Taming Structured Perceptrons on Wild Feature Vectors

Ralf D. Brown
Carnegie Mellon University Language Technologies Institute
5000 Forbes Avenue, Pittsburgh PA 15213 USA
`ralf+@cs.cmu.edu`

Abstract

Structured perceptrons are attractive due to their simplicity and speed, and have been used successfully for tuning the weights of binary features in a machine translation system. In attempting to apply them to tuning the weights of real-valued features with highly skewed distributions, we found that they did not work well. This paper describes a modification to the update step and compares the performance of the resulting algorithm to standard minimum error-rate training (MERT). In addition, preliminary results for combining MERT or structured-perceptron tuning of the log-linear feature weights with coordinate ascent of other translation system parameters are presented.

1 Introduction

Structured perceptrons are a relatively recent (Collins, 2002) update of the classic perceptron algorithm which permit the prediction of vectors of values. Initially developed for part of speech taggers, they have been applied to tuning the weights of the features in the log-linear models used by statistical machine translation (Arun and Koehn, 2007), and found to have performance similar to the Margin-Infused Relaxed Algorithm (MIRA) by Crammer and Singer (2003; 2006) and Minimum-Error Rate Training (MERT) by Och (2003). Parameter tuning is an important aspect of current data-driven machine translation systems, as an improper selection of feature weights can dramatically reduce scores on evaluation metrics such as BLEU (Papineni et al., 2002) or METEOR (Banerjee and Lavie, 2005).

When we recently added new features to the CMU-EBMT translation system (Brown, 1996; Brown, 2008)[1], in addition to splitting a number of composite features into their components, our previous method of parameter tuning via coordinate ascent[2] became impractical. With now more than 50 features partaking in the scoring model, MERT no longer seemed a good choice, as the common wisdom is that it is not able to reliably optimize more than about 20 features (Chiang et al., 2008).

We had been using coordinate ascent because of a need to tune a substantial number of parameters which are not directly part of the log-linear model which can be tuned by MERT or similar methods. Our system generates a translation lattice by runtime lookup in the training corpus rather than using a precomputed phrase table, so important parameters include

- the size of the sample of retrieved training instances for a given input phrase which are aligned,
- the weight of source features for ranking training instances during sampling, and
- the minimum alignment score to accept a translation instance

Decoder parameters which are important to tune, but which are generally not mentioned in the literature include

- how many alternative translations of a phrase to consider during decoding,
- the size of the reordering window, and
- the rank of the language model (4-gram, 5-gram, etc.)

In addition, it is desirable to tune parameters such as beam width to minimize translation time without degrading performance.

[1]Source code for CMU-EBMT is available from `http://cmu-ebmt.sourceforge.net`.

[2]Coordinate ascent is described in more detail in Section 7.

As a result of the non-model parameters, a full system tuning will involve multiple runs of the tuning algorithm for the feature weights, since the other parameters will affect the optimal weights. Thus, speed is an important consideration for any method to be used in this setting. The structured perceptron algorithm is ideally suited due to its speed, provided that it can produce competitive results.

2 Related Work

The perceptron algorithm (Rosenblatt, 1958) itself is over 50 years old, but variations such as voted and averaged perceptrons have gained popularity in the past ten years. In particular, Collins (2002) adapted the perceptron algorithm to structured prediction tasks such as part of speech tagging and noun phrase chunking. Arun and Koehn (2007) subsequently applied Collins' structured perceptron algorithm to the task of tuning feature weights in a statistical machine translation system, demonstrating the extreme scalability of the algorithm by applying it to vectors containing four to six million binary features. However, their work left open the question of how well structured perceptrons would deal with continuous-valued features. They were unable to apply a language model due to the lack of continuous-valued features and hence had to compare performance against a standard statistical machine translation (SMT) system which had been stripped of its language model, with a consequent loss of several BLEU points in performance.

