**Final Report of Johns Hopkins 2003 Summer Workshop on**

# Syntax for Statistical Machine Translation

**Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, Dragomir Radev**

**Team website:**
`http://www.clsp.jhu.edu/ws03/groups/translate/`

**Revised Version: February 25, 2004**

**Abstract**

In recent evaluations of machine translation systems, statistical systems have outperformed classical approaches based on interpretation, transfer, and generation. Nonetheless, the output of statistical systems often contains obvious grammatical errors. This can be attributed to the fact that the syntactic well-formedness is only influenced by local n-gram language models and simple alignment models. We aim to integrate syntactic structure into statistical models to address this problem.

In the workshop we start with a very strong baseline – the alignment template statistical machine translation system that obtained the best results in the 2002 and 2003 DARPA MT evaluations. This model is based on a log-linear modeling framework, which allows for the easy integration of many different knowledge sources (i.e. feature functions) into an overall model and to train the feature function combination weights discriminatively. During the workshop, we incrementally add new features representing syntactic knowledge that deal with specific problems of the underlying baseline. We want to investigate a broad range of possible feature functions, from very simple binary features to sophisticated tree-to-tree translation models. Simple feature functions test if a certain constituent occurs in the source and the target language parse tree. More sophisticated features are derived from an alignment model where whole sub-trees in source and target can be aligned node by node. We also plan to investigate features based on projection of parse trees from one language onto strings of another, a useful technique when parses are available for only one of the two languages. We extend previous tree-based alignment models by allowing partial tree alignments when the two syntactic structures are not isomorphic.

We work with the Chinese-English data from the recent evaluations, as large amounts of sentence-aligned training corpora, as well as multiple reference translations are available. This will also allow to compare results with the various systems participating in the evaluations. In addition, an annotated Chinese-English parallel tree-bank is available. We evaluate the improvement of our system using the BLEU metric. Using the additional feature functions developed during the workshop the BLEU score improved from 31.6% for the baseline MT system to 33.2% using rescoring of a 1000-best list.

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Machine translation is a hard problem because natural languages are highly complex, many words have various meanings and different possible translations, sentences might have various readings and the relationships between linguistic entities are often vague. In addition, it is sometimes necessary to take world knowledge into account. The amount of relevant dependencies is much too large and too complex to take them all into account in a machine translation system. Given these boundary conditions, the machine translation system has to make decisions (produce translations) given incomplete knowledge. In such a case, a principled approach to solve that problem is to use the concepts of statistical decision theory to try to make optimal decisions given incomplete knowledge. This is the goal of statistical machine translation.

The use of statistical techniques in machine translation has led to dramatic improvements in the quality of research systems in the recent years. For example the Verbmobil evaluations (Wahlster, 2000) or the NIST/TIDES MT evaluations 2001 through 2003[1] statistical approaches obtain the best results. In addition, the field of statistical machine translation is rapidly progressing and the quality of systems is getting better and better.

An often noted problem of state-of-the-art statistical MT systems is that the produced translations often contain obvious 'stupid' grammatical errors. In the following we provide various examples of actual mistakes occurring in our Chinese-English baseline translation system:

[1] `http://www.nist.gov/speech/tests/mt/`

```
(*) The resolution urged the to the ceasefire and
    demands ...
    The resolution urges Israel and the Palestinians
    to cease fire and demands ...

(*) Indonesia that oppose the presence of foreign
    troops.
    Indonesia reiterated its opposition to foreign
    military presence.

(*) in brief ) ( South Korea will be on high-ranking
    official to visit North Korea
    (News in Brief) South Korean high-ranking officials
    to visit North Korea in April

(*) Japan to freeze Russia to provide humanitarian aid
    Japan to freeze humanitarian assistance to Russia

(*) ... if the west further sanctions against zimbabwe ...
    ... if western countries impose further sanctions
    against zimbabwe ...

(*) ... he is fully able to activate team.
    ... he is fully able to activate the team.

(*) ... , particularly those who cheat the
    audience the players.
    ... , particularly those players who cheat
    the audience.

(*) ... the trial of the outcome ...
    ... the outcome of the trial ...
```

Frequent problems are missing content words, wrong word order and wrong choice of function words.

In existing statistical MT systems and also in our baseline MT system the syntactic well-formedness of the output sentence is only influenced by local n-gram

language models and simple alignment models. All these models are learned by purely data-driven methods. A potential remedy for the poor syntactic quality lies now in using to incorporate syntactic properties into the statistical translation models, i.e. to develop **explicit syntactic models** in contrast to the **implicit syntactic models** that are used by the baseline system. In the workshop we investigate both, implicit and explicit syntactic models to improve the performance of the baseline MT system.

## 1.2 Log-linear Models for Statistical Machine Translation

The goal is the translation of a text given in some source language into a target language. We are given a source ('Chinese') sentence $\mathbf{f} = f_1^J = f_1, \ldots, f_j, \ldots, f_J$, which is to be translated into a target ('English') sentence $\mathbf{e} = e_1^I = e_1, \ldots, e_i, \ldots, e_I$. Among all possible target sentences, we will choose the sentence with the highest probability:[2]

$$\hat{e}_1^I = \underset{e_1^I}{\operatorname{argmax}} \{Pr(e_1^I|f_1^J)\} \tag{1.1}$$

The argmax operation denotes the search problem, i.e. the generation of the output sentence in the target language.

As alternative to the often used source–channel approach (Brown et al., 1993), we directly model the posterior probability $Pr(e_1^I|f_1^J)$ (Och and Ney, 2002). An especially well-founded framework for doing this is the maximum entropy framework (Berger, Della Pietra, and Della Pietra, 1996). In this framework, we have a set of $M$ feature functions $h_m(e_1^I, f_1^J), m = 1, \ldots, M$. For each feature function, there exists a model parameter $\lambda_m, m = 1, \ldots, M$. The direct translation probability is given by:

$$Pr(e_1^I|f_1^J) = p_{\lambda_1^M}(e_1^I|f_1^J) \tag{1.2}$$

$$= \frac{\exp[\sum_{m=1}^M \lambda_m h_m(e_1^I, f_1^J)]}{\sum_{e'^I_1} \exp[\sum_{m=1}^M \lambda_m h_m(e'^I_1, f_1^J)]} \tag{1.3}$$

[2]The notational convention will be as follows. We use the symbol $Pr(\cdot)$ to denote general probability distributions with (nearly) no specific assumptions. In contrast, for model-based probability distributions, we use the generic symbol $p(\cdot)$.

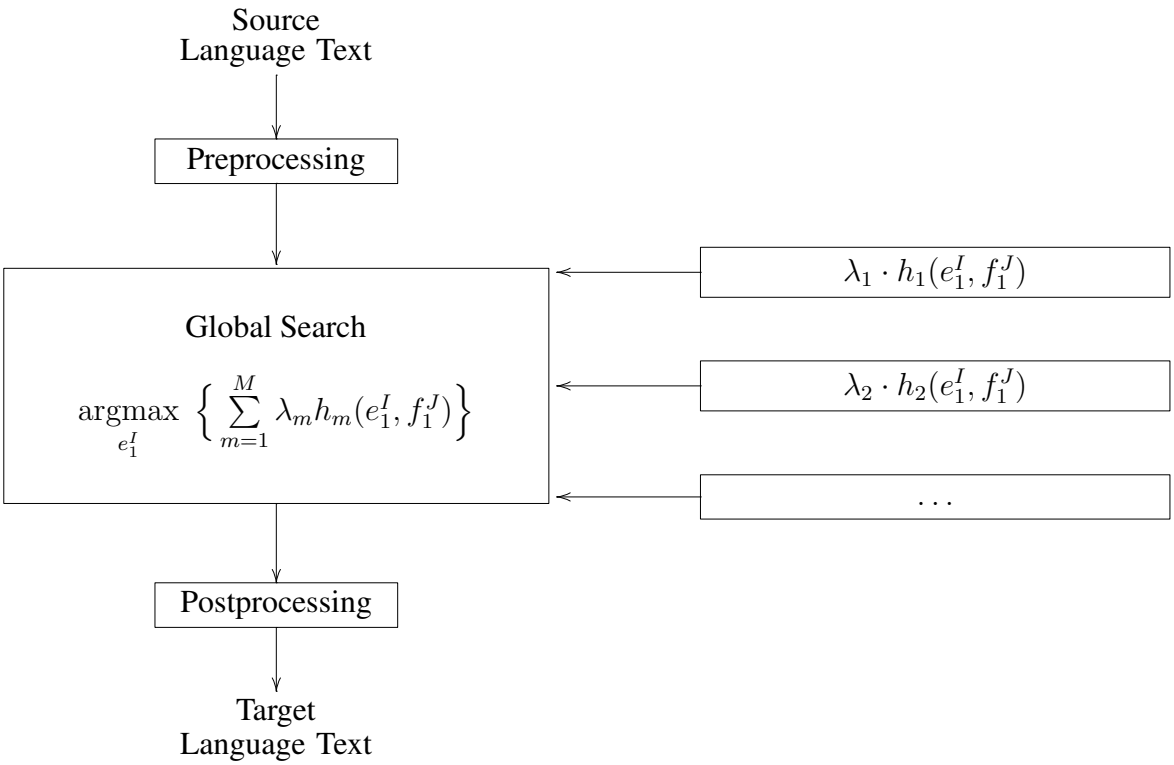



Figure 1.1: Architecture of the translation approach based on a log-linear modeling approach.

This approach has been suggested by (Papineni, Roukos, and Ward, 1997; Papineni, Roukos, and Ward, 1998) for a natural language understanding task.

We obtain the following decision rule:

$$\begin{aligned} \hat{e}_1^I &= \underset{e_1^I}{\text{argmax}} \left\{ Pr(e_1^I | f_1^J) \right\} \\ &= \underset{e_1^I}{\text{argmax}} \left\{ \sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J) \right\} \end{aligned}$$

Hence, the time-consuming renormalization in Eq. 1.3 is not needed in search. The overall architecture of the log-linear modeling approach is summarized in Figure 1.1.

A standard criterion on a parallel training corpus consisting of $S$ sentence pairs $\{(\mathbf{f}_s, \mathbf{e}_s) : s = 1, \ldots, S\}$ for log-linear models is the maximum class posterior

probability criterion, which can be derived from the maximum entropy principle:

$$\hat{\lambda}_1^M = \underset{\lambda_1^M}{\operatorname{argmax}} \left\{ \sum_{s=1}^{S} \log p_{\lambda_1^M}(\mathbf{e}_s|\mathbf{f}_s) \right\} \tag{1.4}$$

This corresponds to maximizing the equivocation or maximizing the likelihood of the direct translation model. This direct optimization of the posterior probability in Bayes decision rule is referred to as discriminative training (Ney, 1995) because we directly take into account the overlap in the probability distributions. The optimization problem under this criterion has very nice properties: there is one unique global optimum, and there are algorithms (e.g. gradient descent) that are guaranteed to converge to the global optimum. Yet, the ultimate goal is to obtain good translation quality on unseen test data.

An alternative training criterion therefore directly optimizes translation quality as measured by an automatic evaluation criterion, which will be described in Section 1.6.

Typically, the translation probability $Pr(e_1^I|f_1^J)$ is decomposed via additional hidden variables. To include these dependencies in our log-linear model, we extend the feature functions to include the dependence on the additional hidden variable. Using for example the alignment $a_1^J$ as hidden variable, we obtain $M$ feature functions of the form $h_m(e_1^I, f_1^J, a_1^J), m = 1, \ldots, M$ and the following model:

$$Pr(e_1^I, a_1^J|f_1^J) = \frac{\exp\left(\sum_{m=1}^{M} \lambda_m h_m(e_1^I, f_1^J, a_1^J)\right)}{\sum_{e'^I_1, a'^J_1} \exp\left(\sum_{m=1}^{M} \lambda_m h_m(e'^I_1, f_1^J, a'^J_1)\right)}$$

Obviously, we can perform the same step for translation models with an even richer set of hidden variables than only the alignment $a_1^J$.

## 1.3 Baseline MT System: Alignment Templates

Our baseline machine translation system is the alignment template system as implemented at ISI/USC (originally implemented at RWTH Aachen (Och, 2002)). In the following, we give a short description of this baseline model. More details can be found in (Och, Tillmann, and Ney, 1999; Och and Ney, 2004).

In the alignment template translation model, a sentence is translated by segmenting the input sentence into phrases, translating these phrases and reordering the translations in the target language. To describe our translation model based on

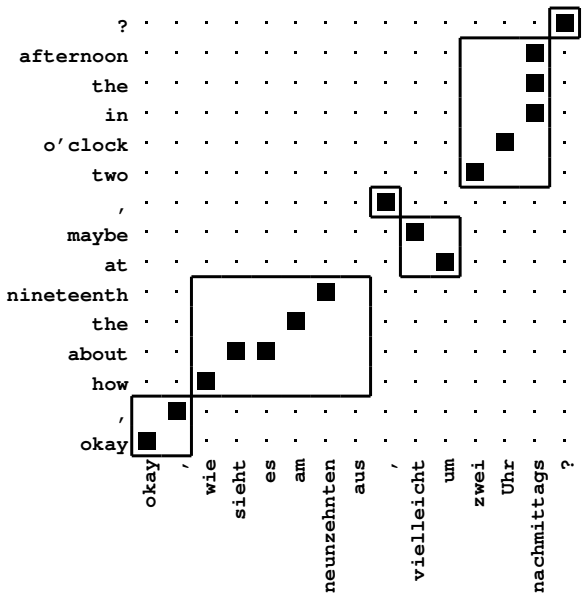



Figure 1.2: Example segmentation of German sentence and its English translation into alignment templates.

the alignment templates described in the previous section in a formal way, we first decompose both the source sentence $f_1^J$ and the target sentence $e_1^I$ into a sequence of phrases ($k = 1, \ldots, K$):

$$f_1^J = \tilde{f}_1^K, \qquad \tilde{f}_k = f_{j_{k-1}+1}, \ldots, f_{j_k} \tag{1.5}$$

$$e_1^I = \tilde{e}_1^K, \qquad \tilde{e}_k = e_{i_{k-1}+1}, \ldots, e_{i_k} \tag{1.6}$$

Note that there are a large number of possible segmentations of a sentence pair into $K$ phrase pairs. In the following, we will describe the model for a certain segmentation. Eventually, the specific segmentation is not known when new text is translated. Hence, as part of the overall search process, we will also search for the optimal segmentation.

To allow possible reordering of phrases, we introduce an alignment on the phrase level $\pi_1^K$ between the source phrases $\tilde{e}_1^K$ and the target phrases $\tilde{f}_1^K$. Hence, $\pi_1^K$ is a permutation of the phrase positions $1, \ldots, K$ and describes that the phrase $\tilde{e}_k$ and $\tilde{f}_{\pi_k}$ are translations of each other. We assume that for the translation between these phrases a specific alignment template $z_k$ is used:

$$\tilde{e}_k \stackrel{z_k}{\longleftrightarrow} \tilde{f}_{\pi_k}$$

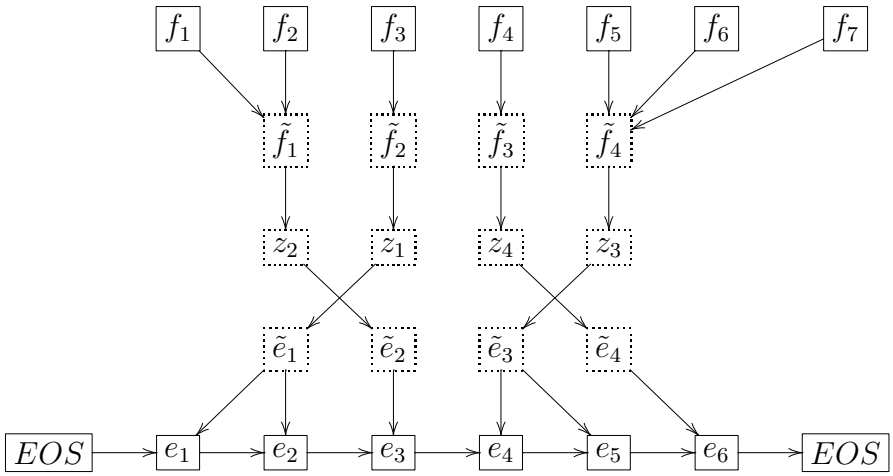



Figure 1.3: Dependencies in the alignment template model.

Hence, our model has the following hidden variables:

$$\pi_1^K, z_1^K$$

Figure 1.2 gives an example of the word alignment and phrase alignment of a German–English sentence pair.

We describe our model using a log-linear modeling approach. Hence, all knowledge sources are described as feature functions which include the given source language string $f_1^J$, the target language string $e_1^I$ and the above stated hidden variables. Hence, we have the following functional form of all feature functions:

$$h(e_1^I, f_1^J, \pi_1^K, z_1^K)$$

Figure 1.3 gives an overview of the decisions taken in the alignment template model. First, the source sentence words $f_1^J$ are grouped to phrases $\tilde{f}_1^K$. For each phrase $\tilde{f}$ an alignment template $z$ is chosen and the sequence of chosen alignment templates is reordered (according to $\pi_1^K$). Then, every phrase $\tilde{f}$ produces its translation $\tilde{e}$ (using the corresponding alignment template $z$). Finally, the sequence of phrases $\tilde{e}_1^K$ constitutes the sequence of words $e_1^I$.

## Feature Functions

### Alignment Template Selection

To score the use of an alignment template, we use the probability $p(z|\tilde{f})$ described in which is just estimated by relative frequency. We establish a corresponding feature function by multiplying the probability of all used alignment templates and taking the logarithm:

$$h_{\text{AT}}(e_1^I, f_1^J, \pi_1^K, z_1^K) = \log \prod_{k=1}^{K} p(z_k | f_{j_{\pi_k-1}+1}^{j_{\pi_k}}) \tag{1.7}$$

Here, $j_{\pi_k-1}+1$ is the position of the first word of alignment template $z_k$ in the source language sentence and $j_{\pi_k}$ is the position of the last word of that alignment template.

Note that this feature function requires that a translation of a new sentence be composed of a set of alignment templates that covers both the source sentence and the produced translation. There is no notion of 'empty phrase' that corresponds to the 'empty word' in the word-based statistical alignment models. The alignment on the phrase level is actually a permutation and no insertions or deletions are allowed.

### Word Selection

For scoring the use of target language words, we use a lexicon probability $p(e|f)$, which is estimated using relative frequencies. The target word $e$ depends on the aligned source words. If we denote the resulting word alignment matrix by $A := A_{\pi_1^K, z_1^K}$ and the predicted word class for word $e_i$ by the symbol $E_i$, then the feature function $h_{\text{WRD}}$ is defined as follows:

$$h_{\text{WRD}}(e_1^I, f_1^J, \pi_1^K, z_1^K) = \log \prod_{i=1}^{I} p(e_i | \{f_j | (i,j) \in A\}, E_i) \tag{1.8}$$

For $p(e_i|\{f_j|(i,j) \in A\})$ we use a uniform mixture of a single-word model $p(e|f)$ which is constrained to predict only words which are in the predicted word class $E_i$:

$$p(e_i | \{f_j | (i,j) \in A\}, E_i) = \frac{\sum_{\{j|(i,j)\in A\}} p(e_i|f_j)}{|\{j|(i,j) \in A\}|} \cdot \delta(C(e_i), E_i)$$

A disadvantage of this model is that the word order is ignored in the translation model. The translations *'the day after tomorrow'* or *'after the day tomorrow'* for the German word *'übermorgen'* receive an identical contribution. Yet, the first one should obtain a significantly higher probability. Hence, we also include a dependence on the word positions in the lexicon model $p(e|f,i,j)$:

$$p(e_i|f_j, \sum_{i'=1}^{i-1}[(i',j) \in A], \sum_{j'=1}^{j-1}[(i,j') \in A]) \tag{1.9}$$

This model distinguishes the positions within a phrasal translation. The number of parameters of $p(e|f,i,j)$ is significantly higher than $p(e|f)$ alone. Hence, there is a data estimation problem especially for words that rarely occur. Therefore, we linearly interpolate the models $p(e|f)$ and $p(e|f,i,j)$.

