Discriminative Syntactic Reranking for Statistical Machine Translation

Simon Carter and **Christof Monz**
ISLA, University of Amsterdam
Science Park 107, 1098 XG Amsterdam, The Netherlands
[s.c.carter] and [c.monz]@uva.nl

Abstract

This paper describes a method that successfully exploits simple syntactic features for n-best translation candidate reranking using perceptrons. Our approach uses discriminative language modelling to rerank the n-best translations generated by a statistical machine translation system. The performance is evaluated for Arabic-to-English translation using NIST's MT-Eval benchmarks. Whilst parse trees do not consistently help, we show how features extracted from a simple Part-of-Speech annotation layer outperform two competitive baselines, leading to significant BLEU improvements on three different test sets.

1 Introduction

Common approaches Statistical Machine Translation (SMT) formulate the translation task as:

$$\hat{e} \approx \mathrm{argmax}_{\mathrm{e}} p(f|e)p(e), \qquad (1)$$

where the translation model is given by $p(f|e)$, and $p(e)$ represents a language model. The most commonly used language models (LMs) are generative word-based n-gram models, which use the Markovian assumption that the probability of a word in a string can be approximated by considering only the limited history of the previous $n-1$ words.

Whilst LMs remain fundamental to SMT systems, they suffer from a number of flaws: first, given the local nature of n-gram models, there is a strong motivation for the applicability of syntactic LMs that can take into account long-range dependencies that are difficult to model by the limited context of n-gram models. However, attempts to integrate syntactic information into language models used by SMT systems have met with mixed success (Och et al., 2004; Post and Gildea, 2008).

Secondly, language model performance is sensitive to changes in genre and domain between training and test sets (Rosenfeld, 2000). Even when training and testing on very similar domains such as news, it has been shown that mundane differences between two sources of news sets can lead to a notable decrease of performance (Rosenfeld, 1996).

Thirdly, the best choice according to the language model can often be outvoted by other models (i.e phrase table, re-ordering model) during the decoding process. This leads to disfluent translations (Post and Gildea, 2008).

Finally, standard generative LMs and full or partial syntax-based models are almost exclusively trained on well-formed English. Given the large feature space they operate in, the accurate assignment of probabilities to unseen events is a difficult problem, and has been a major field of research for the past sixty years (for a detailed overview on language modelling and smoothing techniques, see (Chen and Goodman, 1998)).

Applying models in a reranking phase to achieve better results is a standard technique in SMT (Kumar et al., 2004). In a reranking approach, an SMT system outputs a list of the top n translations (an n-best list). A discriminative language model (DLM) (Roark et al., 2007) takes these n-best lists and reranks them according to a weight vector and a function $\phi(\cdot)$ that maps a sentence into a feature

Perceptron

```
1:  $w \leftarrow 0$
2:  for $t = 1$ to T do
3:      for $i = 1$ to N do
4:          $y^i \leftarrow ORACLE(x^i)$
5:          $z^i \leftarrow argmax_{z \in GEN(x^i)} \phi(z) \cdot w$
6:          if $z^i \neq y^i$ then
7:              $w \leftarrow w + \phi(y^i) - \phi(z^i)$
8:          end if
9:      end for
10: end for
11: return $w$
```

Figure 1: The standard perceptron algorithm

space. The best hypothesis according to the DLM is then returned as the selected translation. DLMs are advantageous as they can make use of negative examples and allow for a relatively easy inclusion of syntactic features that go beyond simple word n-grams.

As our DLM, we use the perceptron algorithm for learning the weight vector. Building on the framework of Roark et al. (2007), Li and Khudanpur (2008) successfully applied the perceptron to SMT using lexical, n-gram features. In this paper, we investigate the addition of syntactic features to an n-gram based perceptron when applied to SMT reranking, taking as a starting point the work of Collins et al. (2005). In particular, we show that:

- the use of full parse tree features do not provide the improvements expected, however,
- improvements can be gained by using features extracted from a novel, context-insensitive Part-of-Speech (POS) tagger, and,
- these features lead to larger gains than using a state of the art conditional random field (CRF) POS tagger on two of the three test sets.