During the same period, Crammer et al (2003; 2006) developed a number of "ultraconservative" learning algorithms, including MIRA, the Margin-Infused Relaxed Algorithm (which was also applied to large binary feature vectors by Arun and Koehn) and variations of what they referred to as Passive-Aggressive algorithms including PA-I and PA-II. These algorithms have in common the notion of updating a weight vector "just enough" to account for a new training instance which is incorrectly predicted by the existing weight vector. In contrast, the perceptron algorithm aggressively updates the weight vector and relies on averaging effects over the whole of the training set.

3 Structured Perceptrons

The structured perceptron algorithm can be applied to tasks where the goal is to select the best among competing hypotheses, where each hypothesis has an associated vector of feature values and the score for a hypothesis is a linear combination of its feature values.

Beginning with a zero vector for the feature weights, the structured perceptron algorithm iterates through each element of the training set, updating the weight vector after processing each training instance. The training set is processed repeatedly (each pass is known as a *training epoch*) until convergence. The update step is very simple: if the best hypothesis according to the product of feature vector and weight vector is not the correct answer, add the difference between the feature vectors of the correct answer and the model's selected answer to the weight vector.

Thus, the entire algorithm may be summarized with just two equations:

$$\vec{w} \leftarrow 0 \quad (1)$$

$$\vec{w} \leftarrow \vec{w} + (\Phi_{oracle} - \Phi_{top1}) \quad (2)$$

where Φ_x is the feature vector $(\phi_1, \phi_2, ..., \phi_n)$ for hypothesis x.

Repeated application of Equation 2 results in a weight vector which reflects the relative importance (on average) of each feature to making the correct selection. Since selecting the best hypothesis is an `arg max` operation, the absolute magnitudes of the weights are not important.

4 More Conservative Updates for Structured Perceptrons

One issue which arises in using learning algorithms for machine translation is that there is no one correct answer. In addition, it may not even be possible for the MT system to generate the reference translation at all. This is commonly addressed by using the highest-scoring (by some metric such as BLEU) translation which the system *can* generate as a pseudo-oracle.

Our initial implementation closely followed the description in (Arun and Koehn, 2007), including the refinement of using the objective-function score of the pseudo-oracle translation from the n-best list to modulate the learning rate of the update step, i.e.

$$\vec{w} \leftarrow \vec{w} + S_{\Phi_{oracle}} \times (\Phi_{oracle} - \Phi_{top1}) \quad (3)$$

As can be seen, the difference between Equations 2 and 3 is simply the additional factor of $S_{\Phi_{oracle}}$.

While we initially used sentence-level smoothed BLEU as the objective function, we found it to perform very poorly (the full BLEU scores on the Haitian Creole tuning set were well below 0.10), and instead adopted the Rouge-S (skip bigrams) metric by Lin and Och (2004a) with a maximum skip distance of four words, which was found to best correlate with human quality judgements (Lin and Och, 2004b).

In early testing, we found that both the feature weights and performance as measured by the average objective score over the tuning set oscillated wildly. Analyzing the results, it became apparent that the update function was overly aggressive. Unlike the binary features used in (Arun and Koehn, 2007), our continuous-valued features have different operating ranges for each feature, e.g. the total distance moved as a result of reordering could reach 100 on a long sentence, while the proportion of training instances with at least six words of adjacent context in the bilingual corpus is unlikely to exceed 0.05, even where sampling is biased toward training instances with adjacent context.

The first attempt to address the disparity in operating ranges was to perform feature-wise normalization on the update. Instead of taking the simple difference in feature vectors between the n-best entry with the highest log-linear score and the one with the highest objective score, we construct Φ_{diff} such that

$$\phi_i(diff) \leftarrow \frac{(\phi_i(oracle) - \phi_i(top1))}{r^2} \quad (4)$$

where

$$r \leftarrow max(0.01, max_j|\phi_i(j)|) \quad (5)$$

i.e. we estimate the operating range by finding the n-best entry with the highest magnitude value of the feature, and then divide by the square of that magnitude since large feature values also magnify the effects of weight changes. Normalization is limited by clipping the normalization factor to be at least 0.01 so that features whose values are always very near zero do not dominate the overall score.