## Phrase Alignment

The phrase alignment feature simply takes into account that very often a monotone alignment is a correct alignment. Hence, the feature function $h_{\mathrm{AL}}$ measures the 'amount of non-monotonicity' by summing over the distance (in the source language) of alignment templates which are consecutive in the target language:

$$h_{\mathrm{AL}}(e_1^I, f_1^J, \pi_1^K, z_1^K) = \sum_{k=1}^{K+1} |j_{\pi_k} - j_{\pi_{k-1}+1}| \tag{1.10}$$

Here, $j_{\pi_{K+1}}$ is defined to equal $J+1$. The above stated sum includes $k = K+1$ to include the distance from the end position of the last phrase to the end of sentence.

The sequence of $K = 6$ alignment templates in Figure 1.2 corresponds to the following sum of seven jump distances: $0 + 0 + 1 + 3 + 2 + 0 + 0 = 6$.

## Language Model Features

As default language model feature, we use a standard backing off word-based trigram language model (Ney, Generet, and Wessel, 1995):

$$h_{\mathrm{LM}}(e_1^I, f_1^J, \pi_1^K, z_1^K) = \log \prod_{i=1}^{I+1} p(e_i|e_{i-2}, e_{i-1}) \tag{1.11}$$

The baseline system actually includes four different language model features which are trained on four different training corpora: the news part of the bilingual training data, a large Xinhua news corpus, a large AFP news corpus and a language model trained from Chinese news texts downloaded from the web.

## Word/Phrase Penalty

To improve the scoring for different target sentence lengths, we use as feature also the number of produced target language words (i.e. the length of the produced target language sentence):

$$h_{\mathrm{WP}}(e_1^I, f_1^J, \pi_1^K, z_1^K) = I \tag{1.12}$$

Without this feature, we typically observe that the produced sentences tend to be too short.

In addition, there is a feature function which counts the number of produced phrases:

$$h_{\mathrm{AP}}(e_1^I, f_1^J, \pi_1^K, z_1^K) = K \tag{1.13}$$

This feature function allows to give preference to short or long phrases should be preferred.

## Phrases from Conventional Lexicon

The baseline alignment template system makes use of the Chinese–English lexicon provided by LDC. Each lexicon entry is a potential phrase translation pair in the alignment template system. To score the use of these lexicon entries (which have no normal translation probability), there is a feature function that counts the number of times such a lexicon entry is used.

## Additional Features

A major advantage of the log-linear modeling approach used is that we can add numerous additional features that deal with specific problems of the baseline statistical MT system. Here, we will restrict ourselves to the described set of features. Yet, we could use grammatical features that relate certain grammatical dependencies of source and target language. For example, using a function $k(\cdot)$ that counts

how many arguments the main verb of a sentence has in source or the target sentence, we can define the following feature, which fires if the verb in each of the two sentences has the same number of arguments:

$$h(f_1^J, e_1^I, \pi_1^K, z_1^K) = \delta(k(f_1^J), k(e_1^I)) \tag{1.14}$$

In the same way, we can introduce semantic features or pragmatic features such as the dialog act classification.

## 1.4 Training Environment and Test Corpora

We work with the Chinese-English data from the recent evaluations, as large amounts of sentence-aligned training corpora, as well as multiple reference translations are available. This data set is a standard data set which makes it possible to compare results with the various systems participating in the evaluations and many published results. In addition, an annotated Chinese-English parallel tree-bank is available (more details in Section 1.7).

For the baseline MT system, we distinguish the following three different sentence- or chunk-aligned parallel training corpora:

- **training corpus (train)**: This is the basic training corpus used to train the alignment template translation model (word lexicon and phrase lexicon). This corpus consists of about 170M English words. Large parts of this corpus are aligned on a sub-sentence level to avoid the existence of very long sentences which would be filtered out in the training process to allow a manageable word alignment training.

- **development corpus (Dev)**: This is the training corpus used in discriminative training of the model-parameters of the log-linear translation model (Section 1.6). In most experiments described in this report this corpus consists of 993 sentences (about 25K words) in both languages. During the workshop was also prepared a larger set of 5765 sentences (about 175K words) to be used for post-workshop experiments.

- **test corpus (test)**: This is the test corpus used to assess the quality of the newly developed feature functions. It consists of 878 sentences (about 25K words).

In addition, there exists a prepared held-out test corpus (**blind-test**) that can be used to make experiments using completely unseen data.

## 1.5 Reranking, $n$-best lists and oracles

For each sentence in the development, test and the blind test corpus a set of 16384 different alternative translations has been produced using the baseline system. For extracting the $n$-best candidate translations, an A* search (Ueffing, Och, and Ney, 2002). These $n$-best candidate translations are the basis for discriminative training of the model parameters and for re-ranking.

The decision to use $n$-best reranking instead of implementing new search algorithms is because handling $n$-best lists is much easier. The development of efficient search algorithms for long-range dependencies is very complicated and a research topic in itself. During the workshop the major goal is to quickly try out a lot of new dependencies which would not be possible if for each new dependency the search algorithm has to be changed.

On the other hand, it has to be emphasized that the use of $n$-best list rescoring severely limits the possibility of improvements to what is available in the $n$-best list. Hence if our $n$-best lists do not include good translations, the rescoring cannot produce good translations. Hence, it is important to analyze the quality of the $n$-best lists by analyzing how good we could get if we had a very good reranking algorithm. We do that by computing the **oracle translations** which is the set of translations that yields the best BLEU score for a given set of candidate translations.[3] This approach to compute the oracle performance of an $n$-best lists or a lattice representation of alternatives is common in the speech community.

A straightforward approach to compute the quality of an oracle translation would consist of the following two steps:

1. take sentences from the $n$-best list which give the highest BLEU score compared **to the set of 4 reference translations**

2. compute BLEU score of oracle sentences using **the same set of reference translations**

If we do that it turns out that with a 1000-best list we obtain oracle translation that outperform the BLEU score of good human translations by achieving a 113% relative human BLEU score on the test data (see below) while the first best translation just obtains a 78.9% relative human BLEU score. Problem of this approach

[3] Note that due to the 'holistic' nature of the BLEU score it is not trivial to compute the optimal set of oracle translations. We use a greedy search algorithm for the oracle translations that might find only a local optimum. Empirically, we do not observe a dependence on the starting point, hence we believe that this does not pose a significant problem.

is that the same references are used to compute the oracle and to score the oracle. It is like asking humans to produce a new translation AND showing them the references as guideline. Obviously this way to compute the oracle leads to inflated BLEU scores.

Hence, a better approach is to use a different reference sentence to select the oracle and to compute the BLEU score using different references. To avoid the need for additional references, we do this using a round-robin approach: First reference number 1 is used to select the oracle and the references 2-4 are used to score the resulting sentences. Then reference 2 is used to select the oracle and references 1, 3 and 4 are used to compute the BLEU score and correspondingly for reference 3 and 4. The resulting BLEU scores are then averaged. The average human BLEU scores are computed in the same manner and are therefore directly comparable to the resulting oracle BLEU score.

The results of this round-robin oracle make much more sense. The corresponding relative human BLEU score on a 1000-best list is 89.3% which indicates that we can actually improve our system only from 78.9% only by about 10% absolute and not by about 35% absolute as is indicated by the above mentioned naive oracle computation.

Note that by construction the round-robin oracle won't (drastically) go over 100% relative human BLEU score.

If we use all four references our baseline BLEU score is 31.6%. Using the above mentioned estimates, we could estimate that our 1000-best list on the test corpus allows us to increase the BLEU score by a factor of $89.3/78.9 = 1.131$. If we multiply our baseline result from 31.6% BLEU score (using four references) with 1.131 we obtain 35.7% BLEU score, which can be seen as a more realistic upper limit of what we can achieve using the 1000-best list.[4]

Table 1.1 shows the resulting oracle BLEU scores for $n$-best list sizes ranging from 1 to 16384. There are shown the BLEU scores computed with three references averaged over the four different sets of three references (avBLEUr3n4) and there are shown the BLEU scores of the optimal oracle computed using all four references (BLEUr4n4). We observe that doubling the size of the $n$-best list yields on average 1.1% improvement of the optimal oracle BLEU score and about 0.33% for the round-robin oracle. For the round-robin oracle the improvements seem to get smaller as the $n$-best list size increases. The gain from going from 512 to 1024 alternatives is just 0.14% while the improvement going from 1 to 2

[4]Presumably using more references to pick the oracle should also give a better oracle, hence the new estimate of the upper bound is probably slightly pessimistic.

Table 1.1: Oracle BLEU scores for different sizes of the $n$-best list. `rr-oracle` refers to the round-robin oracle and `opt-oracle` refers to the optimal oracle with respect to the four reference translations. Note that the provided BLEU scores are computed with respect to three reference reference translations (r3) and are computed by averaging over the four different choices of holding out one reference. The corresponding relative human BLEU score is 35.7%.

| | avBLEUr3n4[%] | | BLEUr4n4 |
|---|---|---|---|
| $n$ | `rr-oracle` | `opt-oracle` | `opt-oracle` |
| human | 35.76 | | - |
| 1 | 28.31 | 28.31 | 31.60 |
| 2 | 28.72 | 29.47 | 32.98 |
| 4 | 29.11 | 30.76 | 34.50 |
| 8 | 29.48 | 31.97 | 35.89 |
| 16 | 29.87 | 33.19 | 37.26 |
| 32 | 30.33 | 34.35 | 37.26 |
| 64 | 30.61 | 35.60 | 38.65 |
| 128 | 30.96 | 36.86 | 41.54 |
| 256 | 31.29 | 37.97 | 42.84 |
| 512 | 31.52 | 38.96 | 44.01 |
| 1024 | 31.66 | 40.00 | 45.25 |
| 2048 | 31.89 | 40.94 | 46.35 |
| 4096 | 31.98 | 41.77 | 47.32 |
| 8192 | 32.16 | 42.57 | 48.26 |

alternative translations is 0.41%. The theoretical upper bound of the oracle on the 1024 list is 45.25% BLEU score.

## 1.6 Maximum BLEU training

Many tasks in natural language processing have evaluation criteria that go beyond simply counting the number of wrong decisions the system makes. Some often used criteria are, for example, F-Measure for parsing, mean average precision for ranked retrieval, and BLEU or multi-reference word error rate for statistical machine translation. The use of statistical techniques in natural language processing

often starts out with the simplifying (often implicit) assumption that the final scoring is based on simply counting the number of wrong decisions, for instance, the number of sentences incorrectly translated in machine translation. Hence, there is a mismatch between the basic assumptions of the used statistical approach and the final evaluation criterion used to measure success in a task.

Ideally, we would like to train our model parameters such that the end-to-end performance in some application is optimal. In this paper, we investigate methods to efficiently optimize model parameters with respect to machine translation quality as measured by automatic evaluation criteria such as word error rate and BLEU.

In the following, we assume that we can measure the number of errors in sentence $\mathbf{e}$ by comparing it with a reference sentence $\mathbf{r}$ using a function $\mathrm{E}(\mathbf{r}, \mathbf{e})$. However, the following exposition can be easily adapted to accuracy metrics and to metrics that make use of multiple references.

We assume that the number of errors for a set of sentences $\mathbf{e}_1^S$ is obtained by summing the errors for the individual sentences: $E(\mathbf{r}_1^S, \mathbf{e}_1^S) = \sum_{s=1}^{S} E(\mathbf{r}_s, \mathbf{e}_s)$.

Our goal is to obtain a minimal error count on a representative corpus $\mathbf{f}_1^S$ with given reference translations $\hat{\mathbf{e}}_1^S$ and a set of $K$ different candidate translations $\mathbf{C}_s = \{\mathbf{e}_{s,1}, \ldots, \mathbf{e}_{s,K}\}$ for each input sentence $\mathbf{f}_s$.

$$\begin{aligned} \hat{\lambda}_1^M &= \operatorname*{argmin}_{\lambda_1^M} \left\{ \sum_{s=1}^{S} E(\mathbf{r}_s, \hat{\mathbf{e}}(\mathbf{f}_s; \lambda_1^M)) \right\} \\ &= \operatorname*{argmin}_{\lambda_1^M} \left\{ \sum_{s=1}^{S} \sum_{k=1}^{K} E(\mathbf{r}_s, \mathbf{e}_{s,k}) \delta(\hat{\mathbf{e}}(\mathbf{f}_s; \lambda_1^M), \mathbf{e}_{s,k}) \right\} \end{aligned} \tag{1.15}$$

with

$$\hat{\mathbf{e}}(\mathbf{f}_s; \lambda_1^M) = \operatorname*{argmax}_{\mathbf{e} \in \mathbf{C}_s} \left\{ \sum_{m=1}^{M} \lambda_m h_m(\mathbf{e}|\mathbf{f}_s) \right\} \tag{1.16}$$

It is straightforward to refine this algorithm to also handle the BLEU score instead of sentence-level error counts by accumulating the relevant statistics for computing these scores (n-gram precision, translation length and reference length)
.

A standard algorithm for the optimization of the unsmoothed error count (Eq. 1.15) is Powells algorithm combined with a grid-based line optimization method (Press et al., 2002). We start at a random point in the $K$-dimensional parameter space and try to find a better scoring point in the parameter space by

making a one-dimensional line minimization along the directions given by optimizing one parameter while keeping all other parameters fixed. To avoid finding a poor local optimum, we start from different initial parameter values. A major problem with the standard approach is the fact that grid-based line optimization is hard to adjust such that both good performance and efficient search are guaranteed. If a fine-grained grid is used then the algorithm is slow. If a large grid is used then the optimal solution might be missed. In (Och, 2003) is described an efficient algorithm to find the optimal solution for the line optimization problem and presents the results of that approach in statistical machine translation applied to various evaluation metrics.

## 1.7 Syntactic Framework

As a precursor to developing the various syntactic features described in this report, the syntactic representations on which they are based needed to be computed. This involved part-of-speech tagging, chunking, and parsing both the Chinese and English side of our training, development, and test sets. This section describes the representations used, how they were computed, and the particular issues involved with parsing and tagging automatically generated machine translation output.

### 1.7.1 Segmentation and Part-of-Speech Tagging

For segmenting Chinese characters into words, the standard tools distributed by the Linguistic Data Consortium (LDC) were used. The English part-of-speech tagger was that of Ratnaparkhi (1996); for Chinese we used the part-of-speech tagger trained by Nianwen Xue using Ratnaparkhi's maximum modeling software. These taggers were trained on data from the Penn Treebank (Marcus, Santorini, and Marcinkiewicz, 1993) and Penn Chinese Treebank (Xue, Chiou, and Palmer, 2002); for complete descriptions of the set of tags used see Santorini (1990) and Fei Xia (2000). Example sentences and tags are shown below:

> Fourteen_CD Chinese_JJ open_JJ border_NN cities_NNS make_VBP significant_JJ achievements_NNS in_IN economic_JJ construction_NN
>
> Zhongguo_NR shisi_CD ge_M bianjing_NN kaifang_NN chengshi_NN jingji_NN jianshe_NN chengjiu_NN xianzhu_VV

The English tagset makes certain distinctions that are not relevant to Chinese, such singular vs. plural nouns (NN vs NNS) where Chinese just has one noun tag NN. Similarly, the English tagset marks present (VBP) and past tense (VBD) verbs, as well as present participles, past participles, etc, where Chinese just has one verb tag VV. Chinese has a number of tags with no English equivalent, including the measure word tag M.

### 1.7.2 Parsing

The English sentences were parsed using the parser of Collins (1999), the Chinese parser was provided to us by Dan Bikel (Bikel and Chiang, 2000; Bikel, 2002). Again, the parsers are trained on the Penn Treebank and Penn Chinese Treebank; sample parse trees are shown in Figures 1.4 and 1.5. For details of the treebank annotation, see Bies et al. (1995) and Nianwen Xue and Fei Xia (2000). The formats provide a similar skeletal syntactic tree representation. The empty constituent (trace) and constituent co-indexation information is not used or returned by our parsers, and will not be referenced by any of our syntactic features in this report. This is also true of the function tag information indicating, for example, temporals (TMP) and locatives (LOC).

The Chinese and English treebanks differ in certain respects. While English has S (sentence) nodes, Chinese has IP (inflectional phrase). A more substantive difference is that the notoriously flat English noun phrases tend to be given more structure in the Chinese treeabnk, as seen in the noun phrase "fourteen Chinese open border cites" in our example sentence. The Chinese treebank tends to adhere to the X-bar syntactic generalization, and consistently annotates maximal projections such as ADJP where they may be missing for single word phrases in English.

### 1.7.3 Chunks

Chunks provide a shallow level of syntax generally corresponding to a low level of non-recursive constituents in the treebank trees. We used the fnTBL chunker of Ngai and Florian (2001), trained on an automatic conversion of the English and Chinese treebanks to a chunk-level repesentation (Tjong Kim Sang and Buchholz, 2000). Examples of chunked sentences follow:

> [$_{\mathsf{NP}}$ Fourteen Chinese open border cities ] [$_{\mathsf{VP}}$ make ] [$_{\mathsf{NP}}$ significant achievements ] [$_{\mathsf{PP}}$ in ] [$_{\mathsf{NP}}$ economic construction ]

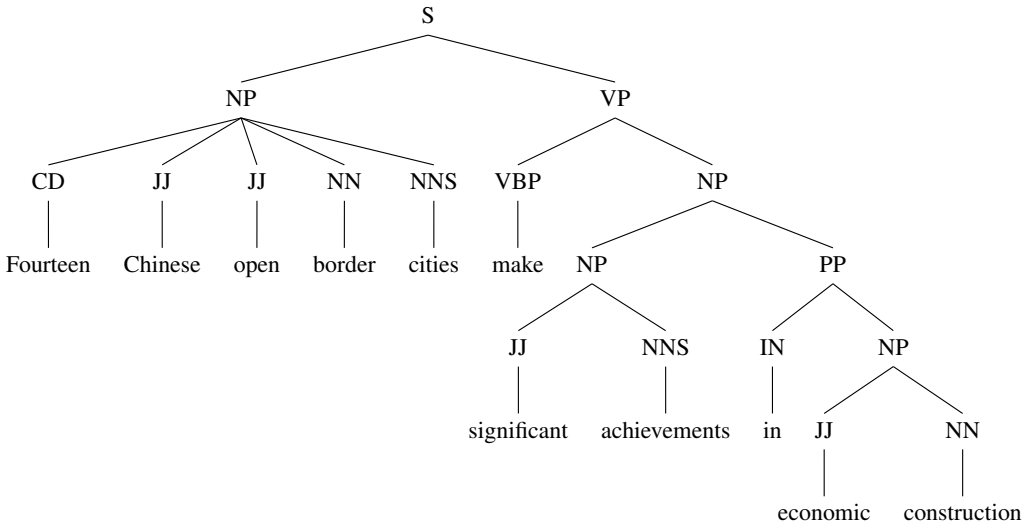



Figure 1.4: An English Parse Tree

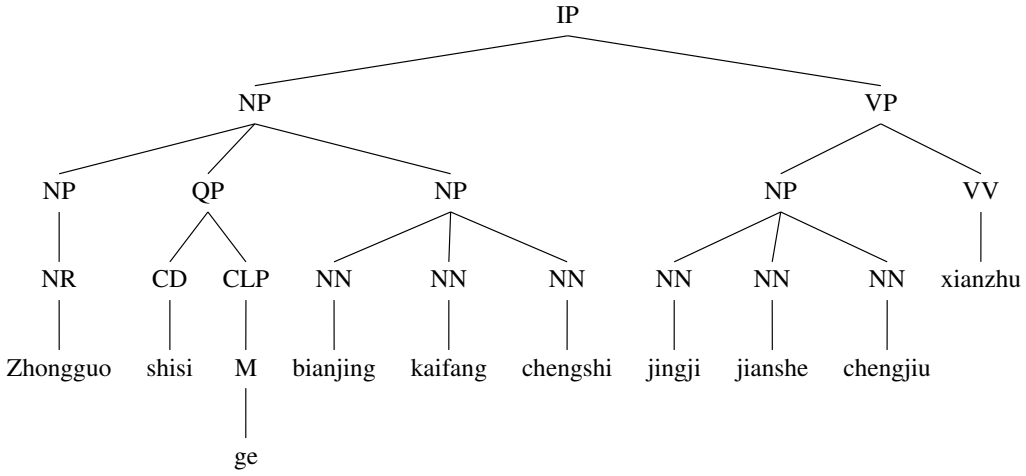



Figure 1.5: A Chinese Parse Tree

[NP Zhongguo ] [QP shisi ge ] [NP bianjing kaifang ] [NN chengshi ]
[NP jingji jianshe chengjiu ] [VP xianzhu ]

We hoped that chunks would represent a more robust representation than complete parse trees.