The remainder of this paper is organised as follows: Section 2 describes different parameter estimation methods. Section 3 introduces the syntactic features used for reranking. Section 4 describes the experimental set up and results. Related work is discussed in Section 5.

2 Parameter Estimation

Different parameter estimation methods to estimate the weight vector w for a DLM have been previously examined in the speech domain (Roark et al., 2004b; Roark et al., 2007). These include optimising the log-likelihood under a log-linear model, a batch algorithm which requires processing all the data before outputting a weight vector as an answer, and approximating a 0/1 loss through the perceptron update rule, an online algorithm which examines and updates the parameter vector sequentially.

The reader is referred to (Roark et al., 2004b; Emami et al., 2007) for a discussion on the benefits of the log-linear model and the perceptron. Given this paper examines the use of a syntactic feature space, which is larger than an already large n-gram feature space, and that perceptrons perform feature selection as a consequent of its learning procedure, we opt to use the perceptron algorithm.

2.1 Perceptron

The perceptron, proposed by (Rosenblatt, 1958) is an error minimisation learner that, assuming linearly separable data, can be shown to converge to a solution that perfectly classifies the data (Freund and Schapire, 1999).

The standard perceptron algorithm is shown in Figure 1. The algorithm takes as input a set of n-best lists X, a function $GEN(x^i)$ that enumerates over each sentence in a n-best list x^i, and an oracle function $ORACLE(x^i)$ that determines the best translation (oracle best) for each of the n-best lists x^i according to the BLEU metric (Papineni et al., 2002). As DLMs make comparisons on the sentence level, we use sentence level BLEU with additive smoothing (Lin and Och, 2004). There are discrepancies between sentence and corpus-level BLEU, however we find sentence-level BLEU sufficient for reranking SMT. T defines the number of iterations and N defines the size of the test set, which in our case is the number of n-best lists. The algorithm iterates over the n-best lists in a sequential manner (lines 2 and 3). If the selected hypothesis and oracle best sentence match, then the algorithm continues to the next n-best list. Otherwise, the weight vector is updated (line 7). At the end, it returns a weight vector as its solution (line 11).

To use the weight vector returned by the perceptron algorithm, each sentence z in an n-best list is scored by:

$$S(z) = \beta\phi_0(z) + w \cdot \phi(z) \tag{2}$$

The SMT model score for each translation hypothesis $\phi_0(z)$ is weighted by β. Roark et al. (2007) argue that, whilst possible to include $\phi_0(z)$ as a feature of the perceptron model, this may lead to undertraining, so we adhere to the convention of using a fixed value for β.

To score an n-best list x^i we use the weight vector returned by the perceptron to assign a score to each sentence and select the best one:

$$z^* = \mathrm{argmax}_{\mathrm{z}\in\mathrm{GEN}(\mathrm{x}^\mathrm{i})} S(z) \tag{3}$$

2.2 Perceptron Variants

A shortcoming of the perceptron is that it can be unstable if the training data is not linearly separable. A number of solutions have been proposed in the literature. One solution proposed is to use an averaged perceptron (Freund and Schapire, 1999), where the parameter vector w output by the algorithm is averaged over each instance $w_{avg} = \Sigma_{t=1}^{T}\Sigma_{i=1}^{N}\frac{w_t^i}{N \cdot T}$. Another solution is the pocket perceptron (Gallant, 1999; Collins and Duffy, 2002), where the weight vector returned is the one that correctly classifies the most training instances in a row, keeping an optimal model in its 'pocket'. A third solution, called the committee or voting perceptron, keeps a cache of optimal models, sorted by their success counts (Roark et al., 2004a; Elsas et al., 2008). The cache sizes differentiate the voting and committee perceptron.