While the feature-wise normalization did largely control the wild swings in feature weights, it did not curb the oscillations in the objective scores and produced only a minor improvement in tuning results.

We next looked at MIRA and related work on so-called Passive-Aggressive algorithms, and in particular at the update functions described in (Crammer et al., 2006). We decided on their PA-II update rule (PA-II being akin to 1-best MIRA), with which the learning step becomes

$$\vec{w} \leftarrow \vec{w} + \delta \times (\Phi_{oracle} - \Phi_{top1}) \quad (6)$$

where

$$loss \leftarrow S_{\Phi_{oracle}} - S_{\Phi_{top1}} \quad (7)$$

$$\delta \leftarrow \frac{loss}{||\Phi_{oracle} - \Phi_{top1}||^2 + \frac{1}{2C}} \quad (8)$$

with C an “aggressiveness” parameter.

This version of the update function produced the desired smooth changes in feature weights from iteration to iteration, though objective scores still do not converge. Allowing multiple passes through the tuning set before re-decoding with updated feature weights now frequently results in weights where the pseudo-oracle is the top-ranked translation in 80 to 90 percent of all sentences. None of our previous experiments had achieved even a fraction of this level due to the erratic behavior of the feature weights. However, as the extreme overfitting necessary to achieve such high rankings of the oracle translation results in poor BLEU scores, we have since used only one pass over the tuning set before re-decoding with updated weights.

5 The Final Algorithm

After the various attempts at taming the behavior of the structured perceptron approach just described, the final algorithm used for the experiments described below was

1. Structured perceptron, with
2. passive-aggressive updates,
3. run in semi-batch mode,
4. using sentence-level modified Rouge-S4 as the objective function

Semi-batch mode here means that while the perceptron algorithm updates the weight vector after each sentence, those updates are not communicated to the decoder until the end of a complete pass through the tuning set. An exception is made for the very first iteration, as it starts with uniform weights of 10^{-9} (rather than the conventional zero, which would cause problems with decoding). This permits the exact determination of the overall objective score for the weight vector which is eventually returned as the tuned optimal weights, and permits parallelization of the decoding (though the latter has not yet been implemented).

We slightly modified the Rouge-S scoring function to use the generalized F-measure

$$F_\beta = \frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall} \quad (9)$$

instead of the standard F_1, allowing us to give more weight to recall over precision by increasing β above 1.0. This change was prompted by the observation that the tuning process strongly favored shorter outputs, resulting in substantial brevity penalties from BLEU.

6 Experiments

We present the results of experiments on three data sets in the next section. The data sets are English-to-Haitian, French-to-English, and Czech-to-English.

The English-to-Haitian system was built using the data released by Carnegie Mellon University (2010). It consists of a medical phrasebook, a glossary, and a modest amount of newswire text, each available as a set of sentence pairs in English and Haitian Creole. For training, we used all of the glossary, all but the last 300 phrase pairs of the medical phrasebook (these had previously been used for development and testing of a "toy" system), and the first 12,500 sentence pairs of the newswire text. Tuning was performed using the next 217 sentence pairs of the newswire text, and the test set consisted of the final 800 sentence pairs of the newswire text. The target language model was built solely from the target half of the training corpus, as we did not have any additional Haitian Creole text.

The French-to-English system was built using the Europarl (Koehn, 2005) version 3 data for French and English. As is usual practice, text from the fourth quarter of 2000 was omitted from the training set. Tuning was performed using 200 sentences from the "devtest2006" file and all 2000 sentences of "test2007" were used as the final test set. Two target language models were built and interpolated during decoding; the first was trained on the target half of the bilingal corpus, and the second was built using the Canadian Hansards text released by ISI (Natural Language Group, 2001).