The following data processing was undertaken before and during the first weeks of the workshop:

- Training data
  - English 1M sents (chunked all, parsed all)
  - Chinese 1M sents (chunked all, parsed 100K sents)
  - English/Chinese FBIS parsed: 70K sents
- $n$-best lists
  - English 5000 sents, 1000 nbest (tagging, chunking, parsing)
  - Chinese 5000 sents (segmentation, tagging, chunking, parsing)

### 1.7.4 Case Issues

Our various tools including the part-of-speech taggers and parsers are trained on and perform best on mixed case input. However the baseline MT system produces lower-case only text.

To address this issue, we wrote a "true-caser" to guess the correct upper/lower case information of all-lower-case text. The approach was a Hidden Markov Model, with the case being the hidden state, implemented using the SRI language modeling toolkit. Measuring performance on recasing lower-cased text gave a result of 3.36% word error rate. The output of this system on the $n$-best was used as the input to the English POS taggers and parsers.

### 1.7.5 Tokenization Issues

The English tokenization used in the treebank and expected as input by our taggers and parser does not always match the internal tokenization of the baseline MT system. For example hyphen are split out as separate token for MT (*high* @-@ *tech*) but not for parsing (*high-tech*). Similarly *and* / *or* is separated for MT but not for parsing (*and\/or*).

A retokenizer was applied to the MT output before parsing. This meant that the word-level alignments from the MT system did not match the parse tokenization. This problem was solved by writing a minimum edit-distance program to align MT with parsed text using dynammic programming. The alignment produced by the MT system was composed with the MT-to-parser token alignment to produce word-level alignments that could be used to reference token in the parse trees.

Furthermore some word level alignment information was missing from $n$-best lists, due to phrases such as number and dates that had been generated by a separate rule-based translation module. Alignments for words in this phrases were added using separate alignment tool trained based on IBM Model 1.

### 1.7.6 Processing Noisy Data

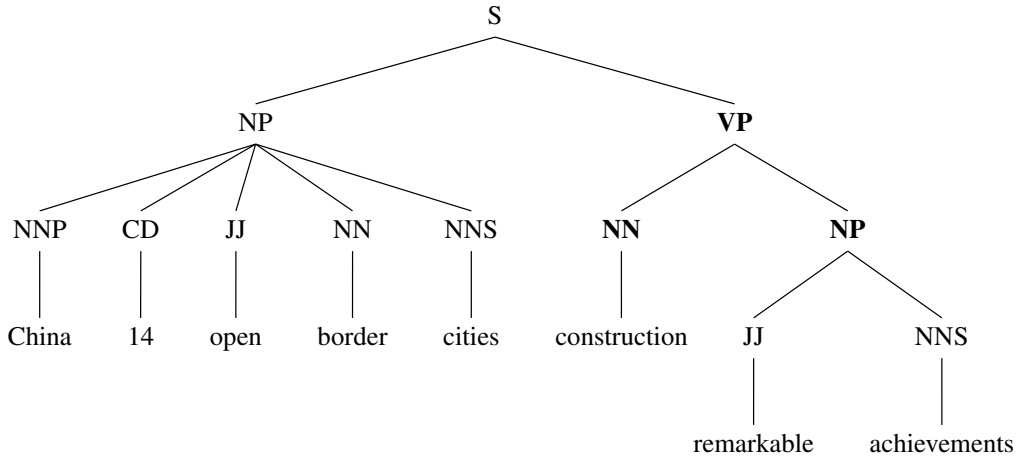
Applying the part-of-speech tagger to often ungrammtical MT output from our $n$-best lists sometimes led to unexpected results. Often the tagger tries to "fix up" ungrammatical sentences, for example by looking for a verb when none is present:

> China_NNP 14_CD open_JJ border_NN cities_NNS **achievements_VBZ** remarkable_JJ

Here, although *achievements* has never been seen as a verb in the tagger's training data, the prior for a verb in this position is high enough to cause a present tense verb tag to be produced. In addition to the inaccuaracies of the MT system, the difference in genre from the tagger's training text can cause problems. For example, while our MT data include news article headlines with no verb, headlines are not included in the Wall Street Journal text on which the tagger is trained. Similarly, the tagger is trained on full sentences with normalized punctuation, leading it to expect punctuation at the end of every sentence, and produce a punctuation tag even when the evidence does not support it:

> China_NNP 's_POS economic_JJ development_NN and_CC opening_VBG up_RP 14_CD border_NN cities_NNS remarkable_JJ **achievements_.**

The same issues affect the parser. For example the parser can create verb phrases where none exist, as in the following example where the tagger correctly did not identify a verb in the sentence:




These effects have serious implications for designing syntactic feature functions. Features such "is there a verb phrase" may not do what you expect. One solution would be features that involve the probability of a parse subtree or tag sequence, allowing us to ask "how good a verb phrase is it?". Another solution is more detailed features examining more of the structure, such as "is there a verb phrase *with a verb*?".

Although our Chinese data are "clean", rather than noisy MT output, a number of issue affect parsing on the Chinese side. Chinese parsing is highly dependent on segmentation, and our automatic segmenters are relatively inaccurate. Chinese parsing accuracy is lower than English even with perfect segmentation (82% parseval vs. 90% for English), as well as being quite a bit slower computationally.

In our data, 3% of English candidates and 4% of Chinese sentences in development set had no parse, meaning that features had to be designed to work around this situation. We generally made use of a separate feature that fired whenever a new feature could not be computed due to a missing parse, allowing the weights for both the new feature and its "not available" feature to be optimized independently.

# Chapter 2

# Implicit Syntactic Feature Functions

## 2.1 A Trio for Punctuation

### Motivation

There are often ungrammatical punctuations, especially parentheses and quotes, in the hypotheses that affect the syntactic quality of the output. Normally, we hope that the punctuations in the output sentences mostly correspond to those in the Chinese sentence, instead of appearing as something new.

### Idea No.1

The simplest approach to attack the punctuation problem is to penalize the occurrences of ungrammatical parentheses and quotes.

### Implementation

$h\_MatchParenMatchQuote(e) =$ the number of the occurrences of unmatched parentheses and quotes or empty parentheses or quotes in the English translation.

### Idea No.2

A more robust feature function applies to nearly all punctuations. Both English and Chinese tokens in any sentence can be grouped according to the nearest punctuations. By calculating the percentage of overlap between the groups of a Chinese sentence and the groups of its corresponding English sentence, this feature

function penalizes word movement around punctuations, and more severely, punctuation deletions.

## Implementation

$h\_Punc(e, f) = \%$ overlap of punctuation-grouped English and Chinese tokens $=$ the percentage of English tokens that agree with the majority their respective groups in terms of their aligned Chinese tokens' grouping

## Idea No.3

Instead of depressing the n-best score of each infected hypothesis by feature functions, we can correct the wrong parentheses and quotes by making new hypotheses.

## Implementation

1. Delete unaligned parentheses and quotes
2. Insert an opening parenthesis/quote before the first word aligned to the first Chineses word inside the parenthese
3. Insert a closing parenthesis/quote after the last word aligned to the last Chinese word inside the parentheses

## Results

None of the above three methods result in statistically significant improvement to the BLEU score.

## Analysis

The first two approaches of feature functions have a restriction in their application. When most of the hypotheses for a Chinese sentence make similar punctuation mistakes, which happens frequently, the feature functions have little discriminating power. In those kind of situations, they tend to be useless.

The second feature function extends its usage to nearly all punctuations. It penalizes punctuation deletion more harshly than word movement around punctuation, since all the tokes following that missing punctuations join the wrong group.

But it doesn't work for the deletion of those punctuations that should appear at the beginning or at the end of a sentense or next to another punctuation. This puts another limit to its application.

To work around those limitations, the third approach makes new hypotheses by making changes in the old one. We are able to do this due to the fact that minor changes in punctuations won't massively affect existing feature function values, so that we can assume the old values are still valid. However, big changes in hypotheses won't be viable under this framework, since we don't have accurate feature function values and, therefore, n-best scores.

The resulting trivial improvement to the BLEU score implies that punctuation soundness has little influence on BLEU.

## 2.2 Specific Word Penalty

### Motivation and Idea

The output of Chinese to English machine translation is often made incoherent by errant non-content (function) words, whether they be wrongly placed, inserted or deleted. For example, compared against its reference translation ("the agreement stipulates that israel is to give more land in west bank of jordan river to palestinian"), the sentence the baseline MT system outputs ("the agreement, israel and jordan in the west bank land to the palestinians") is made awkward by the insertion of "and" and the deletion of "of".

If there could be a way to detect the possibly systematic over-deletion or over-insertion of certain words, a feature function capturing such a pattern could be very useful in improving MT output. To this end, we could identify the misused words and mark each sentence with specific word penalties. In other words, we note for each hypothesis how many of these key words occur.

### Implementation

Taking a unigram count of the training corpus yields a ranking of the most frequently occurring words. By narrowing this list down to the top ten non-content words ("the", ",", ".", "and", "of", "to", "in", "a", "that" and "for"), we can take the count of each of these words and use these numbers as feature functions.

There are two ways to derive feature functions from these counts. One way is to use each word count as an individual feature (i.e. there would be ten feature

functions for the above list, one for each word). Another is to combine all of these counts into one value which could help avoid over-fitting the data. Both approaches were tried in the implementation.

### Results

Using each of the ten individual word counts as features yielded a BLEU score of 31.1 against a 31.6 baseline. Combining the counts into one total value gave a 31.7 BLEU score.

Of the ten specific word features, two ("that" and "a") contributed to the overall maximum BLEU score obtained by a greedy search algorithm over all feature functions. For both of the features, sentences with more of these words scored better on the BLEU criteria.

### Analysis and Conclusions

While using individual word penalty features results in a drop in BLEU, the presence of two word penalty features in the greedy feature combination does imply that certain words are systematically mistranslated in our baseline MT system. The limited list of words used in these experiments could easily be expanded to test other common non-content words as well.

## 2.3 Model 1 Score

### Motivation and Idea

We used IBM Model 1 (Brown et al., 1993) as one of the feature functions. As the baseline MT system uses a very complex model, we expected the simple and robust Model 1 can provide extra information that was missed in the baseline system. Especially, as Model 1 is a bag-of-word translation model and it gives the sum of all possible alignment probabilities, a lexical co-occurrence effect, or *triggering effect*, is expected. This captures a sort of topic or semantic coherence in translations.

## Implementation

As defined in (Brown et al., 1993), Model 1 gives a probability of any given translation pair, which is

$$p(\mathbf{f}|\mathbf{e}) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} t(f_j|e_i).$$

We used GIZA++ to train the model. The training data is a subset (30 million words on the English side) of the entire corpus that was used to train the baseline MT system. As it is a conditional model, we trained both $p(\mathbf{f}|\mathbf{e})$ and $p(\mathbf{e}|\mathbf{f})$, and each of them is a separate feature function. Like other feature functions, the log of the model probability was used as a feature function value, and the length normalization was not applied. For a missing translation word pair or unknown words, where $t(f_j|e_i) = 0$ according to the model, a constant $t(f_j|e_i) = 10^{-40}$ was used as a smoothing value. This constant was determined by experimenting several different values (from $10^{-5}$ to $10^{-80}$) on the development set.

## Results

The average BLEU scores (average of the best four among different 20 search initial points) are 32.5 for $p(\mathbf{f}|\mathbf{e})$, and 30.6 for $p(\mathbf{e}|\mathbf{f})$. Table 2.1 shows the additional results when we changed the model training size from 1 million words to 30 million words. As seen from the table, the scores by $p(\mathbf{f}|\mathbf{e})$ increases as the training size grows. However, the scores by $p(\mathbf{e}|\mathbf{f})$ are always lower than the baseline and fractured. We suspect there is a bug in our script for $p(\mathbf{e}|\mathbf{f})$.

| | $p(\mathbf{f}\|\mathbf{e})$ | $p(\mathbf{e}\|\mathbf{f})$ |
|---|---|---|
| 30MW | 32.5 | 30.6 |
| 20MW | 31.8 | 30.8 |
| 10MW | 31.9 | 31.1 |
| 5MW | 31.1 | 31.0 |
| 2MW | 30.9 | 30.9 |
| 1MW | 30.9 | 30.9 |

Table 2.1: BLEU scores by Model 1 with different training sizes

## Smoothing Value

The smoothing value for $t(f_j|e_i) = 0$ might be tricky. Table 2.2 and 2.3 show the BLEU scores for different smoothing values and different training corpus sizes. Table 2.2 shows the result when only one search initial point was used, and shows the BLEU scores for the development set and the test set. When applied with 20 different search initial points, and experimented with different training size, the results are shown in Table 2.3. The numbers in bold face correspond the results in Table 2.1.

Obviously, the proper smoothing value depends on the training corpus size. Another finding is that the best smoothing value for the development set was not necessarily the best for the test set.

| | $10^{-5}$ | $10^{-10}$ | $10^{-20}$ | $10^{-30}$ | $10^{-40}$ | $10^{-50}$ | $10^{-60}$ | $10^{-70}$ | $10^{-80}$ |
|---|---|---|---|---|---|---|---|---|---|
| 30MW $p(f\|e)$ dev | 32.5 | 32.5 | 32.6 | 32.6 | 32.6 | 32.5 | 32.6 | 32.6 | 32.6 |
| 30MW $p(f\|e)$ test | 32.3 | 32.3 | 31.6 | 31.6 | 31.6 | 32.1 | 31.8 | 31.8 | 31.6 |
| 30MW $p(e\|f)$ dev | 31.8 | 31.9 | 31.9 | 31.9 | 32.1 | 31.9 | 31.9 | 31.9 | 31.9 |
| 30MW $p(e\|f)$ test | 31.0 | 31.0 | 31.0 | 31.0 | 31.0 | 31.0 | 31.0 | 31.0 | 31.0 |

Table 2.2: Effect of smoothing value (search initial point = 1)

| | $10^{-5}$ | $10^{-10}$ | $10^{-20}$ | $10^{-30}$ | $10^{-40}$ | $10^{-50}$ | $10^{-60}$ | $10^{-70}$ | $10^{-80}$ |
|---|---|---|---|---|---|---|---|---|---|
| 30MW $p(f\|e)$ | 32.2 | 32.3 | 32.5 | 32.4 | **32.5** | 32.6 | 32.5 | 32.5 | 32.5 |
| 30MW $p(e\|f)$ | 30.8 | 31.4 | 30.7 | 30.7 | **30.6** | 30.6 | 30.8 | 30.8 | 30.8 |
| 1MW $p(f\|e)$ | 31.3 | 31.1 | 30.8 | 30.9 | **30.9** | 30.7 | 30.9 | 30.7 | 30.7 |
| 1MW $p(e\|f)$ | 31.3 | 30.6 | 30.5 | 30.7 | **30.9** | 30.8 | 30.9 | 30.9 | 30.9 |

Table 2.3: Effect of smoothing value and training size (search initial points = 20).

## Conclusion

The feature function by Model 1 score is one of the best performing features experimented in the workshop. The simplicity and the robustness of the model may be accounted for the performance. In terms of improving the syntactic aspect of translations, it seems to work for detecting missing content words, and for word selection coherence by the triggering effect. It is also possible that the triggering effect might work on selecting a proper verb-noun combination, or a verb-preposition combination. The strange results by $p(\mathbf{e}|\mathbf{f})$ should be investigated further.

## 2.4 Missing Content Words

### Motivation

A frequent and annoying problem of the baseline MT system is the omission of content words. Here are some examples of that phenomenon occurring in the translations produced by the baseline system (rank 1) contrasted with the next lower ranking sentence which includes this missing content word:

```
rank 1:  condemns us interference in its internal
         affairs
rank 3:  ukraine condemns us interference in its
         internal affairs

rank 1:  opposing revision of the united_nations
         charter
rank 3:  france opposes revision of the united_nations
         charter

rank 1:  small number of enterprises will be local
         governments .
rank 25: some small-scale enterprises will be managed
         by local governments .
```

We see that in the translation on ranked highest there is missing an important content word while there is a translation with a smaller probability that includes the translation of the content word. In all these examples and in a large number of occurring cases the missing content words are not due to unknown Chinese words but due to 'wrong' phrase translation pairs that have been learned because of a wrong word or sentence alignment or because the used training sentence pair did not include a plain translation of that content word but used instead for example an anaphoric reference. In all these cases the probability for the 'correct' translation probability for the phrase is much higher than the 'wrong' translation probability. Yet, the language model probability to use that (rare) content word is much lower that finally the 'wrong' translation is chosen.

### Idea

In all the above shown cases the correct translation is part of the $n$-best list. To detect that in a sentence content word translations are missing, we use a version of our single-word translation lexicon where low probability entries with a probability smaller than 0.04 are filtered away. For the purposes of this feature function, we treat all but the 200 most frequent Chinese words as content words.

### Implementation

The feature function is then implemented simply by counting the number of 'content' words that are missing in a candidate translation:

$$h(e_1^I, f_1^J) = \sum_{j=1}^{J} [f_j \text{ is not stop word}] \cdot [\text{probable-translations}(f_j) \notin e_1^I]$$

### Results

This feature function obtains comparatively large, yet not statistically significant improvement of 0.3% BLEU score from 31.6% to 31.9%.

### Analysis and Conclusions

Looking at those sentences that actually change if only that feature function is optimized, we have the feeling that many sentences actually get significantly better. In all the above given examples this feature function actually helps to get the content word in which makes the *adequacy* of that sentence significantly better.

## 2.5 Multi-Sequence Alignment of Hypotheses

### Motivation and Idea

The 1000-best lists used in this workshop research, and even the 16000-best lists to which we had access, often exhibited a lack of diversity compared with the range of possible translations. One way to tackle this problem is by recombining subparts of existing hypotheses into new ones. Multi-sequence alignment (MSA)

algorithms developed in computational genomics are one class of methods for determining what substrings are interchangeable (Gusfield, 1997).

We had hoped to use the MSA lattices to generate new hypotheses to augment the 1000-best lists. We would then have had to produce word alignments, chunkings, and parses of the new hypotheses, and of course we would not have alignment template information for them. Unfortunately, time constraints during the workshop did not allow for this approach. We did, however, use the MSA lattices to produce another class features corresponding to center hypothesis or consensus among the hypotheses. In particular, we determined three features:

1. We calculated the weight of each hypothesis' path through the MSA lattice. Each hypothesis that agreed on a certain arc contributed one to the weight of that arc.

2. We found those arcs that represented the consensus of at least half the hypotheses. A binary feature then indicated whether a particular hypothesis contained this consensus path.

3. Another feature counted the number of arcs on which a hypothesis agreed with the consensus path.

## Implementation

We used multi-sequence alignment software developed at Johns Hopkins by our colleague Gideon Mann. It takes a set of strings and a vocabulary file and outputs lattices in the AT&T FSM toolkit format. (These lattices of course represent more strings than the lattice produced by taking the union of the N-best list and minimizing.) In outline, the MSA system calculates the edit distance between each pair of strings, and then performs hierarchical agglomerative clustering.

## Results

The three implemented features achieved mediocre results (table 2.4).

## Analysis

The relative ease of movement in statistical MT, even in the relatively constrained domain of news translation, makes MSA a less than obvious fit for the problem.

| Method | BLEU % |
|---|---|
| Path score | 31.6 |
| Agrees w/ consensus? | 31.4 |
| Count arcs agreeing w/ consensus | 31.5 |

Table 2.4: MSA feature scores

We can see, nevertheless, that some idea of consensus may emerge from the alignment and help with scoring. Recombining hypotheses is still a largely unexplored in statistical machine translation (Ueffing, Och, and Ney, 2002), and we are interested in pursuing this line of research in the future.