3 Syntactic Features

The syntactic features used for reranking are extracted from either full parse trees (Section 3.1) or POS sequences (Section 3.2). Using parsers allows us to extract features that are global to the sentence and relay deep structural information. Unfortunately they are relatively slow and memory intensive, and may fail to return a parse for large sentences.[1] On the other hand, POS taggers, whilst outputting no higher-level syntactic information, are more reliable, efficient and robust as they always output a complete POS sequence for a given input sentence.

[1]They have $O(n^3)$ complexity and have exponential growth in memory usage, both in relation to sentence length.

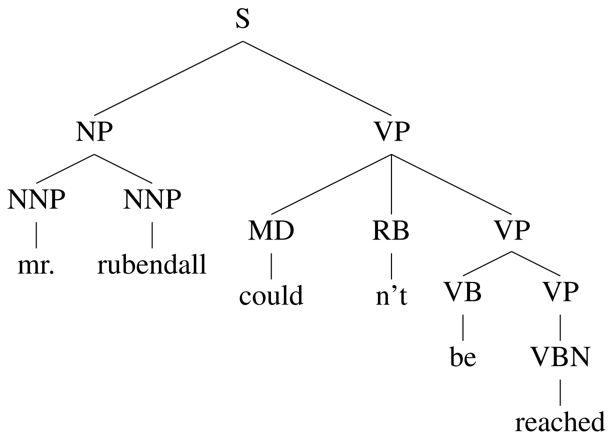

Figure 2: An example parse tree.

(a)

mr/NNP rubendall/NNP could/MD n't/RB be/VB reached/VBN

(b)

mr/NP^b rubendall/NP^c could/VP^b n't/VP^c be/VP^b reached/VP^b

(c)

mr/NNP-NP^b rubendall/NNP-NP^c could/MD-VP^b n't/RB-VP^c be/VB-VP^b reached/VBN-VP^b

Figure 3: Sequences extracted from full parse trees.

3.1 Parsers

Three types of rules are extracted from full parse trees; the first type is based on sequences, the second is based on head information and lastly the raw context free grammar rules are extracted. These feature types are used as a starting point and are taken from Collins et al. (2005), who used them for improving ASR output.

Figure 2 shows an example sentence with its parse, and Figure 3 shows the sequences from which the first set of syntactic features are extracted. In 3(a), the sequences are the POS tags for each word. 3(b) captures chunk-based sequences by associating with each word the first non-POS ancestor node. For each word it is also indicated whether it starts or continues a shallow chunk. The features 3(c) are similar, but also include the POS tag of each word.

The second set of syntactic features is based on lexical heads within the tree. Heads are extracted using the hand crafted rules defined in Appendix A of (Collins, 1999). The simplest set of these features models head-parent relationships, where for each Non-Terminal (NT) in a tree, features of the form NT/H$_{NT}$ and NT/HP$_{NT}$ are extracted, where H$_{NT}$ represents the lexical head of the NT, and HP$_{NT}$ is the POS tag of the corresponding lexical head of NT. Head-to-head dependencies within the parse tree are extracted to catch long range dependencies. For a full description of these features, see (Collins et al., 2005).

The last set of features examined from full parse trees are context free rules. The take the form NT $\rightarrow$ NT NT and NT $\rightarrow$ WORD. These are first order, lexical and non-lexical rules. The use of raw parse tree constituents have been unhelpful (Och et al., 2004). However we include them in this context as we may learn certain rule associations to be helpful in the discriminative context.

3.2 POS Taggers

In addition to full syntactic parse information we also consider shallow syntactic annotations obtained by using Part-of-Speech taggers. Two tagging approaches are used: a Conditional Random Fields tagger and a very simple tagger, using maximum likelihood estimates to assign POS tags to words without using any context. The simple model does not use any smoothing, meaning that out-of-vocabulary items are simply assigned <UNK> as their tag.