The Czech-to-English system was built using the parallel data made available for the 2010 Workshop on Statistical Machine Translation (WMT10). The target language model was built from the target half of the bilingual training corpus. Tuning was performed on a 200-sentence subset of the "news-2008-test" data, and all 2525 sentences of the "news-2009-test" data were used as unseen test data. As these experiments were the very first time that the CMU-EBMT system was applied to Czech, there are undoubtedly numerous pre-processing and training improvements which will increase scores above the values presented here.

Parameter tuning was performed using CMERT 0.5, the reimplemented MERT program included with recent releases of the MOSES translation system (specifically, the version included with the 2010-04-01 release), the annealing-based optimizer included with Cunei (Phillips and Brown, 2009; Phillips, 2010), and the Structured Perceptron optimizer. Feature weights were initialized to a uniform value of 1.0 for MERT and 10^{-9} for annealing and Perceptron (since the usual zero causes problems for the decoder). Both versions of MERT were permitted to run for 15 iterations or until features weights converged and remained (nearly) unchanged from one iteration to the next, using merged n-best lists from the current and the three most recent prior iterations. Annealing was run with gamma values from 0.25 to 4.0, skipping the entropy phase. The Structured Perceptron was allowed to run for 18 iterations and to choose the weights from the iteration which resulted in the highest average Rouge-S score for the top translation in the n-best list. For French-English, this proved to be the sixth iteration, while for English-Haitian it was the twelfth. We have found that the objective score increases for the first six to eight iterations of SP, after which it fluctuates with no trend up or down (but occasionally setting a new high, which is why we decided to run 18 iterations).

For French-English, we determined the best value of β for the Rouge-S scoring to be 1.5, and the best value of the aggressiveness parameter C to be 0.1, using a 40-sentence subset of the French-English tuning set, and then applied those value for the full tuning set. For English-Haitian, we used $\beta = 1.2$ and $C = 0.01$ (lower values of C provide more smoothing and overall smaller updates, which is necessary for sparse or noisy data). Due to limited time prior to submission, the English-Haitian values for β and C were re-used for Czech, with no attempt at tuning.

7 Combining Log-Linear Tuning with Coordinate Ascent

As noted in the introduction, translation systems using SMT-style decoders incorporate various features that affect performance (and/or speed), but which do not contribute directly to the log-linear scoring model. Thus, neither MERT nor the structured perceptron training presented in this paper is a complete solution for parameter tuning.

The CMU-EBMT system has long used a coordinate ascent approach to parameter tuning. Each parameter is varied in turn, with the MT system performing a translation for each setting, and the value which produces the best score is retained while the next parameter is varied. If the best scoring value is the highest or lowest in the list of values to be checked, the range is extended; likewise, unless the interval between adjacent values is already very small, the intervals on each side of the highest-scoring value (which is not one of the extremes) is divided in half and the two additional points are evaluated. This process continues until convergence (cycling through all parameters without changing any of them) or until a pre-set maximum number of parameter combinations is scored. Naturally, the approach becomes slower as the number of parameters increases, but it was still (barely) practical with 20 to 25 parameters.

A recent change in the internals of CMU-EBMT led to a decomposition of multiple composite scores and the addition of numerous others, ballooning the total number of tunable parameters to more than 60. Fortunately, most of the tunable parameters are feature weights, which can all be treated as a unit, leaving only about a dozen features for coordinate ascent.

The tuning program operates by calling an evaluation script which in turn invokes the machine translation on a modified configuration file provided by the tuner and returns the score corresponding to the given parameter settings. When given an optional flag, the evaluation script first invokes either MERT or SP to further adjust the parameters before performing the actual evaluation, and modifies the given configuration file accordingly. The tuner reads the modified parameters from the configuration file and stores then for further use.