# Chapter 3

# Shallow Syntactic Feature Functions

## 3.1 Overview

Shallow Syntactic Feature Functions are those feature functions which depend on Part-of-Speech Tagging and/or the subsequent step of Chunking, which is sometimes referred to as Shallow Parsing.

The idea behind using shallow syntactic analysis in the context of statistical machine translation is to overcome data sparseness with respect to the syntactic behavior of words. If the syntactic behavior of each word in each context were modeled individually, with the current amount of training data there would be insufficient observations to make the correct choice. If we use tools such as POS Taggers and Chunk Parsers, we hope to make use of annotated observations about syntax in order to form generalizations which are stronger than the generalizations we can make based on what is observed in the training data for the baseline MT system.

However, the POS information is modeled implicitly in the baseline system, which was trained on more data than was POS tagged. Also, the Chunks found by the Chunk Parser are not at a much higher granularity than the Alignment Template granularity, and was also trained on the same training set as the POS tagger. Because of these issues there is a danger that at the Shallow Syntactic level it will not be possible to make generalizations that capture information which is not already present in the baseline system.

There are several advantages to working at the shallow syntactic level, rather than with parser output. Part-of-Speech Taggers and Chunkers are efficient. Decisions made by these tools are local, and so they may function better than parsers on

noisy MT hypotheses. 1.3 M tagged parallel sentences were available for training. Finally, using simpler models allows a quicker reaction to the problems evident from contrastive error analysis.

The generalizations possible using shallow syntax must necessarily be less powerful than those available using the full parser output (Deep Syntax), or a decomposition of the full parser output (Tricky Syntax), both of which rely on richer knowledge sources. These feature functions are described in subsequent sections.

## 3.2 Part-of-Speech and Chunk Tag Counts

### Motivation

There are low-level syntactic problems with the baseline system where the baseline is systematically overgenerating or undergenerating certain parts of speech and chunk types. The contrastive error analysis shows that there is overgeneration of article, comma, singular nouns and undergeneration of pronouns, past tense, coordination, plural nouns, etc.

### Idea

The reranker can learn to favor a more balanced distribution of tags by learning whether to favor sentences with less or more of a certain tag. If past tense verbs are being systematically undergenerated, a feature function which simply counts the number of past tense verbs generated should receive a high weight from the Maximum-BLEU optimization. Likewise, a feature function which counts the number of Chinese N tags which are translated only to non-N tags in English (using the word alignment) should receive a highly negative weight in the case that this is generally incorrect.

### Implementation

Individual feature functions were implemented using tag count and tag count differences for both POS tags and Chunk tags. Note that this is not dependent on the word alignment as it is characteristic of the tag distributions on the English side, or of the difference in tag counts from the Chinese to the English.

Tag translation counts which depend on the word alignment were implemented for POS tags. This involves looking at the predefined mapping of a particular set of Chinese tags to a particular set of English tags. In particular, we implemented the number of Chinese N tags translated only as non-N tags in English; the number of V tags in Chinese translated only as non-V tags in English; and the number of P tags in Chinese which were only translated non-P tags in English. Others could easily be imagined, in particular see the appendix of (Xia, 2000) containing a table of tag equivalences from the Penn Treebank to the Chinese Treebank.

## Examples

- Number of NPs in English hypothesis
- Difference in number of NPs from Chinese to English
- Number of Chinese N tags translated to only non-N tags in the English hypothesis

## Results

See table 3.1.

## Analysis and Conclusion

The performance of the simplest features of this type possible, the simple tag count features is not much better than the baseline when considered individually. This is not surprising as this information is implicit in the trigram language model using in the baseline MT system. However, for feature combination these features appeared to be more useful. One possible explanation for this is that they counteract biases present in other more sophisticated feature functions.

The more complex features involving the word alignment did not perform better than the baseline, which is probably a function of the simplicity of the models and their integration into the weight-learning framework. In addition, it may reflect problems with the Chunk and/or POS level analyses.

| | $BLEU$ |
|---|---|
| POS Tag WDT | 31.2 |
| POS Tag VBG | 31.3 |
| POS Tags JJS,NNP,NNS,SYM,TO,UH,VBZ,WP,colon,leftdoublequote | 31.4 |
| POS Tags CC,CD,DT,EX,JJ,MD,NN,WP$,$,period,rightdoublequote | 31.5 |
| POS Tags FW,IN,LRB,LS,NNPS,PDT,PRP$,RB,RBR,RBS,RP,RRB,VBD | 31.6 |
| POS Tags JJR,VB,VBN,VBP,WRB,comma | 31.7 |
| Chunker Tag VP | 31.4 |
| Chunker Tags NP,O | 31.5 |
| Chunker Tags ADJP,ADVP,INTJ,LST,SBAR | 31.6 |
| Chunker Tags CONJP,PP,PRT | 31.7 |
| Difference in number of NPs | 31.4 |
| Difference in number of VPs | 31.4 |
| Verb translated as non-verb word(s) | 31.3 |
| Noun translated as non-noun word(s) | 31.4 |
| Pronoun translated as non-pronoun word(s) | 31.6 |

Table 3.1: BLEU Scores of Tag Features

## 3.3 Tag Fertility Models

### Motivation

The use of feature functions based on probabilistic Tag Fertility models is motivated by the same observation as the previous Tag Count features, which is that the tag distribution of the top hypotheses output by the baseline system is often wrong. In this case however, the idea is to model how surprised we would be that an English hypothesis with a particular tag distribution is the correct translation of the Chinese source sentence.

### Idea

The idea is to model the English tag distribution. There are two options, either to probabilistically model the expected English tag distribution in general, or specifically to model the expected English distribution of tags given the Chinese tags.

## Implementation

The implementation is essentially a bag of tags. In principal, it is similar to IBM Model 1 (Brown et al., 1993), but tags with zero counts are explicitly modeled.

One set of models models the probability of the English tag distribution. This is done only for POS currently, but it could be extended to the chunk distribution:

$P(NN_e = 2)$

Another set of models models the conditional probability of number of English tags given number of Chinese tags (POS, Chunk):

$P(NP_e = 2|NP_f = 1)$

$P(CC_e = 1|CC_f = 0)$

$P(CD_e = 1|M_f = 1)$

The probabilities in both of these cases were estimated using simple maximum likelihood from the 1.3M tagged sentences.

Several different parameter formulations were tried, using the lists of equivalences from (Xia, 2000), and lists formed by hand.

These were then combined into a single feature function by simple multiplication, and a log-linear combination of the individual probabilities learned using Maximum BLEU was also tried.

## Results

| | $BLEU$ |
|---|---|
| Mapping of All to All Tags | 30.7 |
| Mapping of All Chunk Tag to All Chunk Tags | 31.5 |
| Mapping of Identical Tags only | 31.5 |
| Mapping of Chinese POS tags to English POS tags | 31.4 |

Table 3.2: BLEU Scores of Tag Fertility Features

## Analysis and Conclusion

These feature function sdid not work as well as hoped.

Clearly, one large issue is smoothing. The estimation of the parameters here was simplistic, as the counts for obviously related probabilities such as $P(NP_e = 1|NP_c = 1)$ and $P(NP_e = 2|NP_c = 2)$ are independently estimated.

A parameterized distribution on the ratio of the Chinese to the English for each tag might be better, as there would be fewer free parameters.

## 3.4 Projected POS Language Model

### Motivation

The baseline MT system has a weak model of word movement.

### Idea

Use Chinese POS tag sequences as surrogates for Chinese words to model movement. Chinese words are too sparse to model movement, but an attempt to model movement using Chinese POS may be more successful.

### Implementation

Chinese POS sequences are projected to English using the word alignment. Relative positions are indicated for each Chinese tag. The feature function was also tried without the relative positions. NULL alignments were handled by inserting the literal 'NULL'. In addition, a variant was tried where NULL-aligned English words were left in place for both training and testing.

The feature function itself is the log probability of a trigram language model built on the projected Chinese tags and positions.

This is similar to the HMM Alignment model (Vogel, Ney, and Tillmann, 1996) but in this case movement is calculated on the basis of parts of speech.

### Example

| CD_+0_M_+1 | NN_+3 | NN_-1 | NN_+2_NN_+3 |
|---|---|---|---|
| 14 $(measure)$ | open | border | cities |

Table 3.3: Example

The table shows an example tagging of an English hypothesis showing how it was generated from the Chinese sentence. The feature function is the log probability output by a trigram language model over this sequence.

### Results

| | $BLEU$ |
|---|---|
| Projected POS LM no positions | 31.4 |
| Projected POS LM positions | 31.5 |
| Projected POS LM positions with NULL words lexicalized | 31.7 |

Table 3.4: BLEU Scores of Projected POS Features

### Analysis and Conclusion

The Projected POS feature function with positions and lexicalized NULL words was one of the strongest performing shallow syntactic feature functions. This is undoubtedly due to the weak movement model of the baseline system. This feature function should be further investigated, as it may represent an interesting trade-off between purely word-based models, and full generative models based upon shallow syntax.

## 3.5 Aligned POS-Tag Sequences

### Motivation and Idea

We used aligned POS-tag sequences as one of the feature functions. The alignment templates used in the baseline MT system is on the word level, so that the probability was computed on lexical items directly. However, the distribution of lexical items is always very sparse. In natural language processing, people always use POS tags as the smoothing model for lexical items. Therefore we used aligned POS-tag sequences as the smoothing model for the aligned templates in the baseline MT system.

### Implementation

For each template in a given Chinese sentence and its English translation, we first replace all the single words with part-of-speech tags. We have designed two models to use aligned POS sequences.

The first one is a unigram language model on aligned templates

$$p(\mathbf{f}, \mathbf{e}) = \prod_{(s_f, s_e) \in AT} p(s_f, s_e),$$

where $(s_f, s_e)$ is a pair of aligned POS-tag sequences, $AT$ is the set of all the aligned templates for the given Chinese sentence and its English translation. $p(s_f, s_e)$ is estimated in the training data as a unigram model.

The second model is a conditional model given the Chinese POS-tag sequence.

$$p(\mathbf{e}, \mathbf{f}) = p(\mathbf{f}) \prod_{(s_f, s_e) \in AT} p(s_e | s_f),$$

where $p(s_e|s_f))$ is estimated on the training data, and $p(\mathbf{f})$ is estimated with a trigram language model.

## Results

The average BLEU scores (average of the best four among different 20 search initial points) are 31.6 for the unigram model, and 31.4 for the conditional model.

## Analysis and Conclusion

The performance of the feature function of aligned POS-tag sequences is not as good as supposed. The main reason is because of the rule based module used in the baseline MT system. For those words translated with rules, the baseline system did not output alignment information, so that we cannot recover the aligned sequences if they are handled by some rules. In addition, the use of the rules is not consistent in the 1000 best results. This has a great impact on the performance of the alignment based models. Once this problem is solved, the performance of these feature functions is believed to be much more better.

# Chapter 4

# Deep Syntactic Feature Functions

## 4.1 Grammaticality Test of English Parser

We performed an experiment to test if the English (Collins) parser assigns a higher probability to grammatical sentences. We looked at two features - raw parser probabilities output by the Collins parser and parser probabilities divided by the unigram language model probability. We now report experiments under these features.

### 4.1.1 Parser Probability

We first present experiments using the raw parser probabilities.

We report the average log probability assigned by the Collins parser to the 1-best (produced), oracle and the reference translations. The results are presented in Table 4.1.

| Hypothesis | 1-best | Oracle | ref1 | ref2 | ref3 | ref4 |
|---|---|---|---|---|---|---|
| log(parseProb) | -147.2 | -148.5 | -148.0 | -157.5 | -155.6 | -158.6 |

Table 4.1: Average Parser Log Probability assigned to produced/oracle/reference Translations on the development set.

We observe that the average parser log-probability of the 1-best translation is higher than the average parse log probability of the oracle or the reference translations.

We next report the number of sentences (expressed as a percentage) on which the parser assigned higher probability to oracle/reference translations in comparison to the produced translation. These results are summarized in Table 4.2.

| Hypothesis | Oracle | ref1 | ref2 | ref3 | ref4 |
|---|---|---|---|---|---|
| % of sentences | 42.6 | 46.0 | 32.3 | 32.7 | 29.9 |

Table 4.2: % of sentences in the development set for which the parser assigned higher probability to the oracle/reference translation compared to the produced translation

### 4.1.2 Parser Probability Divided by Unigram Language Model Scores

We now present the results obtained by dividing the parser-probability by the unigram language model probability. The unigram language model was trained on the Wall Street Journal corpus.

We first report the average normalized log probability scores obtained on the produced, oracle and the reference translations under this feature in Table 4.3.

| Hypothesis | 1-best | Oracle | ref1 | ref2 | ref3 | ref4 |
|---|---|---|---|---|---|---|
| log(parseProb/UniProb) | 17.85 | 18.44 | 26.33 | 30.97 | 28.77 | 24.63 |

Table 4.3: Average Normalized Parser Log Probability assigned to produced/oracle/reference Translations on the development set.

We next report the number of sentences (expressed as a percentage) on which the parser assigned higher normalized probability to oracle/reference translations in comparison to the produced translation. These results are summarized in Table 4.4.

| Hypothesis | Oracle | ref1 | ref2 | ref3 | ref4 |
|---|---|---|---|---|---|
| % of sentences | 50.4 | 65.6 | 72.3 | 67.5 | 62.0 |

Table 4.4: % of sentences in the development set for which the parser assigned higher normalized probability to the oracle/reference translation compared to the produced translation.

## 4.2 Tree to String Model

### Model Description

A *tree-to-string* model is one of syntax-based translation models. The model is a conditional probability $p(\mathbf{f}|T(\mathbf{e}))$. Here, we used a model defined in (Yamada and Knight, 2001) and (Yamada and Knight, 2002).

Internally, the model performs three types of operations on each node of a parse tree. First, it *reorders* the child nodes, such as changing VP $\rightarrow$ VB NP PP into VP $\rightarrow$ NP PP VB. Second, it *inserts* an optional word at each node. Third, it *translates* the leaf English words into Chinese words. These operations are stochastic and their probabilities are assumed to depend only on the node, and are independent of other operations on the node, or other nodes. The probability of each operation is automatically obtained by a training algorithm, using about 780,000 English parse tree - Chinese sentence pairs. The probability of these operations $\theta(e^k_{i,j})$ are assumed to depend on the edge of the tree being modified, $e^k_{i,j}$ but independent of everything else, giving the following equation,

$$p(\mathbf{f}|T(\mathbf{e})) = \sum_{\Theta} \prod_{\theta(e^k_{i,j})} p(\theta(e^k_{i,j})|e^k_{i,j}) \tag{4.1}$$

where $\Theta$ varies over the possible alignments between the $\mathbf{f}$ and $\mathbf{e}$ and $\theta(e^k_{i,j})$ is the particular operations (in $\Theta$) for the edge $e^k_{i,j}$.

The model is further extended to incorporate phrasal translations performed at each node of the input parse tree (Yamada and Knight, 2002). An English phrase covered by a node can be directly translated into a Chinese phrase without regular reorderings, insertions, and leaf-word translations.

### Implementation

The model was trained before the workshop, using about 780,000 English parse tree - Chinese sentence pairs. There are about 3 million words on the English side, and they were parsed by Collins' parser.

Since the model is computationally expensive, we added some limitations on the model operations. As the base MT system does not produce a translation with a big word jump, we restrict the model not to reorder child nodes when the node covers more than seven words. For a node which has more than four children, the reordering probability is set to be uniform. We also introduced pruning, which

discards partial (subtree-substring) alignments if the probability is lower than a threshold.

The model gives a sum of all possible alignment probabilities for a pair of a Chinese sentence and an English parse tree. We also calculate the probability of the best alignment according to the model. Thus, we have the following two feature functions.

$$h_{\text{TreeToStringSum}}(\mathbf{e}, \mathbf{f}) = \log(\sum_{\Theta} \prod_{\theta(e^k_{i,j})} p(\theta(e^k_{i,j})|e^k_{i,j})) \tag{4.2}$$

$$h_{\text{TreeToStringViterbi}}(\mathbf{e}, \mathbf{f}) = \log(\max_{\Theta} \prod_{\theta(e^k_{i,j})} p(\theta(e^k_{i,j})|e^k_{i,j})) \tag{4.3}$$

## Results

As the model is computationally expensive, we sorted the $n$-best list by the sentence length, and processed them from the shorter ones to the longer ones. We used 10 CPUs for about five days, and 273/997 development sentences, and 237/878 test sentences were processed.

The average BLEU score (average of the best four among different 20 search initial points) was 31.7 for both $h_{\text{TreeToStringSum}}$ and $h_{\text{TreeToStringViterbi}}$. Among the processed development sentences, the model preferred the oracle sentences over the produced sentence in 61% cases.

## Analysis and Conclusion

The biggest problem of this model is that it is computationally very expensive. It only processed less than 30% of the $n$-best lists in long CPU hours. In addition, we processed short sentences only. For long sentences, it is not practical to use this model as it is.

However, even though it processed small portion of the $n$-best lists, the BLEU score was not that bad. Actually, it is one of the best results among the feature functions which utilized parse trees.

One of the possible ways to reduce the computational cost is to break down a long sentence into a set of small fragments. We used such a tool, called *machete* (see Section ?.?), to preprocess the $n$-best lists. Using the same model described above, it now covers almost all sentences. The average BLEU score was 31.5, which is lower than the case described above. However, we found that the tool

had some problems in handling punctuation and tokenization. We expect better results by sharpening the machete.

## 4.3 Tree to Tree Alignment

A *tree-to-tree* model translation makes use of syntactic tree for both the source and target language. As in the tree-to-string model, a set of operations apply, each with some probability, to transform one tree into another. However, when training the model, trees for both the source and target languages are provided, in our case from the Chinese and English parsers.

Work during the workshop began with the tree-to-tree alignment model presented by Gildea (2003). We begin with a description of the basic probability model in the following section before describing the extensions developed during the workshop. The major extension were adapting the model to handle dependency trees (Section 4.4), and making use of the word-level alignments produced by the baseline MT system for the purposes of rescoring (Section 4.3). The basic feature function produced by a tree-to-tree alignment is the probability assigned by the model. Further features derived from the alignments are described in Section 4.5.

### Tree to Tree Model

The tree-to-tree alignment model has tree transformation operations similar to those of the tree-to-string model described above. However, the transformed tree must not only match the surface string of the target language, but also the tree structure assigned to the string by the parser. In order to provide enough flexibility to make this possible, additional tree transformation operations allow a single node in the source tree to produce two nodes in the target tree, or two nodes in the source tree to be grouped together and produce a single node in the target tree. The model can be thought of as a synchronous tree substitution grammar, with probabilities parameterized to generate the target tree conditioned on the structure of the source tree.

The probability $P(T_b|T_a)$ of transforming the source tree $T_a$ into target tree $T_b$ is modeled in a sequence of steps proceeding from the root of the target tree down. At each level of the tree:

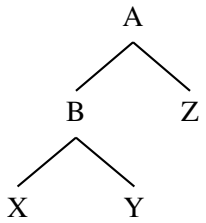
1. At most one of the current node's children is grouped with the current node in a single *elementary tree*, with probability $P_{elem}(t_a|\varepsilon_a \Rightarrow children(\varepsilon_a))$,

conditioned on the current node $\varepsilon_a$ and its children (ie the CFG production expanding $\varepsilon_a$).

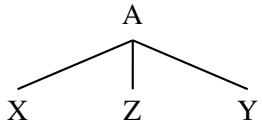
2. An alignment of the children of the current elementary tree is chosen, with probability $P_{align}(\alpha|\varepsilon_a \Rightarrow children(t_a))$. This alignment operation is similar to the re-order operation in the tree-to-string model, with the extension that 1) the alignment $\alpha$ can include insertions and deletions of individual children, as nodes in either the source or target may not correspond to anything on the other side, and 2) in the case where two nodes have been grouped into $t_a$, their children are re-ordered together in one step.

In the final step of the process, as in the tree-to-string model, lexical items at the leaves of the tree are translated into the target language according to a distribution $P_t(f|e)$.