3.3 Feature Types

From sequences, extracted from either full parse trees or taggers, there are a number of features we can extract. Potential feature types include:

1. $(t_{i-2}t_{i-1}t_i), (t_{i-1}, t_i), (t_i), (t_i w_i)$
2. $(t_{i-2}t_{i-1}w_i)$
3. $(t_{i-2}w_{i-2}t_{i-1}w_{i-1}t_i w_i), (t_{i-2}t_{i-1}w_{i-1}t_i w_i)$, $(t_{i-1}, w_{i-1}, t_i, w_i), (t_{i-1}, t_i, w_i)$

Here w_i refers to a word at position i, and t_i refers to a either a POS or NT tag, according to the sequence type. We will refer to these sequences simply as sequence 1, 2 or 3. Collins et al. (2005) show that for ASR sequences 2 and 3 did not provide improvements over sequence 1. Given we are working in SMT, where re-ordering problems are more prevalent, it will be interesting to see if they bring further improvements.

It is also possible to extract features from the POS layers that capture frequency based information. In particular, we wish to model the frequency of POS types for a given translation length. These include features of the form:

$$length(x)/num(POS, x)$$

The length of a sentence is represented by $length(x)$, and the frequency a specific POS tag occurs in the hypothesis translation x is $num(POS, x)$. These features tie the number of POS tags to the length of a sentence. In this way we hope to model the under-or-over production of certain POS types. In this paper we examine five such types: verbs, nouns, adverbs, adjectives and determiners. A similar feature is one that models a lack of certain POS types, regardless of sentence length. Here again we model a lack of either verbs, nouns, adverbs, adjectives or determiners.

The last feature, verb agreement, is meant to capture agreement between verb tenses that should match. We learn this feature by starting with each co-ordinating conjunction and comma in a sentence, and examine a window 5 words large on either side for verbs. If there are multiple verbs in this window, we return the nearest one either side. Only if we find a verb on both sides do we extract this feature.

For example, given the sentence "George/NNP was/VBD shouting/VBG and/CC screaming/VBG", the verb agreement feature would be:

VBG CC VBG

This feature can hopefully discriminate between the correct form "shouting and screaming " and the incorrect "shouting and screamed". Note this is not a tri-gram POS feature. The verbs do not have to appear directly before or after the comma or co-ordinating conjunction.

	POS Accuracy
CM2	94.4%
CRF	**97.0%**
S-POS	86.8%

Table 1: POS Tagging accuracy of the different syntactic tools we used.

4 Experimental Results

The effectiveness of the different syntactic features discussed above is evaluated for Arabic-to-English translation using NIST's MT-Eval benchmarking sets from 2002 through to 2006 (henceforth referred to as MT02, MT03, MT04, MT05 and MT06).[2]

4.1 SMT System

Moses is used as a state-of-the-art baseline SMT system for reporting experimental results (Koehn et al., 2007). It is a phrase-based MT system using stacks to organise partial translation candidates. The parameters used for the experiments discussed here are: stack size of 100, distortion limit of 6, and phrase table limit of 20.

4.2 Training Data

To build the phrase table and language model, we used various corpora distributed by the Linguistic Data Consortium (LDC), totaling 300 thousand sentence pairs.[3] Alignments were extracted using the GIZA++ toolkit (Och and Ney, 2000). The AFP and Xinhua portions of the English Gigaword corpus (LDC2003T05) and the English side of the bitext were used to build the target tri-gram language model using the SRILM toolkit with modified Kneser-Ney smoothing (Stolcke, 2002).

4.3 Parameter Optimisation

The NIST Machine Translation MT02 and MT03 sets were used as a development set for optimising the parameters of the Moses baseline SMT system using Minimum Error Rate Training (MERT) (Och, 2003). Since the parameters of the perceptron reranker need to be optimised as well, the development set was split into K folds. MERT was run on the union of the $K-1$ folds to optimise the parameters. The resulting setting was used to translate the remaining fold and to generate the n-best lists used for learning the parameter settings of the perceptron reranker. Note, that the Moses baseline was still trained on all development data at once.

To generate the parses from which the syntactic features for our perceptron reranker are extracted, the n-best lists of the development and test sets are parsed by Dan Bikel's implementation of Collins Model 2 (CM2) (2002). The parser was optimised on sections 02-21 of the Wall Street Journal of the Penn Tree Bank (PTB) (Marcus et al., 1994), approximately 40,000 sentences, with section 23 reserved for testing. As we parse SMT output, all sentences were tokenised and lowercased in accordance with the output of the SMT system prior to training the parser. Whenever the parser failed to generate a parse, the sentence was assigned the <NOPARSE> feature.