Both MERT and SP can produce settings which actually *decrease* the resulting BLEU score, since they are optimizing toward a surrogate metric. If the evaluation score after an invocation of MERT or SP is less than 0.98 times the previous best score, the parameter settings are rolled back; otherwise, the best score is set to the evaluation score. This permits MERT/SP to move the parameters to a different space if necessary, without allowing them to substantially degrade overall scores.

There was time for only one experiment involving complete tuning, as summarized in Table 4. Starting with the Haitian-Creole feature weights found for the results in Table 1, the tuner randomly perturbed the non-feature-weight parameters by a small amount (up to 2% relative) twenty times, then started coordinate ascent from the best-scoring of those 20 trials. The tuner requested a MERT/SP run before ascending on the first parameter, and after every fourth parameter was processed thereafter. Because both MERT and SP started from previously-tuned feature weights, the number of iterations was reduced from 15 to 4 for MERT and from 18 to 5 for SP. The maximum number of parameter combinations for coordinate ascent was set to 750, which is approximately four cycles through all parameters (the exact number of combinations per cycle varies, as the tuner can add new combinations by extended the range which is searched or adding intermediate points around a maximum).

In Table 4, the three different Perceptron entries refer to the results starting from the previous experiment's feature weights ("Perceptron 1"), starting from the results of the complete tuning ("Perceptron 2"), and starting from uniform feature weights ("Perceptron 3"). The third run was stopped before convergence due to the looming submission deadline.

8 Results

Tables 1, 2, and 3 present the results of running the tuning methods on the English-Haitian, French-English, and Czech-English data sets, respectively. Performance is shown both in terms of the time required to perform a tuning run as well as the BLEU score achieved using the resulting feature weights.

Structured perceptrons are the clear winner for speed, thanks to the simplicity of the algorithm.

Method	Run-Time	Iter	BLEU (dev)	BLEU (test)	#words / ratio
CMERT 0.5	73m	5	0.0993	–	
new MERT	58m	3	0.0964	–	
CMERT 0.5[1]	138m	15	0.1073	0.0966	22298 / 1.213x
new MERT[1]	187m	15	0.1516	0.1347	17375 / 0.945x
Perceptron	22m	18	0.1619	0.1534	15565 / 0.847x

[1] omitting several unused features, as noted in the text

Table 1: English-to-Haitian tuning performance

Method	Run-Time	Iter	BLEU (dev)	BLEU (test)	#words / ratio
CMERT 0.5	3h53m	15	0.12952	0.13927	100875 / 1.709x
new MERT	5h52m	15	0.22533	0.23315	60354 / 1.023x
Annealing	6h46m	-	0.25017	0.25943	58518 / 0.992x
Perceptron	1h23m	18	0.24214	0.26048	57408 / 0.973x

Table 2: French-to-English tuning performance

While MERT takes two to three times as long to process ten random starting points as it does to decode the test set, SP is three orders of magnitude faster than decoding. As a result, SP tuning requires one-third or less of the time that MERT does, even though we used 18 iterations of SP compared to 15 for MERT. Note that the time difference between the two versions of MERT is in part due to different amounts of time spent decoding as a result of the different feature weights.

MERT unexpectedly has considerable difficulty with our new feature set, as can be seen by its much lower BLEU scores, particularly in the case of CMERT. An analysis of the actual feature weights produced by MERT shows that it places nearly all of the mass on a single feature, and that the feature receiving the bulk of the mass *changes from iteration to iteration*. In contrast, SP produces BLEU scores consistent with those produced by pure coordinate ascent prior to the proliferation of features.

We believe that the difference in performance between the two versions of MERT is due primarily to the simple difference in output format: CMERT 0.5 prints its tuned weights using a fixed-point format having six digits after the decimal point, while the new MERT program prints using scientific notation. Because the tuned weight vector is highly skewed, most features have low weights after L_1 normalization, and thus CMERT truncated many weights to zero (and indeed, loses significant digits for any features assigned weights less than 0.1), including such critical weights as length features and language model scores. We suspect that this preservation of significant digits contributes substantially to the improved BLEU scores Bertoldi et al (2009) reported for the new implementation compared to CMERT.