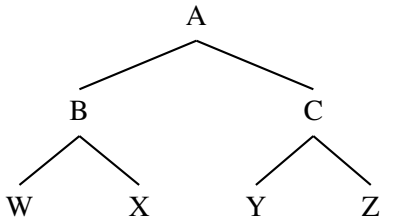
Allowing non-1-to-1 correspondences between nodes in the two trees is necessary to handle the fact that the depth of corresponding words in the two trees often differs. A further consequence of allowing elementary trees of size one or two is that some reorderings not allowed when reordering the children of each individual node separately are now possible. For example, with our simple tree




if nodes A and B are considered as one elementary tree, with probability $P_{elem}(t_a|\mathrm{A} \Rightarrow \mathrm{BZ})$, their collective children will be reordered with probability $P_{align}(\{(1,1)(2,3)(3,2)\}|\mathrm{A} \Rightarrow \mathrm{XYZ})$




giving the desired word ordering XZY. However, computational complexity as well as data sparsity prevent us from considering arbitrarily large elementary trees, and the number of nodes considered at once still limits the possible alignments. For example, with our maximum of two nodes, no transformation of the tree




is capable of generating the alignment WYXZ.

| | Tree-to-String | Tree-to-Tree |
|---|---|---|
| elementary tree grouping | | $P_{elem}(t_a \mid \varepsilon_a \Rightarrow children(\varepsilon_a))$ |
| re-order | $P_{order}(\rho \mid \varepsilon \Rightarrow children(\varepsilon))$ | $P_{align}(\alpha \mid \varepsilon_a \Rightarrow children(t_a))$ |
| insertion | $P_{ins}(\text{left}, \text{right}, \text{none} \mid \varepsilon)$ | $\alpha$ can include "insertion" symbol |
| lexical translation | $P_t(f \mid e)$ | $P_t(f \mid e)$ |
| with cloning | $P_{ins}(\text{clone} \mid \varepsilon)$ | $\alpha$ can include "clone" symbol |
| | $P_{makeclone}(\varepsilon)$ | $P_{makeclone}(\varepsilon)$ |

Table 4.5: Model parameterization

In order to generate the complete target tree, one more step is necessary to choose the structure on the target side, specifically whether the elementary tree has one or two nodes, what labels the nodes have, and, if there are two nodes, whether each child attaches to the first or the second. Because we are ultimately interested in predicting the correct target string, regardless of its structure, we do not assign probabilities to these steps. The nonterminals on the target side are ignored entirely, and while the alignment algorithm considers possible pairs of nodes as elementary trees on the target side during training, the generative probability model should be thought of as only generating single nodes on the target side. Thus, the alignment algorithm is constrained by the bracketing on the target side, but does not generate the entire target tree structure.

While the probability model for tree transformation operates from the top of the tree down, probability estimation for aligning two trees takes place by iterating through pairs of nodes from each tree in bottom-up order, as sketched below:

**for all** nodes $\varepsilon_a$ in source tree $T_a$ in bottom-up order **do**
  **for all** elementary trees $t_a$ rooted in $\varepsilon_a$ **do**
    **for all** nodes $\varepsilon_b$ in target tree $T_b$ in bottom-up order **do**
      **for all** elementary trees $t_b$ rooted in $\varepsilon_b$ **do**
        **for all** alignments $\alpha$ of the children of $t_a$ and $t_b$ **do**
          $\beta(\varepsilon_a, \varepsilon_b)$ += $P_{elem}(t_a \mid \varepsilon_a) P_{align}(\alpha \mid \varepsilon_i) \prod_{(i,j) \in \alpha} \beta(\varepsilon_i, \varepsilon_j)$

**end for**
**end for**
**end for**
**end for**
**end for**

The outer two loops, iterating over nodes in each tree, require $O(|T|^2)$. Because we restrict our elementary trees to include at most one child of the root node on either side, choosing elementary trees for a node pair is $O(m^2)$, where $m$ refers to the maximum number of children of a node. Computing the alignment between the $2m$ children of the elementary tree on either side requires choosing which subset of source nodes to delete, $O(2^{2m})$, which subset of target nodes to insert (or clone), $O(2^{2m})$, and how to reorder the remaining nodes from source to target tree, $O((2m)!)$. Thus overall complexity of the algorithm is $O(|T|^2 m^2 4^{2m}(2m)!)$, quadratic in the size of the input sentences, but exponential in the fan-out of the grammar.

### Tree-to-Tree Clone Operation

Allowing m-to-n matching of up to two nodes on either side of the parallel treebank allows for limited non-isomorphism between the trees, as in Hajič et al. (2002). However, even given this flexibility, requiring alignments to match two input trees rather than one often makes tree-to-tree alignment more constrained than tree-to-string alignment. For example, even alignments with no change in word order may not be possible if the structures of the two trees are radically mismatched. This leads us to think it may be helpful to allow departures from the constraints of the parallel bracketing, if it can be done in without dramatically increasing computational complexity.

For this reason, we introduce a clone operation, which allows a copy of a node from the source tree to be made anywhere in the target tree. After the clone operation takes place, the transformation of source into target tree takes place using the tree decomposition and subtree alignment operations as before. The basic algorithm of the previous section remains unchanged, with the exception that the alignments $\alpha$ between children of two elementary trees can now include cloned, as well as inserted, nodes on the target side. Given that $\alpha$ specifies a new cloned node as a child of $\varepsilon_j$, the choice of which node to clone is made as in the tree-to-string model:

$$P_{clone}(\varepsilon_i|\text{clone} \in \alpha) = \frac{P_{makeclone}(\varepsilon_i)}{\sum_k P_{makeclone}(\varepsilon_k)}$$

Because a node from the source tree is cloned with equal probability regardless of whether it has already been "used" or not, the probability of a clone operation can be computed under the same dynamic programming assumptions as the basic tree-to-tree model. As with the tree-to-string cloning operation, this independence assumption is essential to keep the complexity polynomial in the size of the input sentences.

For reference, the parameterization of all four models is summarized in Table 4.5.

## Experiments

We trained the parameters of the tree transformation operations on 42,000 sentence pairs of parallel Chinese-English data from the Foreign Broadcast Information Service (FBIS) corpus. The lexical translation probabilities $P_t$ were trained using IBM Model 1 on the 30 million word training corpus. This was done to overcome the sparseness of the lexical translation probabilities estimated while training the tree-to-tree model, which was not able to make use of as much training data.

As a test of the tree-to-tree model's discrimination, we performed an oracle experiment, comparing the model scores on the 1st sentence in the $n$-best list with candidate giving highest BLEU score. On the 1000-best list for the 993 sentence development set, restricting ourselves to sentences with no more than 60 words and a branching factor of no more than five in either the Chinese or English tree, we achieved results for 480, or 48% of the 993 sentences. Of these 480, the model preferred the produced over the oracle 52% of the time, indicating that it does not in fact seem likely to significantly improve BLEU scores when used for reranking.

## Using Word-Level Alignments

One source of information in our $n$-best list that we have not exploited is the word-level alignments. Since the English sentences were in fact generated by the system, it "knows" which Chinese word or words were used to generate each English word, meaning that the tree-to-tree model does not need to figure this out for itself.

We modified the tree-to-tree alignment code to take as input a word-level alignment for each sentence. In this setting, rather using the model to find the best word-level alignment, it is serving merely to provide a measure of the syntactic similarity of the Chinese sentence and its English candidate translation, given

the alignment between to two. This was implemented by using a new set of lexical translation probabilities for each sentence, based on the word-level alignment. Word pairs that align were given probability one (this is not strictly a probability now, as a word on the source side can be aligned with more than one target word, meaning the translation probabilities sum to more than one). Words with no alignment in the other language were given insertion or deletion probability of one, and all other lexical translation probabilities were fixed to zero.

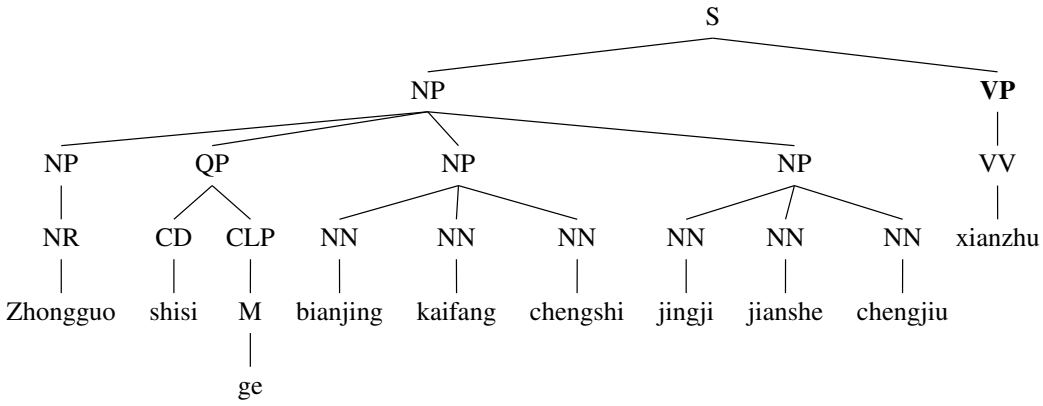
This modification to the alignment algorithm had the benefit of significantly reducing the search space and making the tree-to-tree alignment run much faster. The speed-up made it possible to align all 1000 candidates from our $n$-best list. Three features were added to the baseline system: the probability of the tree-to-tree alignment, another feature that fired when one or both sentences were skipped because of a branching factor of more than six or length of more than sixty, and a third feature that fired when no alignment was found despite both sentences being within the specified branching factor. This can happen because the root of the dependency tree cannot be cloned.

## 4.4 Dependency Tree-to-Tree Alignments

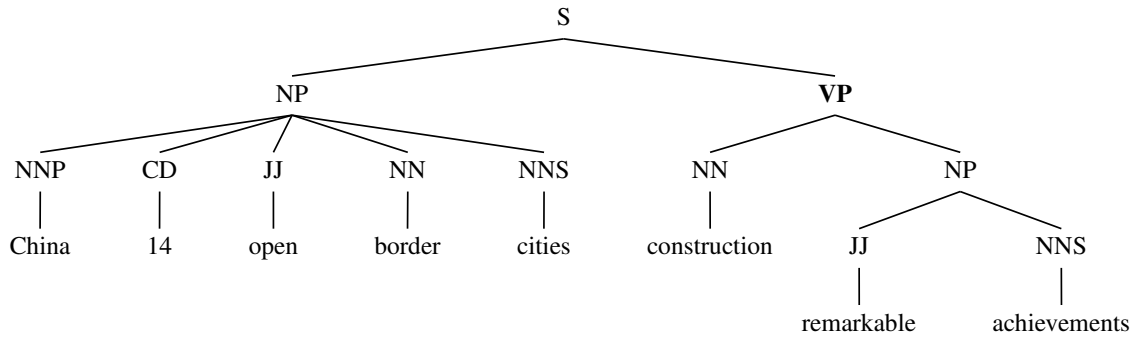
### Motivation

When performing alignment for the Chinese and English n-best constituency parse trees, two problems arise.

First, the n-best parses are often unreliable and may contain "hallucinated" structures. As parsers which have been trained on well-formed sentences expect to parse well-formed sentences, the parser's output can be corrupted for ungrammatical n-best hypotheses. For example, in the following set of sentences, the English n-best has no tagged verbs. However, at a higher level in the parse tree the parser has inserted a verb phrase label.




**Chinese constituency tree**




**English constituency tree**

Unreliable parses can be detrimental to the alignment process. In the above example, if the alignment program sees a verb phrase in both the Chinese and English, it may assign a high probability to the two trees aligning when in fact the English tree contains no verb.

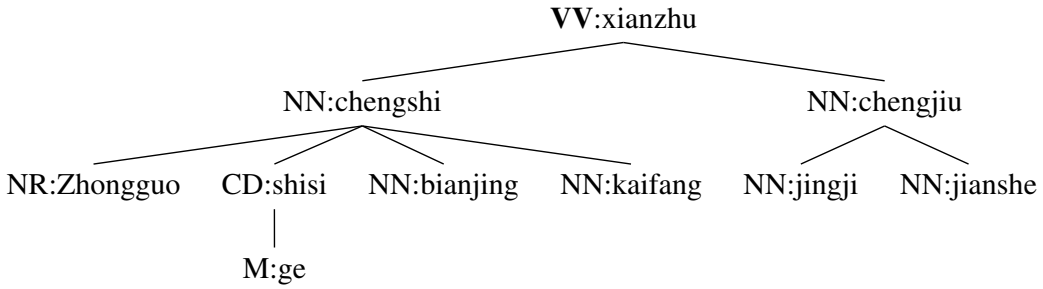
The second problem with aligning constituency trees is the sometimes arbitrary depth within a parse tree. For example, looking at the two above trees, we see that the noun phrase structures on the left of the two trees have varying depth (the Chinese tree reaches a depth of six while the English side only four). However, despite the different depths, the words that the head noun nodes span are actually very accurate translations; the number of intermediate noun phrase nodes that have been inserted above the leaves are more or less arbitrary.

As a result, even though two structures might be very similar, a difference in tree depth causes the aligner to perform node merges in order to align the terminal nodes correctly which comes at a cost to the alignment probability. In a more

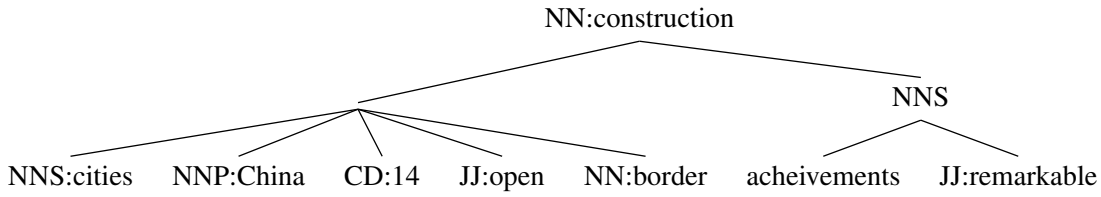
severe case, the aligner may not even be able to align the corresponding terminal nodes because only one merge is possible at any point in the tree. In this case the aligner will clone the subtree which comes at an even greater probability cost. As a result, the alignment score for this good translation has been unnecessarily lowered.

## Idea

Dependency trees offer a solution to both of the drawbacks that constituency trees present for alignment. Instead of representing the overall grammatical structure of a sentence, dependency trees represent the relationships between the words. Essentially, they show graphically which words depend on each other via a language's head rules. The result is a tree in which every node represents a word, and its children are words that depend on it. Traversing down the tree, a node's child may in turn have children (dependents) of its own. Below are the previous Chinese and English constituency trees converted to dependency trees.




**Chinese dependency tree**




**English dependency tree**

By converting the parsed constituency trees into dependency trees, we eliminate the "hallucinated" node labels. By only using the words and their part of

speech tag, any of the parser's corrupted output is ignored while still preserving the overall grammatical relationships between the words. Now, in the above example there is no verb phrase in the English tree where there should not be.

Dependency trees also solve the second problem we encountered with aligning constituency tree by ignoring the depth within a parse tree structure. Since all non-terminal nodes are ignored and only the grammatical relations between words are shown, the number of non-terminal nodes above the words is not represented. In the above example, the similar noun phrases in the left of the trees now have very similar tree structures as well.

## Implementation

Once we use English and Chinese head rules to convert each constituency tree to a dependency tree, we modify the tree-to-tree alignment algorithm in two ways.

First, we must address the issue of word translation. Now that the trees contain words in every single node as opposed to only leaf nodes, we must account for translation occuring in these new locations. This is done simply by multiplying in the probability of word translation occuring between the words of every two nodes that align.

The second change we make to the tree-to-tree alignment algorithm is to introduce a lexicalized reordering parameter. The original alignments take into consideration the probability that, given the label, a Chinese node's children have been so reordered in the English tree. However now that we have dependency trees representing word relationships, we can make a more informative and perhaps powerful estimation which judges the probability that, given the Chinese word and its children's labels, its children have been so reordered in the English tree.

With these modifications, we then train the train the alignment program on 40,000 pairs of Chinese-English dependency parses to obtain the parameters for the EM algorithm. With these trained parameters, we can run the alignment program and use the probability of each n-best hypothesis aligning with its source Chinese sentence as a feature function.

Another feature to be extracted from the alignment compares what type of words align with other types of words via the part of speech label. Counting the number of times that a noun aligns with a noun or a verb aligns with a verb gives an estimate as to how often similar words are aligning and could indicate at a finer grain how well two sentences align.

### Results

Using the probability of the source Chinese dependency parse aligning with the n-best hypothesis dependency parse as a feature function, making use of the word-level alignments as decribed in Section 4.4, yields a 31.6 BLEU score (against a 31.6 baseline).

Using noun alignment counts as a feature function yields a 31.4 BLEU score, and using verb alignment as a feature function yields a 31.4 BLEU score.

### Analysis and Conclusions

Even by using dependency trees to eliminate much of the noise created by unreliable parsing, using the alignments as a source for feature functions is currently not very effective.

One reason we see insignificant improvements can be explained by inadequate parsing tools for ungrammatical n-best sentences. Even at the most basic level of part of speech, words are incorrectly tagged (e.g. the last word of a headline being tagged as a punctuation mark). If parsing for machine translation output could be improved, one could also expect to gain more utility for using tree-to-tree alignment probability and node part of speech alignment as an indicator for the accuracy of a hypothesis.

## 4.5 Main Verb Arguments

### Motivation and Idea

It is not uncommon to see machine translation output which has misplaced or simply omitted entire syntactic entities. For example, the object of one verb may have become the subject of another or may have been simply untranslated. It would be desirable to make sure that the most important "players", such as the subject and object of a sentence, have been preserved.

Dependency trees offer an organization that is naturally conducive to tracking the argument structure of a sentence. Because they are formed by applying a language's head rules, one can see if a word is an argument of another by looking for a parent-child relationship (children are arguments of their parent node). By analyzing the dependency tree parses of a Chinese sentence and English hypothesis, we can compare if the correct arguments have been transfered in translation.

### Implementation

Here we create a feature function that counts the difference in the number of arguments to the main verb between a Chinese sentence and an English translation.

To do this, we perform a breadth-first search through the dependency tree. The first verb that is found is labeled as the main verb of the sentence since it is closest to the top of the dependency tree. After the main verb is found, counting the number of its immediate children yields the number of arguments the main verb has taken.

### Results

Using the difference in the number of arguments to the main verb as a feature function yields a BLEU score of 31.6 (over a 31.6 baseline).

### Analysis and Conclusions

While preserving the number of arguments to the main verb would seem to be helpful in preserving the meaning of a translation, the BLEU score does not show any significant increase.

One explanation could lie in the sometimes unreliable parsing of the n-best sentences. As described in dependency tree-to-tree alignment, parsers that have been trained on grammatical sentences create errors when parsing ungrammatical machine translation output. The noise generated from this could explain the inefficacy of a feature function that tracks the main verb arguments via dependency parse trees.

Another problem in this approach is the lack of consideration to which arguments exactly have been transferred via the word alignment. Similar feature functions which do incorporate word alignment yield much higher performance (e.g. Projected Dependency feature functions).

## 4.6 Flipped Dependencies

### Motivation

Word dependencies (see e.g., section “Dependency Tree-to-Tree Alignments” are often used as a robust representation for sentences. Dependencies capture the

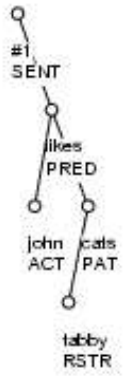


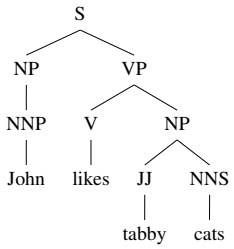

Figure 4.1: Sample dependency tree.

relationship between a head word and the words that depend on it. For example in the sentence

John likes tabby cats.

The root-level head is “likes”; two words “John” and “cats” depend on “likes” and, finally, “tabby” depends on “cats”.

The corresponding constituent tree for the same sentences is shown below:




English language head rules allow for dependency trees to be built with little effort from arbitrary sentences.