The simple uni-gram tagger and Xuan-Hieu Phan's implementation of a CRF tagger (available at http://crftagger.sourceforge.net) were trained analogously. Table 1 shows the POS accuracy of all three syntactic models on section 23 of the WSJ corpus. We refer to our simple POS tagger as S-POS.

To find the optimal β value used in equation 2.1, we did a grid search, with values at 0.1 intervals examined between 0 and 1, and values at 2^x thereafter, on the MT0203 training set.

The n-best lists contain the top 1000 most likely and distinct translation candidates (different alignments can lead to sentences which are lexically identical but have different derivations). Untranslated source words were not removed from translations.

	MT04	MT05	MT06
Moses	48.97	53.92	38.40
+ DLM n-gram	**49.57**	**54.42**	**39.08**
Oracle	61.09	66.34	50.11

Table 2: Baseline results on MT test sets. BLEU scores reported are uncased. 'DLM n-gram' refers to the competitive n-gram perceptron reranker we aim to outperform.

[2]Statistics for each set (#source sentences/#refs): MT02 (1043/4), MT03 (663/4), MT04 (1353/5), MT05 (1056/5), MT06(1796/4).

[3]The parallel text includes Arabic news LDC2004T18, automatically extracted parallel text LDC2007T08, eTIRR news LDC2004E72 and translated Arabic treebank data LDC2005E46.

	MT04	MT05	MT06
Moses	48.97	53.92	38.40
+ DLM n-gram	**49.57**	54.42	39.08
+ n-gram + POS	49.47	**54.48**	39.07
+ n-gram + SEQ-B	49.09	54.11	*39.47*
+ n-gram + SEQ-C	49.46	54.19	39.07
+ n-gram + CFG	49.53	*54.44*	**39.58**
+ n-gram + H	49.44	54.09	*33.45*

Table 3: Results on MT test sets using syntactic features from full parse trees.

4.4 Results on NIST MT Test Sets

Table 2 presents baseline results on MT04, MT05 and MT06 test sets. In addition to the Moses baseline, we compare our results to a re-implementation of the state-of-the-art perceptron re-ranker of (Li and Khudanpur, 2008), which uses uni-gram, bi-gram and tri-gram lexical features.

In Table 3 we present the results of using our perceptron rerankers with features extracted from full parse trees. All results are reported using sequence 1 type features, as we did not find further information to be helpful. SEQ-B refers to features extracted from shallow parse tag sequences shown in Figure 3(b). SEQ-C refers to the sequence shown in Figure 3(c). H refers to the head based features and CFG refers to the context-free rule based features. The models built include lexical n-grams features. The use of features from full parse trees did not help at all for MT04. For MT05, the CFG and POS feature sets show small improvements. For the MT06 test set, all syntactic models apart from H achieve improvements. These improvements against the lexical only reranker do not hold for MT04 and MT05. Given unseen test data, we do not know if the inclusion of syntactic features from full parse trees will improve or harm the translation quality of the system.

A further explanation for this under-performance is a lack of parses generated for our n-best lists. Adding syntactic features to our model increases the feature space considerably. Yet as we do not apply sentence-level thresholding to our data sets, we are unable to generate a parse for all sentences in the n-best lists. Table 4 shows the percentage of sentences in the training and test sets that we were able to retrieve a parse for (p.p.s). Every nbest list contained at least one parse. While we are able to parse over 80% of the sentences in each of the test sets, we still lack syntactic information for under 20%.

	#sentences	p.p.s%
MT0203	1282287	**87.3%**
MT04	1075563	81.9%
MT05	744049	82.6%
MT06	1526170	80.7%

Table 4: Percentage of sentences (p.p.s) that have a parse.