The features which, at one time or another, receive the bulk of the mass have one thing in common: for most translations, they have a default value, and in a small proportion of cases they have a value which varies from the default by only a small amount. Initially, most such features had a default value of zero in CMU-EBMT, but this meant that the line optimization in MERT had absolutely no constraint on raising the weight of the feature, and thus obtaining feature vectors where one feature has 10^{18} or even 10^{20} times the weight of any other feature. The same problem occurs with features that are unused but have a small jitter in their values due to rounding errors, for example, if there are no document boundaries (as is the case for the Haitian data described previously), the document-similarity score may be 1.000000 for 99% of the arcs in the translation lattices and 0.999999 for the remainder. Offsetting the mostly-zero features so that their default value is 1 or -1 (depending on the sense of the feature) and eliminating unused features mitigated but did not entirely solve the problem. In Table 1, two results are shown for both CMERT and new MERT; the first includes all 52 features while the second excludes five features which are not used in a baseline-trained CMU-EBMT system. In the former case, both programs placed all the mass on a single fea-

Method	Run-Time	Iter	BLEU (dev)	BLEU (test)
new MERT	56m	15	0.0584	0.0743
Perceptron	14m	18	0.0830	0.1163

Table 3: Czech-to-English tuning performance

Method	Run-Time	BLEU (dev)	BLEU (test)	length ratio
new MERT	48h	0.1821	0.1633	0.942
Perceptron 1	25h	0.1675	0.1547	0.833
Perceptron 2	38h	0.1738	0.1597	0.837
Perceptron 3	12h*	0.1705	0.1647	0.939

* truncated run (see text)

Table 4: English-to-Haitian tuning performance (including coordinate ascent)

ture and left all the others at 10^{-14} or less (displayed as 0.000000 in the case of CMERT).

The full tuning runs summarized in Table 4 show that SP is often competitive with MERT while running more quickly, but still requires further analysis to determine the causes of variability in its performance. One initial conclusion from examining the logs of the SP runs is that weight updates are perhaps *too* conservative when applied in conjunction with coordinate ascent. While MERT frequently shifted settings in response to changes in the non-feature parameters, SP rarely does so, typically preferring to retain the existing feature weights as the best setting encountered during the five iterations performed at each invocation. The "Perceptron 3" run starting with small uniform feature weights resulted from the observation that a first, buggy attempt at integration reached tuning-set BLEU scores in excess of 0.18 before early termination. The bug in question was that many of the feature weights were initially read in from the configuration file as zero rather than the correct value.

As shown in the rightmost column of Tables 1, 2 and 4, the Perceptron algorithm tends toward short output, yielding translations which are about 97% as long as the reference translation in French-English, a mere 85% as long for English-Haitian, and even shorter than that in two of three Czech runs. This tendency towards short translations prompted the inclusion of the β parameter – the French-English output was originally much shorter, but β has little effect on Haitian given the sparse training data. The extremely long output for CMERT on French-English is due to a large number of zero weights, including those for length features.

9 Conclusion and Future Work

Structured perceptrons with passive-aggressive updates are a viable alternative to the usual MERT feature-weight tuning, particularly where the number of features exceeds that which MERT can reliably handle, or when some of the features have characteristics which confuse MERT. Structured perceptrons are also a good alternative where speed is important, such as in a hybrid tuning scheme which alternates between (re-)tuning the log-linear model and performing coordinate ascent on parameters which do not directly contribute weight to the log-linear model.