In 4.8 we indicated that a large number of the NBEST candidates are not grammatical sentences and that fact challenges the construction of S-rooted parse trees by either the Collins or Charniak parsers.

Under such circumstances one may wonder whether some sentences that fail to parse to S or some sentences which parse to S *incorrectly* may not contain some interesting information at a level below S, e.g., at the level of individual NPs. It makes only sense to consider whether dependency information can be used to capture some interesting properties of the NBEST sentences.

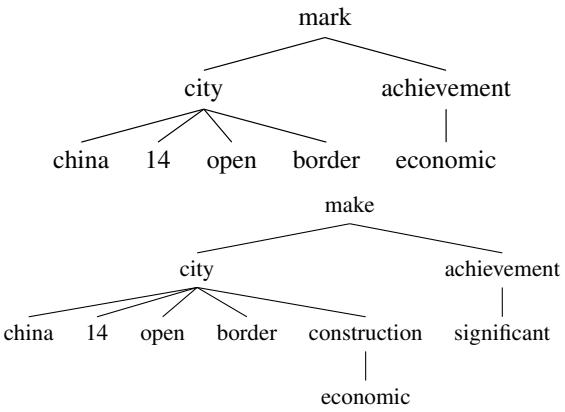



Figure 4.2: Dependency trees for two sample NBEST sentences.

```
fourteen chinese open border cities make significant achievements in
economic construction

significant accomplishment achieved in the economic construction of
the fourteen open border cities in china

in china , fourteen cities along the border opened to foreigners
achieved remarkable economic development

economic construction achievement is prominent in china 's fourteen
border opening up cities .
```

Figure 4.3: Four reference translations for sentence 0 of the development set.

Other dependency models are described elsewhere in this report. This section addresses the *FlipDefs* family of dependency-based syntactic features. These features are monolingual (they don't depend on the Chinese counterparts of the English dependencies).

The following example (Figure 4.2) shows the motivation behind FlipDefs. In the first tree, the word "economic" depends on "achievement" whereas in the second one, it depends on "construction".

Clearly at most one of these dependencies can be correct given a Chinese sentence. In this example, the four reference translations into English are shown in Figure 4.3. Three of these four sentences contain the "$construction''$ → "$economic''$ dependency, the fourth one contains a different dependency, namely "$development''$ → "$economic''$ but none of the four contains a dependency "$economic''$ → "$construction''$.

```
(S1 (S (NP (CD fourteen)
           (ADJP (JJ chinese)
                 (JJ open))
           (NN border)
           (NNS cities))
       (VP (VBP make)
           (NP (JJ significant)
               (NNS achievements))
           (PP (IN in)
               (NP (JJ economic)
                   (NN construction))))))

(S1 (NP (NP (JJ significant)
            (NN accomplishment))
        (VP (VBN achieved)
            (PP (IN in)
                (NP (NP (DT the)
                        (JJ economic)
                        (NN construction))
                    (PP (IN of)
                        (NP (NP (DT the)
                                (CD fourteen)
                                (JJ open)
                                (NN border)
                                (NNS cities))
                            (PP (IN in)
                                (NP (NNP china)))))))))))

(S1 (S (PP (IN in)
           (NP (NNP china)))
       (, ,)
       (NP (NP (CD fourteen)
               (NNS cities))
           (PP (IN along)
               (NP (DT the)
                   (NN border))))
       (VP (VBN opened)
           (PP (TO to)
               (NP (NP (NNS foreigners))
                   (VP (VBN achieved)
                       (NP (JJ remarkable)
                           (JJ economic)
                           (NN development)))))))))

(S1 (S (NP (JJ economic)
           (NN construction)
           (NN achievement))
       (VP (AUX is)
           (ADJP (JJ prominent)
                 (PP (IN in)
                     (S (NP (NP (NP (NNP china)
                                    (POS 's))
                                (NP (CD fourteen)
                                    (NN border)))
                            (VP (VBG opening)
                                (PRT (RP up))
                                (NP (NNS cities)))))))))))
```

Figure 4.4: Parsed reference sentences.

```
(S1 (S (NP (NP (NNP china)
               (POS 's))
           (CD 14)
           (ADJP (JJ open))
           (NN border)
           (NNS cities))
       (VP (VBD marked)
           (NP (JJ economic)
               (NNS achievements)))))
```

Figure 4.5: Parsed produced sentence.

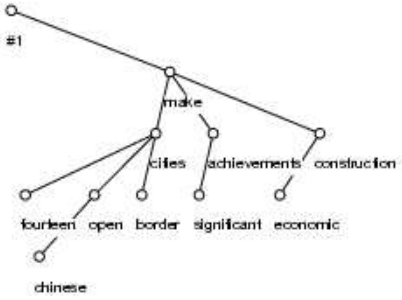



Figure 4.6: Reference 0.

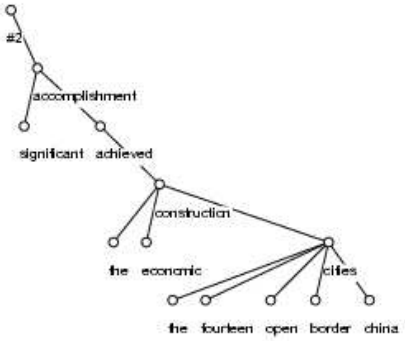



Figure 4.7: Reference 1.

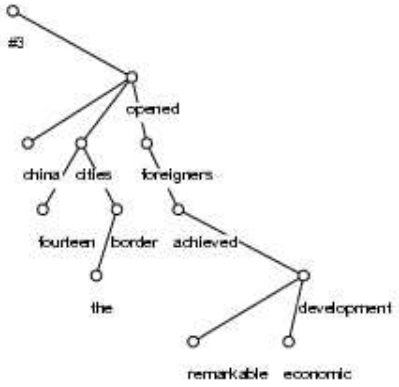



Figure 4.8: Reference 2.

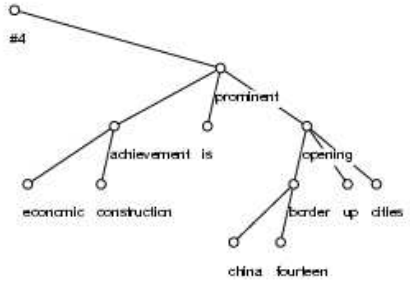



Figure 4.9: Reference 3.

## Idea

The main idea is to penalize NBEST sentences that have too many *flipped* dependencies. Figure 4.10 illustrates how the *FlipDefs* feature is computed. The first candidate gets penalized for its incorrect use of the "$achievement''$ $\rightarrow$ "$economic''$ dependency.

## Implementation

A large amount of preprocessing was needed to compute the *FlipDefs* features. The following steps were needed, in order.

1. Preprocess the input text
2. Parse with the Collins/Charniak parsers
3. Extract heads using code from Jason Eisner
4. Lemmatize and convert the resulting structures into the Prague dependency format using tools provided by Martin Čmejrek, Jan Hajič, Ivona Kučerová, and Zdeněk Žabokrtský
5. Extract all dependencies

```
PRODUCED: china 's 14 open border cities marked economic achievements

mark->city
city->china
city->14
city->open
city->border
mark->achievement
achievement->economic

score: 2.3219 - rank 493

ORACLE: china 's 14 open border cities in economic construction made
significant achievements

make->city
city->china
city->14
city->open
city->border
city->construction
construction->economic
make->achievement
achievement->significant

score: 2.4404 - rank 327
```

Figure 4.10: Example features for two of the NBEST translations of sentence 0 from the dev set.

```
0. china 's 14 open border cities marked economic achievements
1. china 's 14 open border cities achievements remarkable
2. china 's 14 open border cities building remarkable achievements
3. china 's 14 open border cities , remarkable achievements
4. china 's 14 open border cities construction remarkable achievements
5. china 's 14 open border cities achievements marked
6. china 's 14 open border cities achievements significant
7. china 's 14 open border cities economic achievements remarkable
8. china 's 14 open border cities economic remarkable achievements
9. china 's 14 open border cities economic construction remarkable achievements
```

Figure 4.11: The first ten NBEST translations of sentence 0 in the dev set.

```
0. mark->city city->china city->14 city->open city->border
   mark->achievement achievement->economic
1. achievement->city city->china city->14 city->open city->border
   achievement->remarkable
2. city->china city->14 city->open city->border city->build
   build->achievement achievement->remarkable
3. city->china city->14 city->open city->border city->achievement
   achievement->remarkable
4. achievement->city city->china city->14 city->open city->border
   achievement->construction achievement->remarkable
5. mark->city city->china city->14 city->open city->border
   mark->achievement
6. achievement->city city->china city->14 city->open city->border
   achievement->significant
7. city->china city->14 city->open city->border city->remarkable
   remarkable->achievement achievement->economic
8. economic->city city->china city->14 city->open city->border
   economic->achievement achievement->remarkable
9. achievement->city city->china city->14 city->open city->border
   city->economic achievement->construction achievement->remarkable
```

Figure 4.12: Dependency representations of these sentences.

6. Build a FlipDeps model from the training/dev data

7. Run the FlipDefs model on the test data

Figure 4.11 shows the first ten NBEST translations for sentence 0 of the dev set. Figure 4.12 shows the same sentences in dependency format:

## The FlipDefs model

We define the "small" dependency feature $d(w_1, w_2)$ to be:

$$d(w_1, w_2) = \log \frac{p(w_1 \rightarrow w_2)}{p(w_2 \rightarrow w_1)}$$

| $w_1$ | $w_2$ | $ct(w_1)$ | $ct(w_1) + ct(w_2)$ | $\frac{ct(w_1)}{ct(w_1)+ct(w_2)}$ |
|---|---|---|---|---|
| china | have | 74 | 77 | 0.9610 |
| china | training | 4 | 5 | 0.8000 |
| china | room | 9 | 12 | 0.7500 |
| china | become | 2 | 3 | 0.6667 |
| china | it | 2 | 4 | 0.5000 |
| china | present | 5 | 11 | 0.4545 |
| china | Japan | 8 | 18 | 0.4444 |
| china | billion | 4 | 9 | 0.4444 |
| china | Bank | 3 | 7 | 0.4286 |
| china | fund | 10 | 28 | 0.3571 |
| china | loan | 6 | 20 | 0.3000 |
| china | trade | 62 | 317 | 0.1956 |
| china | be | 9 | 47 | 0.1915 |
| china | production | 1 | 20 | 0.0500 |
| china | Dingha | 1 | 26 | 0.0385 |
| china | use | 14 | 737 | 0.0190 |
| china | growth | 5 | 323 | 0.0155 |
| china | enterprise | 9 | 588 | 0.0153 |
| china | and | 8 | 3745 | 0.0021 |

Table 4.6: Sample values of the dependency feature for dependencies including the word “china”.

For example:

- $f(\text{"}mark\text{"} \rightarrow \text{"}achievement\text{"}) = 8$
- $f(\text{"}achievement\text{"} \rightarrow \text{"}mark\text{"}) = 4$
- $d(\text{"}mark\text{"}, \text{"}achievement\text{"}) = \log(8/4) = \log 2$

Similarly, the “big” dependency feature is:

$$D = \sum_i d_i$$

Table 4.6 shows some representative examples of dependencies.

There are four features in the *FlipDefs* family: *FlipDefs1*, *FlipDefs2*, *FlipDefs3*, and *FlipDefs4*. The last two are exprimental. The rest (FlipDefs1 and FlipDefs2) will now be described:

- FlipDefs1 - using only the first 100,000 sentences from the dev-set as training.
- FlipDefs2 - using the entire dev-set (990,952 sentences) for training.

Let's now consider the 1000 NBEST translations for sentence 0. The value of FlipDefs1 for them ranges from 2.9069 to -2.6630.

### Results

| | 20 iterations, BLEU (%) | 50 iterations, BLEU (%) |
|---|---|---|
| FlipDeps1 | 31.6234 | 31.5735 |
| FlipDeps2 | 31.6328 | 31.7700 |

### Future ideas

- In these experiments, we haven't done any backoff for rare dependencies. Backoff is likely to help.
- Using transitive dependencies seems to be an idea worth trying.
- We should also investigate different functions $f$ for $D = f(d_i)$.

## 4.7 Word Popularity

### Motivation

It is known that human translations of the same input sentence can be very different from one another. This is one of the motivations behind the BLEU evaluation (Papineni et al., 2001). The idea behind the *Flipped Dependencies* family of features is also motivated by the diversity of human translations.

Translation models are traditionally trained from a single reference translation. As a result, they are likely to underestimate the probabilities of certain less frequent translations of a word while overemphasizing the single translation present in the given reference translation.

Figure 4.3 shows why this is the case. If only the first human translation were used for training, the translation model would learn an inappropriately high probability for the translation "achievements" while discounting the fact that other valid translation of the same Chinese word may be "accomplishment" or "achieve"/.

If one were to train on all four reference translations, then the relative probabilities of the different translations would more accurately reflect the tendency of humans to use them.

## Idea

To exploit multiple translations of the same sentence, we built several models of *Word Popularity* based on the development set. We didn't have access to multiple translations of the training set.

Let's consider the four translations shown in Figure 4.3. We can build several equivalence classes of words, e.g., (fourteen); (chinese, china); (open, opened, opening); (achievements, accomplishment, achievement, achieved); (construction, development), etc. A quick analysis reveals the existence of several types of words based on their presence in the multiple translations.

- very popular words (e.g., fourteen) - these words appear in all four references.
- alternatives based on syntax and morphology (e.g., open, opened, opening).
- synonyms or near synonyms (e.g., construction, development)
- idiosyncratic cases or words that appear only in a small percentage of the translations (e.g., along).

## Implementation

We trained a WordPop family of models using the development set.

IDF (inverse document frequency) is a measure borrowed from Information Retrieval which measures how frequent a word is in a large corpus. Low-content words such as prepositions and articles have low IDF while rare words have very high IDF values. IDF is measured on a logarithmic scale so a word with an IDF of 10 is $2^{10}$ times as rare as a word with an IDF of 0.

The IDF values for our experiments were computed from a training corpus of TDT documents (Radev, Hatzivassiloglou, and McKeown, 1999). Some

typical values of IDF are “the” (0.010), “of” (0.110), “open” (2.388), “China” (2.391), “along” (3.006), “achievement” (4.381), “cities” (4.528), and “achievement” (4.993). Rare words include “oratorios” (8.420), “physiological” (8.707), and “textures” (9.806).

The WordPop model assigns a given word a score roughly proportional to the certainty with which it is used in a set of four translations. If a word appears in all four translations, its score is 1.0. If it appears in 3 or 2 of them, then its score is 0.75 and 0.5, respectively. Finally, if it only appears in one, we only assign it a score of 0.1 to reflect the possible randomness of the reason for its presence.

Example:

- economic = 1
- fourteen = 1
- border = 1
- significant, construction = 0.75; development = 0.1
- china = .75, chinese = .1
- open = .5, opened = .1, opening = .1
- achieved = .5, achievement = .1, achievements = .1, made = .1

The full set of *WordPop* models is described here:

- *WordPop*
- *WordPopIdf* - same as WordPop but the score for each word is first multiplied by its IDF.
- *WordPop2Idf* - same as WordPopIdf but the WordPop score is given a weight of two.
- *WordPop3Idf* - same as WordPopIdf but the WordPop score is given a weight of three.
- *WordPopIdf2* - same as WordPopIdf but the IDF score is given a weight of two.

| parser | total nb sentences | failed to parse | parsed to S | parsed to NP | parsed to other |
|---|---|---|---|---|---|
| Collins | 990952 | 30761 | 811723 | 85577 | 62891 |
| Charniak | 990952 | 0 | 887202 | 41745 | 62005 |

Table 4.7: Statistics about the Collins and Charniak parsers on the development set. Sentences parsed to constituents other than "S" or "NP" typically parse to "FRAG", "SINV", "X".

### Results

| | 20 iterations, BLEU (%) | 50 iterations, BLEU (%) |
|---|---|---|
| WordPop | 31.5980 | 31.6675 |
| WordPop-Idf | 31.8329 | 31.4092 |
| WordPop2-Idf | 31.7577 | 31.7577 |
| WordPop3-Idf | 31.0741 | 31.8826 |
| WordPop-Idf2 | 31.4164 | 31.5971 |

## 4.8 CharColl

### Motivation

A cursory analysis of the NBEST list shows that a large number of them are not grammatical sentences. In other words, state of the art parsers like Collins's and Charniak's are likely to encounter problems. As Section (XXXX) reports, more than 3% of the sentences cannot be parsed by the Collins parser at all. The Charniak parser parses all sentences however often both parsers fail to parse them all to way to an S and instead parse the entire NBEST input to a lower-level constituent such as NP (See Table 4.7)

### Example

Figures 4.13 and 4.14 illustrate how the distance function is computed. The Charniak parser uses the S1 symbol to denote the root of the parse tree, while Collins uses TOP. Other than that, the distance metric captures all other differences between the rest of the parse trees (or rather, parse strings). In the first example below, Charniak "discovers" an Adjective Phrase (ADVP) on top of the adjective "open" whereas Collins doesn't. The overall distance for the first example is therefore rather small (12). The second example indicates a larger disagreement. First, Charniak parses the input as a sentence fragment (FRAG) while Collins

```
(S1 (S (NP (NP (NNP china) (POS 's)) (CD 14) (ADJP (JJ open)) (NN
border) (NNS cities)) (VP (VBD marked) (NP (JJ economic) (NNS
achievements)))))

(TOP (S (NP (NP (NNP China) (POS 's)) (CD 14) (JJ open) (NN border)
(NNS cities)) (VP (VBD marked) (NP (JJ economic) (NNS achievements)))))
```

Figure 4.13: Sample parses of the same sentence by Charniak (top) and Collins (bottom) (dev set sentence number 1) - the edit distance is 12.

```
(S1 (FRAG (NP (NP (DT the) (JJ first) (CD two) (NNS months)) (PP (IN
of) (NP (DT this) (NN year)))) (ADJP (NNP guangdong) (JJ high)) (: -)
(NP (NP (NNP tech) (NNS products)) (NP (QP (CD 3.76) (CD billion)) (NP
(PRP us)) (NNS dollars)))))

(TOP (NP (NP (DT the) (JJ first) (CD two) (NNS months)) (PP (IN of)
(NP (QP (ADVP (NP (DT this) (NN year)) (IN Guangdong) (NP (JJ
high-tech) (NNS products))) (CD 3.76) (CD billion)) (NNP US) (NNS
dollars)))))
```

Figure 4.14: Sample parses of the same sentence by Charniak (top) and Collins (bottom) (test set sentence number 1) - the edit distance is 74.

identifies an NP. Next, the sequence "this year" is parsed quite differently by the two parsers. Finally, the sequence "3.76 Billion U.S. dollars" is also parsed in two very different ways. The overall distance between these two parse strings is 74.

## Implementation

We decided to use the Levenshtein edit distance at the string level to determine the discrepancy between the two parses. The Levenshtein distance is equal to the number of edits (insertions, deletions, and substitutions, all with the same weight) needed to transform one string into another.

To run the Charniak parser, we used the unix command "ulimit -s unlimited" to set the stack size to unlimited. Not doing that would cause the Charniak parser to fail approximately 1-2% of the time. With unlimited stack size, none of the approximately 2 Million sentences in dev-test and test failed to parse. This is in contrast the Collins parser falied completely on 30,761 out of 990,952 sentences.

We report results for the following features:

- CharColl - the Levenshtein edit distance between the two parses represented as strings.

| Situation | Count |
|---|---|
| Collins fails | 30,761 |
| Collins succeeds and $d > 100$ | 175,307 |
| Collins succeeds and $d > 50$ | 579,912 |
| Collins succeeds and $d > 25$ | 852,497 |

Table 4.8: Histogram of the edit distances between Collins and Charniak (dev set only).

- CollFails - this feature is equal to 1 if the Collins parser fails to parse a given sentence and -1 otherwise.
- CharColl2 - if Collins fails, this feature is equal to 10,000, otherwise it is equal to CharColl.

We looked at the distribution of edit distances for the sentences on which Collins didn't fail.