	MT04	MT05	MT06
DLM n-gram	49.57	54.42	39.08
DLM n-gram + POS	49.47	54.48	39.07
Improvement	-0.10	0.06	-0.01
DLM n-gram + CRF	**49.74**	54.51	39.45
Improvement	**0.17**	0.09	0.37
DLM n-gram + S-POS	49.59	**54.60**	**39.48**
Improvement	0.02	**0.18**	**0.40**

Table 5: BLEU scores and improvements when using features from our two POS taggers and POS annotations from the full tree parser. POS features extracted from a simple uni-gram, maximum likelihood tagger give largest improvements on two of three sets.

Experiments with features extracted from the POS taggers can help determine if the lack of parse annotations is a cause of the under-performance. The POS taggers provide a POS sequence for all hypothesis translations, giving us full syntactic 'coverage' of our training and test data. The results are displayed in Table 5.

The CRF DLM outperforms the n-gram only DLM model on all three test sets. The S-POS DLM yields gains over the DLM n-gram model on all three of the test sets also. Even though our S-POS tagger uses no back-off model or context, for two of the three test sets, it provides larger gains than the CRF tagger. Because the S-POS tagger results in higher scores than the CRF tagger for two of the three test sets, we only use the simple POS annotation layer for the following experiments.

Table 6 summarizes the results of using the POS tag frequency by sentence length features, the lack of syntactic POS type features, and the verb agreement features. We refer to the features that capture

	MT04	MT05	MT06
+ DLM n-gram	49.57	54.42	39.08
+ S-POS+vn+dn	49.65†	**54.60**‡	39.67‡
+ S-POS+allnum	49.65	**54.60**	39.67‡
+ S-POS+noall	**49.70**‡	54.46	**39.69**‡
+ S-POS+verbagr	49.44	54.56	39.55‡

Table 6: Model results using POS tag frequency (vn, dn and allnum), lack of POS type (noall) and verb agreement (verbagr) features. Significance at $p < 0.01$: ‡. Significance at $p < 0.05$: †.

Task	System	n-gram precision (%)			
		1	2	3	4
MT04	n-gram	**81.86**	58.36	41.72	30.28
	+syntax	81.76	**58.48**	**41.92**	**30.43**
improvement (%)		–0.1%	0.2%	0.5%	0.5%
MT05	n-gram	**83.31**	62.74	47.20	35.54
	+syntax	83.28	62.96	47.43	35.74
improvement (%)		–0.04	0.3%	0.5%	0.6%
MT06	n-gram	**74.43**	47.84	31.75	21.50
	+syntax	74.31	**47.92**	**31.87**	**21.58**
improvement (%)		–0.2%	0.2%	0.4%	0.4%

Table 7: N-gram precisions and relative improvements on MT test sets.

the number of determiners per translation sentence length as 'dn'. Similarly, we refer to the verb based equivalent features as 'vn'. The features which together capture the number of all five POS types per sentence length is called 'allnum'. We refer to the features that together model a lack of all five POS types as 'noall'. Verb agreements features are represented by 'verbagr'. For MT04, the best performing model is S-POS+noall, with a significant improvement at $p < 0.01$ over the DLM n-gram model of 0.13 corpus level BP. [4] For MT05, the best performing model is S-POS+vn+dn with a significant improvement of 0.18BP at $p < 0.01$. The S-POS+allnum model gives the same absolute BP improvement for MT05, but is insignificant. For MT06 we have a larger improvement of 0.41BP, again at $p < 0.01$, using S-POS+noall. The S-POS+vn+dn model is not the best performing model on MT04 or MT06, but consistently gives significant improvements.

Table 7 presents the individual n-gram precisions for our best syntactic models in comparison to the n-gram only DLM. There is a degrade in relative unigram precision on MT04, MT05 and MT06, but we see an increase in bi-gram, tri-gram and four-gram precisions, indicating our syntactic features resolve some of the word re-ordering problems.