We have thus far implemented two objective functions which operate on individual sentences without regard for choices made on other sentences. When the final evaluation metric incorporates global statistics, however, an objective function which takes them into account is desirable. For example, when using BLEU, it makes a big difference whether individual sentences are both longer and shorter than the reference or systematically shorter than the reference, but these two cases can not be distinguished by single-sentence objective functions. Our plan is to implement a windowed or moving-average version of BLEU as in (Chiang et al., 2008).

We also plan to further speed up the tuning process by parallelizing the decoding of the sentences in the tuning set. As we have used a semi-batch update method which leaves the decoder's weights unchanged for an entire pass through the tuning set, there is no data dependency between individual sentences, allowing them to be decoded in parallel. The perceptron algorithm itself remains sequential, but as it is three orders of magnitude faster than the decoding, this will have negligible impact on overall speedup factors until hundreds of CPUs are used for simultaneous decoding.

References

Abhishek Arun and Phillip Koehn. 2007. Online Learning Methods for Discriminative Training of Phrase Based Statistical Machine Translation. In *Proceedings of the Eleventh Machine Translation Summit (MT Summit XI)*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgements. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-2005)*, June.

Nicola Bertoldi, Barry Haddow, and Jean-Baptiste Fouet. 2009. Improved Minimum Error Rate Training in Moses. *The Prague Bulletin of Mathematical Linguistics*, pages 1–11, February.

Ralf D. Brown. 1996. Example-Based Machine Translation in the PANGLOSS System. In *Proceedings of the Sixteenth International Conference on Computational Linguistics*, pages 169–174, Copenhagen, Denmark. http://www.cs.cmu.edu-/~ralf/papers.html.

Ralf D. Brown. 2008. Exploiting Document-Level Context for Data-Driven Machine Translation. In *Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas (AMTA-2008)*, October. http://www.amtaweb.org/papers/-2.02_Brown.pdf.

Carnegie Mellon University. 2010. Public release of haitian-creole language data, January. http://-www.speech.cs.cmu.edu/haitian/text.

David Chiang, Yuval Marton, and Philis Resnik. 2008. Online Large-Margin Training of Syntactic and Structural Translation Features. In *Proceedings of the Conference on Empirical Methods in Natural Langauge Processing (EMNLP-2008)*, pages 224–233, October.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of EMNLP-2002*. http://people.-csail.mit.edu/mcollins/papers/-tagperc.pdf.

Koby Crammer, Ofer Deke, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online Passive-Aggressive Algorithms. *The Journal of Machine Learning Research*, 7:551–585, December.

Koby Cranmer and Yoram Singer. 2003. Ultraconservative Online Algorithms for Multiclass Problems. *The Journal of Machine Learning Research*, 3:951–991, March.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the Tenth Machine Translation Summit (MT Summix X)*, pages 79–86.

Chin-Yew Lin and Franz Joseph Och. 2004a. Automatic Evaluation of Machine Translation Quality using Longest Common Subsequence and Skip-Bigram Statistics. In *Proceedings of ACL-2004*.

Chin-Yew Lin and Franz Joseph Och. 2004b. ORANGE: A Method for Evaluating Automatic Evaluation Metrics for Machine Translation. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*.

USC Information Sciences Institute Natural Language Group. 2001. Aligned *Hansards* of the 36th Parliament of Canada, Release 2001-1a. http://-www.isi.edu/natural-language/-download/hansard/.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics (ACL-2003)*, Sapporo, Japan, July 6–7.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, July. http://acl.ldc.upenn.edu/P/P02/.

Aaron B. Phillips and Ralf D. Brown. 2009. Cunei Machine Translation Platform: System Description. In *Proceedings of the Third Workshop on Example-Based Machine Translation*, Dublin, Ireland, November.

Aaron B. Phillips. 2010. The Cunei Machine Translation Platform for WMT'10. In *Proceedings of the ACL 2010 Joint Fifth Workshop on Statistical Machine Translation and Metrics MATR*, July.

F. Rosenblatt. 1958. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65:386–408.