### Results

| | 20 iterations, BLEU (%) | 50 iterations, BLEU (%) |
|---|---|---|
| CharColl | 31.2004 | 31.5103 |
| CollFails | 31.6369 | 31.6130 |
| CharColl2 | 31.2482 | 31.2482 |

### Conclusion

Our results indicate that the edit distance is a reasonable feature, however the simpler "Does the Collins parser fail" on a given input is a much better predictor of a poor translation.

## 4.9 Some Basic Grammar Feature Functions

### Motivation

Looking at the Contrastive Error Analysis, we can see that there are often basic grammar-related mistakes in the original output, the so-called "stupid mistakes." For example, there are sometimes independent sentences that are not headed by verbs. Or there may be an adjective standing in a noun's position. There may be a

noun phrase on one side of a conjunction word with an adverb phrase on the other side. To reduce such kind of stupid mistakes, we developed a few feature functions that penalize the occurrences of some specific mistakes in the hypotheses.

## Ideas

Chinese grammar is more difficult to pin down than English grammar. For instance, there are no morphologies in Chinese, and switches between nouns, adjectives, and verbs happen so often that the Chinese parser broke down frequently. Therefore, we relied solely on the English side here. Dependency trees gave us not only parsing information but also the headwords of all constituents. Therefore, we extracted the POS information as well as the word-hierarchy structure of each hypothesis. The POS trees made it easier to detect some specific grammatical mistakes. Feature functions were then developed to penalize each occurrence of those mistakes.

## Implementation

1. $$h\_Verb\_headed(e) = \begin{cases} -1 & \text{if the sentence is not headed by a verb;} \\ 0 & \text{otherwise.} \end{cases}$$

2. h_Match_Conn(e) = the number of occurrences of conjunction words that are not balanced on both sides.
   Balance is determined by whether within the boundary of a constituent, the sections before and after a conjuction word end with the same or similar POS tags. For example, both NN-NN and NN-NNS pairings are regarded as balanced.

3. h_Wrong_PP_Position(e) = the number of occurrences of prepositional phrases that appear at wrong positions.
   Being wrong means that a prepositional phrase occurs neither at the beginning nor at the end of a constituent, or that a prepositional phrase isn't headed by a prepositional word.

4. h_Wrong_CD_Position(e) = the number of occurrences of numerical words that appear at wrong positions.

Being wrong means that a numerical word is preceded by a noun and followed by a punctuation. Most mistakes of this kind are due to the wrong translation of classifier words in Chinese.

5. h_Wrong_JJ_Position(e) = the number of occurrences of adjectives that appear at wrong positions.
   Being wrong means that an adjective appears at the end of a constituent excluding the situations of an adjective phrase and 'the + JJ' structure. This feature function is especially targeted at reordering mistakes between nouns and adjectives.

6. h_Wrong_NN_Position(e) = the number of occurrences of nouns that appear at wrong positions.
   Being wrong means that a plural noun is followed by a word other than a conjunction word or punctuation within the boundary of its constituent. This feature function penalizes not only reordering mistakes between adjectives and plural nouns, but also the mistake that a descriptive noun appears in its plural form.

## Results

| | BLEU Score |
|---|---|
| Baseline | 31.6 |
| Verb_headed | 31.6 |
| Match_Conn | 31.4 |
| Wrong_PP_Position | 31.7 |
| Wrong_CD_Position | 31.7 |
| Wrong_JJ_Position | 31.6 |
| Wrong_NN_Position | 31.8 |

## Analysis

The above six feature functions are targeted at specific mistakes made by the original system. They are based on basic English grammar. The improvements that they bring about are not well reflected by the BLEU score. On one hand, this implies that these syntactic features are only fine-tuning the output results instead of providing profound improvements to the system. On the other hand, it shows that BLEU is not adept at capturing these fine tunes. A limitation on these feature functions is that they don't give any consideration to the original Chinese

sentences. It is possible that a translation is absurdly wrong while grammatically right. In that case, these features are futile.

## 4.10 Projecting Dependencies

### Motivation and Idea

It has been observed that a syntactic dependency in one sentence more often than not corresponds to a syntactic dependency in a translation of that sentence. In (Hwa et al., 2002), about 70% of the unlabeled dependencies in a English sentence mapped to a dependency in a corresponding Chinese sentence. The projection rate of *labeled* dependencies must of course be lower.

The simplest dependency projection feature functions count the number of source dependencies for which we can find a corresponding dependency in the target (figure 4.15). Alternately, we can count the number of source dependencies that do not project to the target. Finally, we can model the probability that a dependency will project, conditioned on the type of Chinese dependency under consideration.

We have another degree of freedom in the types of dependency relations that we will consider valid for projection. Let $\rightarrow$ indicate the "depends on" relation and $\parallel$ indicate "aligns with." If $A \rightarrow B$ in Chinese and $A' \rightarrow C' \rightarrow B'$ in English; and $A \parallel A'$; and $B \parallel B'$; we may or may not wish to count the transitive dependency $A' \rightarrow B'$ as a valid projection of $A \rightarrow B$. Similarly, we can also consider whether transitive dependencies in the Chinese project onto the English.

### Implementation

Given word alignments between the source and target sentences and a set of syntactic dependencies in each source and target sentence, we can calculate the three types of features enumerated above: counting matching dependencies, counting broken dependencies, and calculating the likelihood of the dependencies projections.

In the third case, for calculating the conditional probability of a dependency projection, we used the MT system's training data and the single best word alignment induced during training by GIZA++. Conditioning on the parts of speech at each end of the Chinese dependency, we then determined the relative frequency of that part-of-speech pair's projecting successfully onto an English dependency. We

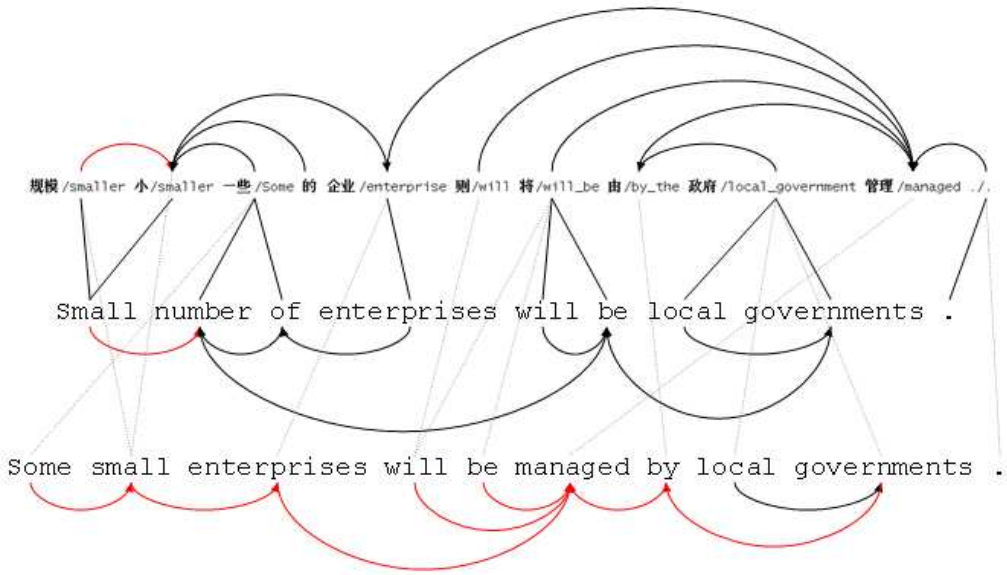



Figure 4.15: Projecting dependencies: the first sentence, originally produced by the system, had only one dependency in common with the Chinese; the oracle has seven.

could then calculate the likelihood of a source/translation pair under this model as the product of the one term per source sentence dependency. If the dependency did project, the term was the probability $p(\text{projection}|\text{POS tags})$ determined in training. If the dependency did not project, it was $1 - p$.

## Results

Results for various experimental conditions are shown below (table 4.9). Counting successful dependencies, with transitivity on the English side alone, achieved the best results.

## Analysis

Although some results are encouraging, they are not nearly as high as for some of the "implicit" syntactic features. It is also discouraging that the probability model of syntactic dependency does not outperform simple counting. Since we expect some dependencies not to project due to different linguistic structures in Chinese

| Method | Transitivity | | |
|---|---|---|---|
| | None | English | Both |
| Count projecting dependencies | 31.6 | **31.7** | 31.4 |
| Count non-projecting dependencies | 31.0 | 30.9 | 30.9 |
| Dependency projection likelihood | 31.6 | 31.5 | — |

Table 4.9: Dependency projection feature scores

and English, we do not want to count all dependencies equally.

We conditioned the probability of a dependency projection only on the parts of speech of the Chinese words in the dependency relation. Given a more sophisticated model of what projects from Chinese to English and what does not, we could expect to improve our performance. A bound may be placed on our performance in this direction, however: during the workshop, team member Kenji Yamada noted that the human authors of the reference translations would often preserve Chinese word order in English even at the expense of preserving exact syntactic relations.

Finally, as noted above, Chinese parsing achieves only about 82% parseval even with perfect segmentation. We might, therefore, find it helpful to incorporate into our models the probability of a dependency as assigned by the parser.

# Chapter 5

# Tricky Syntactic Feature Functions

## 5.1 Cross-lingual Constituent Alignment Features

### Motivation and Idea

Parse trees provide a rich source of linguistic information. By combining Chinese and English trees with word-level alignments provided by the translation system, we would like to introduce syntactic features that score candidate English translations. Hypotheses that syntactic differ from the source Chinese sentence should be penalized, while those translations with similar syntactic structure should be preferred.

### Implementation 1: Tree to String Penalty

For each constituent in the Chinese parse tree, the English words corresponding to the leafs (i.e., Chinese words) of the subtree rooted at the constituent are discovered via the alignments provided by the MT system. If there exists any words between the left-most and right-most aligned English word that was *not* aligned to any word within the Chinese subtree, then a penalty of ”1” is given for that constituent. In Figure 5.1, for example, the unaligned word ”marked” appears between the left-most aligned English word ”China’s” and right-most ”cities”. Thus, a penalty is assigned for the NP constituent that roots the shown Chinese subtree. The penalties are accrued for all Chinese parse constituents, and this sum is then provided as the feature. Note that this feature makes no use of the English parse tree.

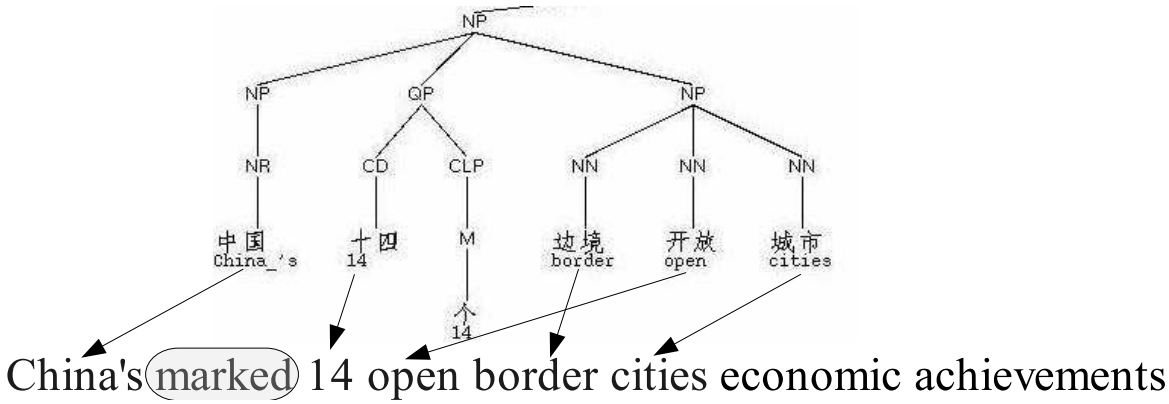



Figure 5.1: Tree to String Penalty

## Implementation 2: Tree to Tree Penalty

This feature is similar to the former Tree to String penalty, but additionally uses the English parse tree. After discovering the aligned English words from the leafs of a given Chinese parse subtree, the smallest English parse subtree that contains all the aligned words is computed. If there exist any leafs in that English subtree which is not aligned to a leaf of the original Chinese subtree, then a penalty of "1" is given for that Chinese constituent. In Figure 5.2, for example, the unaligned word "remarkable" appears in the English subtree which the words "open", "border", and "cities" are contained by, and thus a penalty is assigned for the shown Chinese subtree rooted by an NP node. The penalties are accrued for all Chinese parse constituents, and this sum is then provided as the feature.

## Implementation 3: Constituent Label Probability

This feature learns node label mapping probabilities from training data (via simple counting) and then applies the learned distributions to the ranking of candidate translations. Specifically, this feature learns the probability of a node label ("VP", "NP", etc) *Y* being the root of an English subtree that aligns to a Chinese subtree rooted by a node with label *X*:

$$p(\mathbf{englishlabel} = \mathbf{VP}|\mathbf{chineselabel} = \mathbf{NP}) = 0.019.$$

Alignments between parse constituents are not given, and thus are discovered by aligning leaf nodes (words) between subtrees, and then finding the minimum

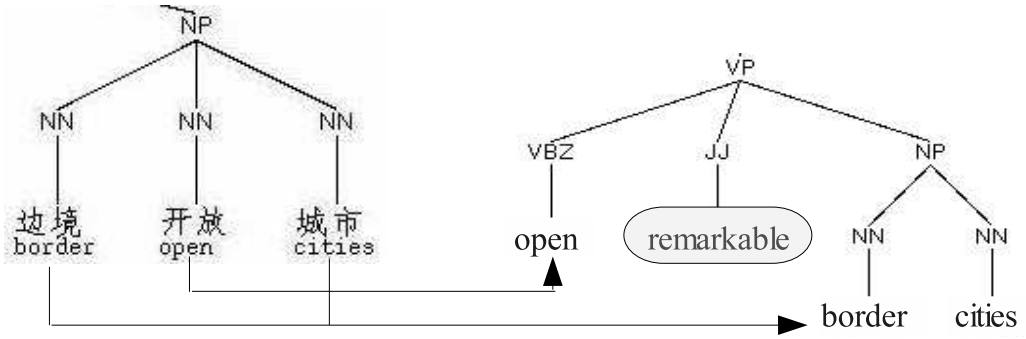



Figure 5.2: Tree to Tree Penalty

common ancestor within the parse tree of those leafs. The whole feature is given by the product of the probabilities of each English node label given its aligned Chinese subtree label.

The transition probabilities generated by the training procedure of this feature are themselves interesting, as they inform us as to whether the relationships between the two trees are consistent with linguistic intuition and, to a certain extent, the degree of noise in the parse information. Table 1 provides a sampling of this information, in which the left column denotes the Chinese source label, and the list of labels denotes English labels and their corresponding conditional probability. In addition to the label transition probability, a similar feature was implemented

| | | | | | |
|---|---|---|---|---|---|
| | $CC$ | $CONJP$ | $RB$ | $TO$ | $NN$ |
| CC | .845 | .023 | .02 | .026 | .008 |
| | $NP$ | $VP$ | $PP$ | $JJ$ | $NN$ |
| VP | .327 | .327 | .101 | .036 | .033 |
| | $NP$ | $VP$ | $S$ | $PP$ | $SBAR$ |
| IP | .48 | .172 | .20 | .047 | .026 |

Table 5.1: Chinese to English Node label transition probabilities

which calculated the probability of the number of leafs under a subtree with a given label, conditioned on the label of the source root node and the number of leafs in the source tree.

### Results

| | $BLEU$ |
|---|---|
| Baseline | 31.6 |
| Tree to String | 31.6 |
| Tree to Tree | 31.1 |
| Constituent Label Probability | 31.2 |
| Number of Leaf Nodes Probability | 31.7 |

Table 5.2: BLEU Scores of Constituent Alignment Features

### Analysis and Conclusion

The constituent alignment features seem to have little or no impact on the performance of the current system. A variety of reasons could account for this: noisy parses, imperfect BLEU measurement of subtle syntactic improvement, or noisy alignments. Mostly likely, it is a combination of all of these factors that at this time prevent such complicated features from being sufficiently robust for overall performance gains.

## 5.2 Additional Syntax-based Alignment Features

In this section we will consider various additional syntax-based alignment features. In contrast with the other tree-tree or tree-string alignment features discussed elsewhere in this report, in this section we will consider those syntax-based alignment features that use trees in their alignment, but do not align the full parse tree. The alignments take advantage of various parts of the parse tree, usually in an attempt to simplify the extensive computation required to align full parses.

The particular alignment strategies we consider in this section are:

- Parser probabilities over alignment template boundaries
- Hacking up the parse tree: a Markov assumption for tree alignments
- Using TAG elementary trees for scoring word alignments

### 5.2.1 Parser probabilities over alignment template boundaries

Let us consider a method that uses individual constituent probabilities from the parser. We use these constituent probabilities to score alignment template boundaries.

In this particular feature function definition, we only use the parse on the target side (English) to provide us with constituent probabilities. We use a statistical parser (the Collins parser) which provides us with a parse for the target sentence. In addition, from the Collins parser we obtain for each constituent that comprises the parse tree, the log probability of that constituent.

Alignment templates have been explained earlier. We can see an example of an alignment between source and target in Figure 5.3 that uses the alignment template approach. The black filled squares in the figure represent the word alignments, while the rectangles that enclose these filled squares are the alignment templates. Note that while the alignments inside an alignment templates have strong evidence from the training data, the boundaries between these alignment templates are not modelled as directly in the statistical MT model. The way that the MT model scores these alignment template boundaries is by using four probability models that are estimated from the MT training data. These two models are based on two concepts:

- *P(left-monotone)* or *P(right-monotone)*: this is the probability that in the training data a particular alignment template occurred with it's left/right boundary touching another alignment template.

- *P(left-continuous)* or *P(right-continuous)*: this is the probability that in the training data a particular alignment template occurred with it's left/right boundary touching another alignment template and in addition the word alignments at the boundary are contiguous in the parallel text.

The motivation for this feature function is to provide extra evidence for these alignment template boundaries using the constituent probabilities from the statistical parser. In addition to the four probability models described above, we add a new model *P(min-parser-prob)*. This model takes the minimum log probability from each of the constituents that span across the alignment template boundary and uses this probability as evidence for the boundary.

Figure 5.3 shows an example of how the probability for an NP constituent *P(min-parser-prob) = -20.56* is used as a score for the alignment template boundary. The feature function for each candidate sentence in the n-best list is computed

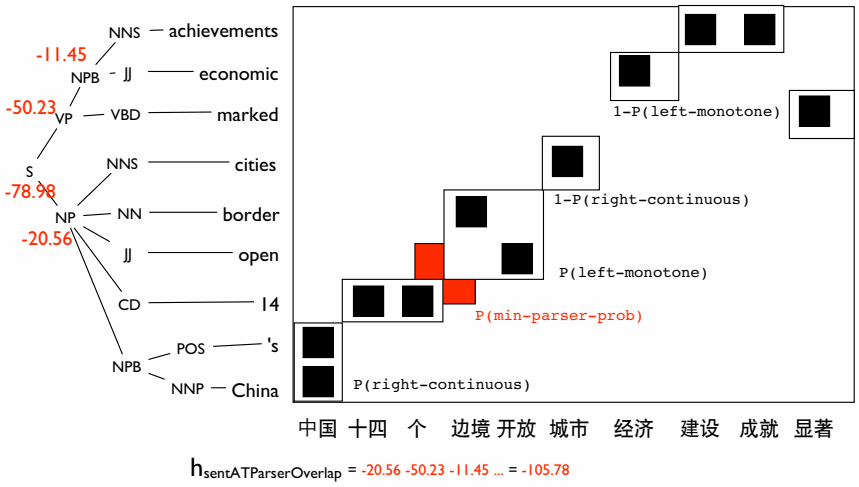



Figure 5.3: Feature function that uses the constituent log probabilities from a statistical parser to score the boundaries between alignment templates.

as a linear combination of the scores for each alignment template boundary. For $n$ alignment templates for a particular candidate the feature function is computed as follows:

$$h_{sentATParserOverlap} = P(\textit{min-parser-prob-AT}[1]) + \ldots + P(\textit{min-parser-prob-AT}[n-1])$$

### 5.2.2 A Markov assumption for tree alignments

In previous sections we have seen two different methods for computing translation models for machine translations that are constrained using parse trees. In the tree-to-tree translation model, both the source and target languages are parsed using a statistical parser, while in the tree-to-string translation model, the parse on the source side is used to translate to the target by using a model that produces a parse on the target side. In this section, we consider some computational limitations on these approaches and explore a solution that involves a Markov assumption for the tree-to-tree and tree-to-string translation models.