To see how the S-POS features help, Table 8 presents the different POS sequences assigned by the three different syntactic tools to the translation: *he reiterated " full support of the islamic republic for islamic government interim " in afghanistan*. This sentence is chosen by the perceptron reranker using POS features from the CM2 parser. The bigram "government interim" is tagged as "NN NN" by CM2 and the CRF tagger. This feature has a positive weight, and therefore is not penalised. Only S-POS tags "interim" as an adjective. In the last row of Table 8 we show the translation chosen by the perceptron reranker using features from the S-POS tagger. This leads to an improvement of 17.5 sentence-level BLEU, and is an example of the significant gains we make using our S-POS tagger. It appears that 'better' taggers try to apply a more English like tag sequence to ill-formed sentences. In doing so, we potentially lose information that can lead to a better discriminative model.

5 Related Work

Perceptrons have been successfully applied to parse reranking (Collins and Duffy, 2002), document reranking for IR (Crammer and Singer, 2001; Elsas et al., 2008; Chen et al., 2009), ASR reranking (Roark et al., 2004b; Collins et al., 2005; Singh-Miller and Collins, 2007), and finally to SMT translation reranking, where significant improvements have been achieved for Chinese-English translation systems (Shen et al., 2004; Li and Khudanpur, 2008).

The work most closely related to ours is the discriminative syntactic LM proposed in (Collins et al., 2005). The work presented in this paper differs in two important aspects: first, we focus on the use of syntactic features in SMT. Second, we propose the use of a simple POS tagger, which speeds up the data

[4] Statistical significance is calculated using the paired bootstrap resampling method (Koehn, 2004).

System	POS Sequence	BLEU
CM2	PRP/he VBD/reiterated ”/” JJ/full NN/support IN/of DT/the NNP/islamic NNP/republic IN/for JJ/islamic **NN/government NN/interim** IN/in ”/” NNP/afghanistan ./.	50.30
CRF	PRP/he VBD/reiterated ”/” JJ/full NN/support IN/of DT/the JJ/islamic NN/republic IN/for JJ/islamic **NN/government NN/interim** ”/” IN/in JJ/afghanistan ./.	50.30
S-POS	PRP/he VBD/reiterated ”/” JJ/full NN/support IN/of DT/the NNP/islamic NNP/republic IN/for NNP/islamic **NN/government JJ/interim** ”/” IN/in NNP/afghanistan ./.	50.30
S-POS	PRP/he VBD/reiterated ”/” JJ/full NN/support IN/of DT/the NNP/islamic NNP/republic IN/for NNP/islamic **JJ/interim NN/government** IN/in NNP/afghanistan ./. ”/”	**67.64**

Table 8: POS assignments made by the three syntactic tools for the sentence *he reiterated ” full support of the islamic republic for islamic government interim ” in afghanistan*. We present sentence-level BLEU scores for both translations.

annotation phase considerably, and gives us significant improvements over the state-of-the-art.

Shen et al. (Shen et al., 2004) are the first to use a perceptron like algorithm in a small scale application of reranking SMT n-best lists. They used the algorithm to optimise weights for a small number of features (tens instead of millions). The use of perceptron type algorithms with millions of features for SMT has been explored by (Arun and Koehn, 2007). They examine the use of online algorithms for the discriminative training of a phrase based SMT system. In this paper we focus on the use of perceptrons for reranking using only target side syntactic information.

The first application of a large scale discriminative language model to SMT reranking is undertaken by Li and Khudanpur (2008). Using a standard n-gram feature set they outperformed the 1-best output of their baseline SMT system. They focus on the application of n-gram only models to SMT and the use of data filtering thresholds. We concentrate on the efficient syntactic annotation of large amounts of data and its successful exploitation.

A study in the use of a range of generative syntactic models was undertaken by Och et al. (2004), who did n-best reranking as we have done. Syntactic features were not successful. More recently, the use of Collins parser, similar to the parser used in this paper, during the hypothesis generation phrase was reported to be unsuccessful (Post and Gildea, 2008).

6 Conclusions

In this paper, we successfully applied a large scale, discriminative LM to SMT. We report experiments using a feature set only previously used in ASR. We have shown that use of features extracted from a simple POS tagger outperforms those from a state-of-the-art parser as well as a CRF POS tagger.

