards.

8 Alternate Accuracy Evaluations

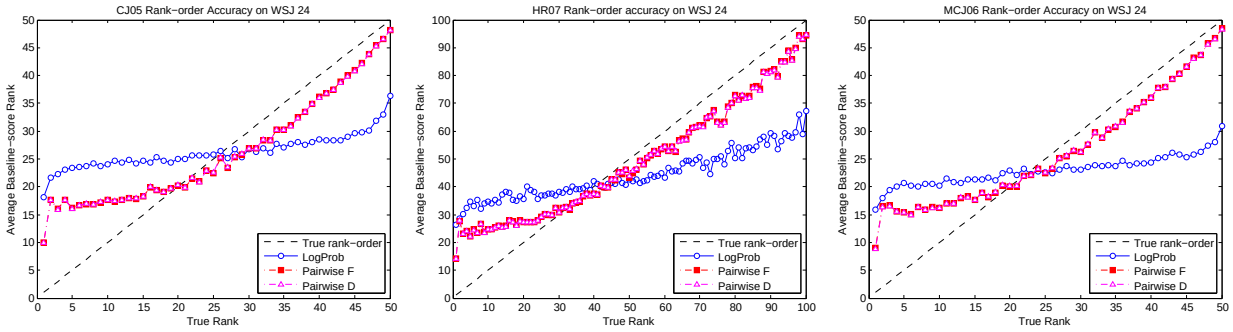
The empirical results presented in this paper have focused on improving, or better understanding, the performance of reranking models as influenced by features derived from some type of candidate score. In this section we take a brief side-excursion to look more directly at the accuracy of these candidate scores, i.e. ignoring, for the moment, their effectiveness as features in reranking.

Note that either of our pairwise comparison metrics may effect a total re-ordering of the list; the top-ranked candidate will be the one that is most similar to all other candidates in the list, which may or may not be the same as the top-ranked candidate according to the baseline-scoring. While candidate-order will not affect reranker performance, we can determine whether the one-best candidate according to each of our metrics is improved over the one-best candidate according to the baseline system score. Indeed, by comparing the second column (Baseline) of Tables 1 and 2, we can see that either of the pairwise metrics improves over the baseline system score. The improvements for HR07 and MCJ06 are small, from 0.1-0.2 F-score, but CJ05 is significantly improved by 0.5 absolute F-score.

8.1 Rank-Order

As previously mentioned, we wished to examine the effects of changing the relative ranking of candidates input to the reranking model. Figure 1 shows the accuracy of each of our three scoring metrics against the true rank of the candidate for each of our three parser outputs. In each graph in Figure 1, the circles represent the average candidate-rank as derived from the log-probability scores output by the baseline parser, and the triangles and squares represent the average candidate-rank as derived from the pairwise F-score and pairwise D-score comparisons, respectively. Note that although the lines representing the pairwise comparisons appear nearly identical, they do in fact differ on about 20% of the data points. As can be inferred from the graphs, both of the pairwise metrics come closer to the true ranking (shown as a dotted line along the diagonal), and thus more closely approximate the true candidate ranking than does the log-probability score. Also, the pairwise metrics appear to more accurately rank the candidates in the bottom half of the n-best lists. However, none of the three scoring metrics are significantly more accurate in determining the best-ranked (one-best) candidate.

We can take a more exact measurement of rank-order accuracy by calculating a margin between the rank value assigned by each of our three scoring metrics and the true rank value of each candidate. Let i equal the true rank value of the candidate, as



CJ05 HR07 MCJ06

Figure 1: Rank-order accuracy produced by re-ordering the n-best candidates according to the log-probability score, pairwise F-score, and pairwise D-score of each candidate; pairwise F-score and D-score lines appear to overlap in these graphs. Compares candidate rank, as determined by one of the three baseline-scores, against the true rank of each candidate (dotted black line).

calculated by evaluating each candidate against the known true solution then ranking the candidates according to F-score, and j equal the rank of the candidate as derived from each of our three scoring metrics. We can follow Shen and Joshi (2005) by calculating *even* and *uneven margins*. *Even margins* are calculated as $(i - j)$, such that ranking mistakes are equally penalized regardless of the rank value; thus ranking the second candidate as the third is as bad as ranking the 49th candidate at the 50th. *Uneven margins* are calculated as $(1/i - 1/j)$ so that errors in the lower ranks are not penalized as heavily as errors in the higher ranks.

Using the baseline system score as a reranking feature results in an even margin of 744 and an uneven margin of 6.7 on WSJ section 24. In comparison, using either of the pairwise scores as the reranking score feature results in an even margin of 467 and an uneven margin of 6.3, demonstrating that, indeed, both the pairwise F-score and D-score provide a more accurate ranking than the parser log-probability score.

9 Conclusion & Future Work

In this paper we have shown that different levels of reranker performance can be obtained by altering the functions for deriving the baseline system score as a feature in the reranking model. By using characteristics of the list itself to re-score and re-order candidates within the list, we demonstrated that scores which better capture the relative differences between candidates can provide utility in a reranking model. The methods presented herein, for deriving candidate scores and learning feature-weights for these scores, provide several additions to current-best practices for training reranking models.

In our future work, we intend to pursue different methods for empirical optimization within the reranking framework. We also plan to explore different options for representing baseline score features, such as adding a non-linear representation of the score as a feature in the linear reranking model.

References

A. Bernal, K. Crammer, A. Hatzigeorgiou, and F. Pereira. 2007. Global discriminative learning for higher-accuracy computational gene prediction. *PLoS Computational Biology*, 3.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 173–180.

Michael Collins, Murat Saraclar, and Brian Roark. 2005. Discriminative syntactic language modeling for speech recognition. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*.

J. Fiscus. 1997. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*.

V. Goel, S. Kumar, and W. Byrne. 2000. Segmental minimum Bayes-risk ASR voting strategies. In *Proceedings of ICSLP*, pages 139–142.

Kristy Hollingshead and Brian Roark. 2007. Pipeline iteration. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 952–959.

Kristy Hollingshead, Seeger Fisher, and Brian Roark. 2005. Comparing and combining finite-state and context-free parsers. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 787–794.

Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul Vitànyi. 2004. The similarity metric. *IEEE Transactions on Information Theory*, 50.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the HLT-NAACL Annual Meeting*, pages 152–159.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30.

Brian Roark, Murat Saraclar, and Michael Collins. 2007. Discriminative n-gram language modeling. *Computer Speech and Language*, 21:373–392.

Izhak Shafran and William Byrne. 2004. Task-specific minimum Bayes-risk decoding using learned edit distance. In *Proceedings of the 8th International Conference on Spoken Language Processing (ICSLP)*, volume 3, pages 1945–48.

Libin Shen and Aravind K. Joshi. 2005. Ranking and reranking with perceptron. *Machine Learning, Special Issue on Learning in Speech and Language Technologies*, 60.

Libin Shen, Anoop Sarkar, and Franz Och. 2004. Discriminative reranking for machine translation. In *Proceedings of the HLT/NAACL Annual Meeting*.

Charles Sutton, Michael Sindelar, and Andrew McCallum. 2006. Reducing weight undertraining in structured discriminative learning. In *Proceedings of the HLT/NAACL Annual Meeting*, pages 89–95.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International COLING*, pages 947–953.

Peter N. Yianilos. 1991;2002. Normalized forms for two common metrics. Technical report, NEC Research Institute.