As future work, we believe there is potential for the use of partial and shallow parsers. Whilst providing less information than a full parser, they would generate deeper information than a POS tagger, and supply features for all sentences in the sets used.

Further, we believe there is potential for the use of ensemble techniques. If we train our perceptron models on different subsets of the data, we get large differences in BLEU results, indicating the potential for more improvements.

Finally, work needs to be done to answer the question of why it is a simple POS tagger outperformed state-of-the-art syntactic tools. A detailed examination, outside the scope of this paper, needs to be undertaken.

Acknowledgments

We would like to thank Valentin Jijkoun for helpful discussions, and the anonymous reviewers for their constructive comments. This work has been funded in part by the European Commission through the CoSyne project FP7-ICT-4-248531.

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)*.

Daniel M. Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of the second international conference on Human Language Technology Research*, pages 178–182.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing methods for language modelling. Technical Report TR-10-98, Harvard.

Xue-wen Chen, Haixun Wang, and Xiaotong Lin. 2009. Learning to rank with a novel kernel perceptron method. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 505–512.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 224–233, Morristown, NJ, USA. Association for Computational Linguistics.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 263–270.

Michael Collins, Brian Roark, and Murat Saraclar. 2005. Discriminative syntactic language modeling for speech recognition. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 507–514.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Koby Crammer and Yoram Singer. 2001. Pranking with ranking. In *Advances in Neural Information Processing Systems 14*, pages 641–647.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

Jonathan L. Elsas, Vitor R. Carvalho, and Jaime G. Carbonell. 2008. Fast learning of document ranking functions with the committee perceptron. In *Proceedings of the International Conference on Web search and Web Data Mining*, pages 55–64.

A. Emami, K. Papineni, and J. Sorensen. 2007. Large-scale distributed language modeling. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 37–40.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Stephen I. Gallant. 1999. Perceptron based learning algorithms. *IEEE Transactions on Neural Networks*, 1(2):179–191.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Shankar Kumar, William Byrne, and Speech Processing. 2004. Minimum bayes-risk decoding for statistical machine translation. In *Proceedings of Human Language Technologies: North American Chapter of the Association for Computational Linguistics*.

Zhifei Li and Sanjeev Khudanpur. 2008. Large-scale discriminative n-gram language models for statistical machine translation. In *In Proceedings of the The Eighth Conference of the Association for Machine Translationin the Americas (AMTA) 2008*.

Chin Y. Lin and Franz J. Och. 2004. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *COLING*, pages 501–507.

Mitchell Marcus, Grace Kim, Mary A. Marcinkiewicz, Robert Macintyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *ARPA Human Language Technology Workshop*, pages 114–119.

Franz J. Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447.

Franz J. Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of the 2004 Meeting of the North American chapter of the Association for Computational Linguistics*, pages 161–168.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 160–167.

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*. Association for Computational Linguistics.

Matt Post and Daniel Gildea. 2008. Parsers as language models for statistical machine translation. In *Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas*.

Brian Roark, Murat Saraclar, and Michael Collins. 2004a. Corrective language modeling for large vocabulary asr with the perceptron algorithm. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 749–752.

Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. 2004b. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 47–54.

Brian Roark, Murat Saraclar, and Michael Collins. 2007. Discriminative n-gram language modeling. *Computer Speech and Language*, 21(2):373–392.

Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Neurocomputing: foundations of research*, 65:6:386–408.

Ronald Rosenfeld. 1996. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language*, 10:187–228.

Ronald Rosenfeld. 2000. Two decades of statistical language modeling: Where do we go from here? In *Proceedings of the IEEE*, volume 88, pages 1270–1278.

Libin Shen, Anoop Sarkar, and Franz J. Och. 2004. Discriminative reranking for machine translation. In *Proceedings of HLT-NAACL*.

N. Singh-Miller and C. Collins. 2007. Trigger-based language modeling using a loss-sensitive perceptron algorithm. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 25–28.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP 2002)*.