First, we describe some of the tractability considerations on the tree-to-tree and tree-to-string models. In particular, we consider two limitations:

- Full parse tree models are expensive to compute for long sentences and for trees with flat constituents

- There is limited reordering observed in the n-best lists that form the basis of our experiments. In addition to this, higher levels of parse tree rarely observed to be reordered between source and target parse trees.

In this section, we provide an algorithm for hacking full parse trees into tree fragments. These tree fragments are then aligned between target and source by exploiting the word alignments taken from the base machine translation model. These aligned tree fragments are then used in an unconstrained tree-to-tree or tree-to-string translation model. Using fragments rather than full parse trees avoids the problematic issues with these models that we mentioned above.

This approach provides a simple Markov model for tree-based alignments. It guarantees tractability. Compared to a coverage of approximately 30% of the n-best list by the unconstrained tree-based models, using the Markov model approach provides 98% coverage of the n-best list. In addition, this approach is robust to inaccurate parse trees (which are often produced since the candidate translations are often completely unlike the training data for the statistical parser).

The algorithm works as follows: we start with word alignments and two parameters: $n$ for maximum number of words in tree fragment and $k$ for maximum height of tree fragment. We proceed from left to right in the *source* sentence (Chinese, in our case) and incrementally grow a pair of subtrees, one subtree in Chinese and the other in English, such that each word in the Chinese subtree is aligned to a word in the English subtree. We grow this pair of subtrees until we can no longer grow either subtree without violating the two parameter values $n$ and $k$. Note that these aligned subtree pairs have properties similar to alignment templates. They can rearrange in complex ways between source and target.

For example, consider the sentence pair with word alignments shown in shown in Figure 5.4. Figure 5.4 shows how subtree-pairs for parameters $n = 3$ and $k = 3$ can be drawn for this sentence pair.

Once these subtree-pairs have been obtained, we can easily assert a Markov assumption for the tree-to-tree and tree-to-string translation models that exploits these pairings. Let consider a sentence pair in which we have discovered $n$ subtree-pairs which we can call $Frag_0, \ldots, Frag_n$. We can then compute a feature function for the sentence pair using the tree-to-string translation model as follows:

$$h_{MarkovTreeToString} = logP_{tree\text{-}to\text{-}string}(Frag_0) + \ldots + logP_{tree\text{-}to\text{-}string}(Frag_n)$$

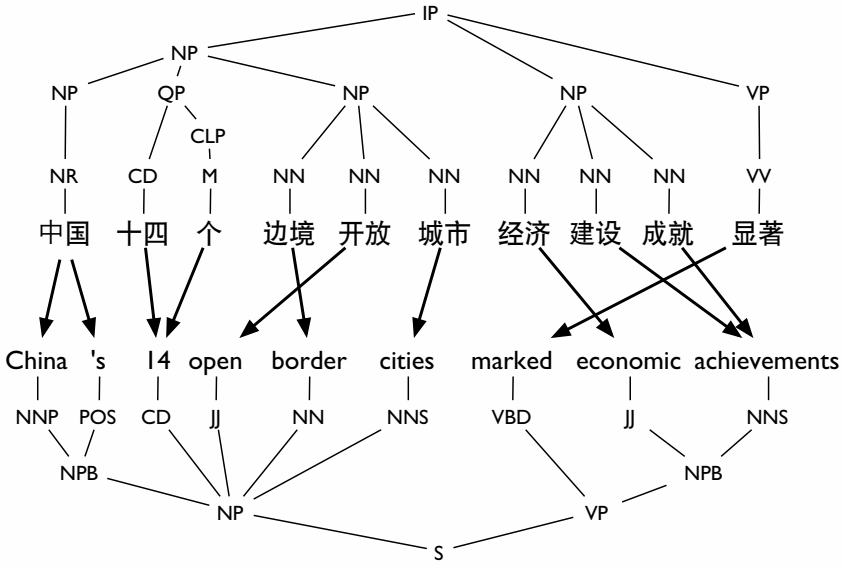



Figure 5.4: Overlay of the parse trees in the source and target languages showing the word alignments produced by the base MT system.

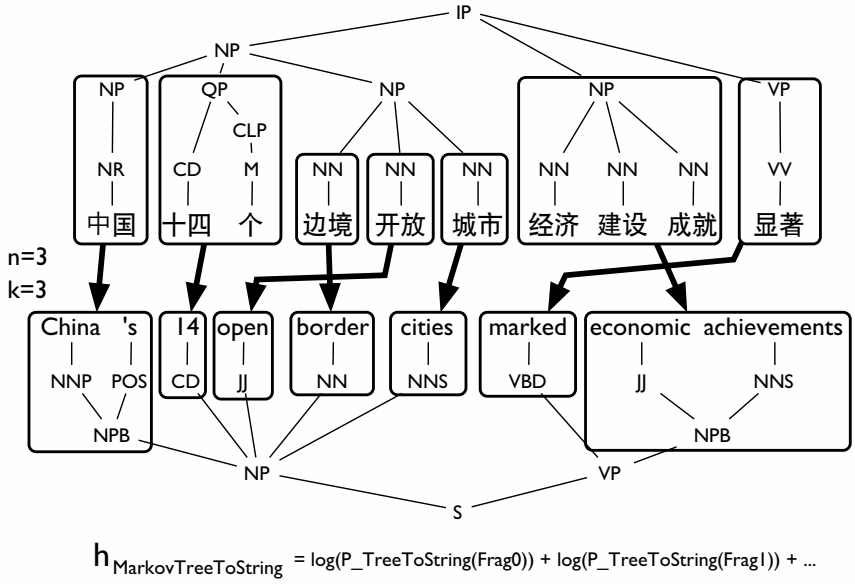



Figure 5.5: Result of carving up the tree using a Markov assumption over the size of sub-trees aligned with respect to the word alignments.

### 5.2.3 Using TAG elementary trees for scoring word alignments

In this section, we consider another method for carving up the full parse tree. However, in this method, instead of subtree-pairs we consider a decomposition of parse trees that provides each word with a fragment of the original parse tree. In addition, the decomposition provides a method of pasting each of these fragments to recover the original parse tree.

The formalism of Tree-Adjoining Grammar (TAG) provides the definition what each tree fragment should be and in addition how to decompose the original parse trees to provide the fragments. Each fragment is a TAG elementary tree and the composition of these TAG elementary trees in a TAG derivation tree provides the decomposition of the parse trees.

The decomposition into TAG elementary trees is done by augmenting the parse tree for source and target sentence with head-word and argument (or complement) information. This kind of information is added to the parse tree using standard heuristics. These heuristics are common to most contemporary statistical parsers and easily available for both English and Chinese.

The first step is adding head-word and argument information to the parse trees. Figure 5.6 shows what the parse trees look like after this information is added to the parse trees (the original trees are shown in Figure 5.4). Head non-terminals are marked with superscript $H$, while argument non-terminals are marked with superscript $A$. Each non-terminal in the parse tree has a unique distinguished child node which is marked as a head. In addition, some sister non-terminals of a head non-terminal node is marked as an argument non-terminal.

Removing all tree relationships that are not between head nodes, and duplicating argument nodes gives us the tree fragments shown in Figure 5.7. Drawing it slightly differently as shown in Figure 5.8 we can immediately observe that each word is assigned a single TAG elementary tree (the fragments we were after). Note that we do not use the word alignment information for the decomposition into TAG elementary trees.

Once we have a TAG elementary tree per word, we can create several models that score word alignments by exploiting the alignments between TAG elementary trees between source and target. Let $f_i$ be a word in the target sentence, let $e_i$ be a word in the source sentence aligned with $f_i$. In addition, let $t_{f_i}$ and $t_{e_i}$ be the TAG elementary trees associated with the words $f_i$ and $e_i$ respectively. We experimented with the following two models over alignments:

- Unigram model over alignments: $\prod_i P(f_i, t_{f_i}, e_i, t_{e_i})$

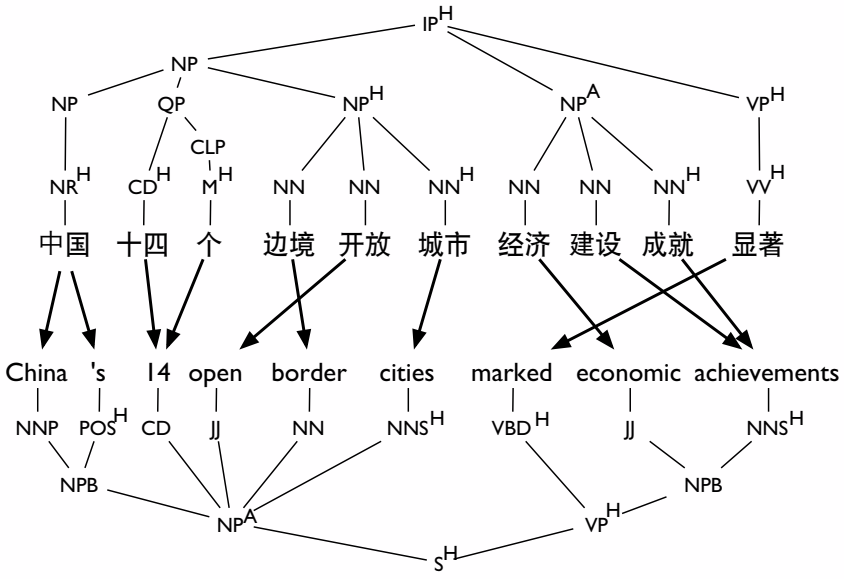



Figure 5.6: Marking head word and argument information on the non-terminals in the parse trees in source and target languages.

- Conditional model: $\prod_i P(e_i, t_{e_i} \mid f_i, t_{f_i}) \times P(f_{i+1}, t_{f_{i+1}} \mid f_i, t_{f_i})$
-

We trained both of these models using the SRI Language Modeling Toolkit using 60K aligned parse trees. We extracted 1300 TAG elementary trees each for Chinese and for English.

In addition, we also trained IBM Model 1 on aligned TAG elementary trees and the words.

### 5.2.4 Results

Figure 5.2.4 show the results for all the models described in this section.

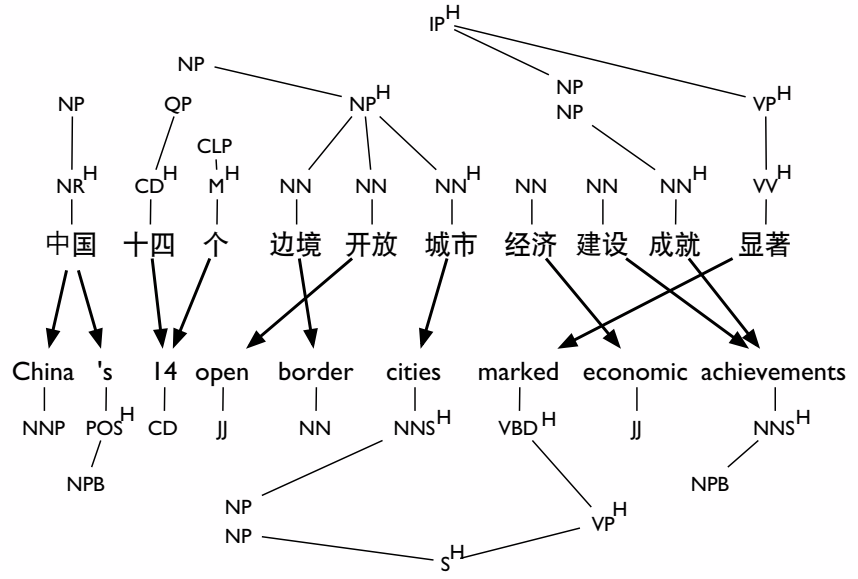



Figure 5.7: TAG elementary trees are defined by the connections between head and argument non-terminals.

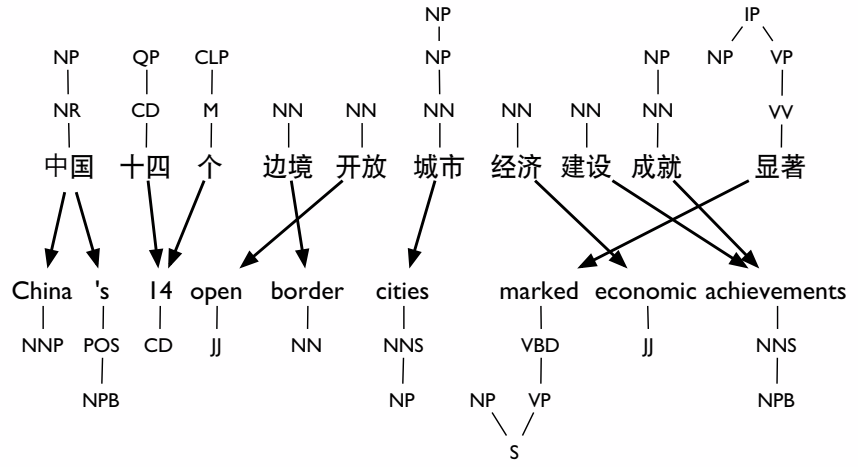



Figure 5.8: Word alignments associated with TAG elementary trees.

| Method | BLEU[%] |
|---|---|
| Baseline | 31.6 |
| AT Boundary Parser Prob | 31.7 |
| Tree-to-string without machete<br>(covers only 273/993 for dev, 237/878 for test)<br>Tree-to-string with machete | 31.7<br><br>32.0 |
| Model 1 on elementary trees | 31.6 |
| Unigram model over aligned elementary trees | 31.7 |
| Conditional bigram model over aligned elementary trees | 31.9 |

Figure 5.9: Results for additional syntax-based alignment features

# Chapter 6

# Reranking with Perceptron

## 6.1 Discriminative Reranking

In the recent years, discriminative reranking techniques have been successfully used in some natural language processing tasks, such as sentence parsing, POS tagging. Various learning algorithms have been employed in parse reranking, such as Boosting (Collins, 2000), Perceptron (Collins and Duffy, 2002), Support Vector Machines (Shen, Sarkar, and Joshi, 2003) and Log-linear models (Charniak, 2000; Collins, 2000). The use of the reranking techniques gives rise to a significant increase in labeled recall/precision over the previous best generative parsing systems.

There are several advantages of using discriminative reranking techniques. First, discriminative ranking enables us to use global features which are unavailable with the baseline system. Second, we can use features of various kinds and do not need to worry about fine grained smoothing issues. Finally, the statistical machine learning approach is theoretically well founded, and has been shown effective many applications.

## 6.2 Reranking for MT

Inspired by the works in parse reranking, we apply discriminative reranking to machine translation. We hope to improve the performance of MT systems by exploiting the advantages of the rerank techniques.

The procedure of MT reranking is similar to parse reranking. We first use a baseline MT system to generate N-best translations. Then, we analyze the struc-

tures of the source sentence and its translation by POS tagging, parsing and derivation tree extraction, and then extract features from these linguistic structures. Finally, we rerank the N-best translations with respect to these features.

For machine translation, two classes of features could be used to handle the language model and the translation model respectively. For the first class, we may use features extracted for produced translations, such as N-gram POS tags, NP chunks, segments of parse trees, and LTAG elementary trees. For the second class, we use pairs of aligned syntactic structures (or fragments of these structures) as features.

As far as machine translation is concerned, the reranking approach also helps to decrease the decoding complexity. The underlying generative system uses simple language models and alignment models. Then we extract syntactic structures from the source and target sentences, and we get the alignment on the deep syntactic structures. Thus, we decrease the computational complexity of tree alignment. Compared with some generative models for alignment, the reranking system is relatively easy to implement since the decoding on complex structures is avoided.

However, MT reranking is more complicated than parse reranking. In MT translation, there is no unique best parse for each sentence. Thus we cannot simply use the pairwise translations as samples, which was used in parse reranking. For each source sentence, we have several manually translated references. Similarity to these references are used to measure the quality of a translation.

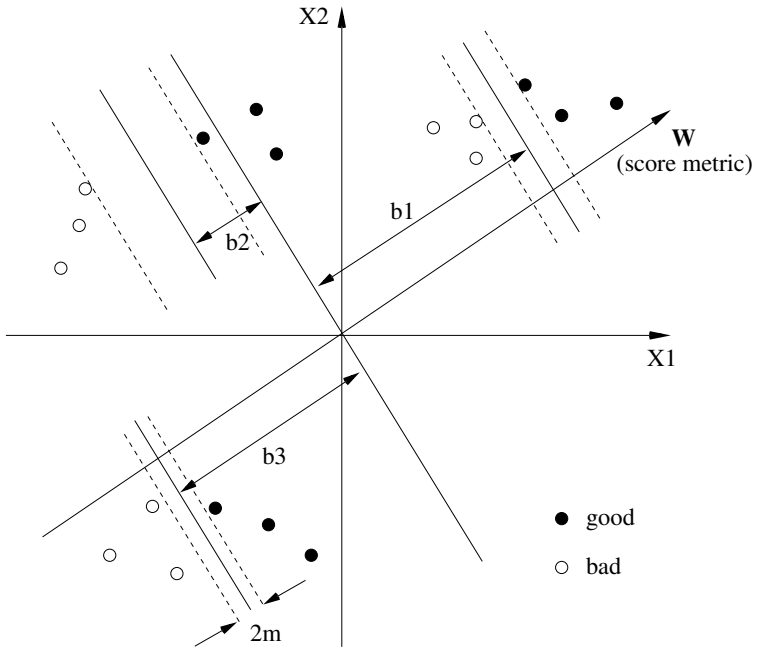
Another problem is that we don't have a reliable measure for the quality of a single translation. This is also the reason why BLEU and NIST scores are defined on the whole data set instead of each single sentence. Although, we can extend those measures to single sentences, such as the delta BLEU score that we have used, the score itself is not reliable at all; a translation with lower score could be better than a translation with a higher score.

Therefore, we cannot directly use those algorithms that have been used in parse reranking or other NLP tasks. We need to adapt those algorithms to the MT reranking task, considering the distinction of machine translation.

## 6.3 Multi-Bias Perceptron Algorithm

In this section, we will introduce a variant of the traditional perceptron algorithm, the Multi-Bias Perceptron with Margin, for the MT reranking task.

We cannot use the traditional perceptron to the reranking task directly since we need a unique *bias*. For example, there are three clusters in the following figure.




Distinct bias for each cluster

Each one contains several English translations for the same Chinese sentence. Black points are the good translations, and white points are the bad translations.

Obviously, it is impossible to find a hyperplane to successfully separate all the black points and white points. This is because translations for the same sentence are usually very similar in the feature space. Therefore we need a unique bias $b_i$ for each Chinese sentence $i$.

However, we can find three parallel hyperplanes to separate each cluster successfully. This means we have a global weight vector $\mathbf{w}$, which is expected to be in the same direction of the score metric. The resulting weight vector $\mathbf{w}$ can be used to rerank translations for a Chinese sentence.

The basic idea of the multi-bias perceptron is to train a distinct bias for each cluster, which all the separating hyperplanes are parallel by sharing the same weight vector $\mathbf{w}$. We assume that the weight vector is in the same direction of the score metrics, which means that the quality of a translation is determined by its distance to the separating hyperplane of the corresponding cluster.

There are two ways to handle the issue of multiple bias. One is to use pairwise translations as training samples, which is similar to approach to parse reranking. Since we don't have good standard translation for a sentence, how to define pair-

wise translations becomes a problem.

We must first define good translations and bad translations, which is still a big problem, due to the lack of a reliable measure for a single translation. In practice, we do like this. We first list the 1000 best translations according to the delta BLEU scores. Then we say the translations in the top one third are good translations, while those in the bottom one third are bad translations.

If we use a pair of a good translation and a bad translation as a sample, we will generate a huge number of related samples which are bad to any machine learning algorithm. So our solution is to take the other approach; we train a bias explicitly for all the translations of the same Chinese sentence. The resulting weight vector $\mathbf{w}$ is used as the score vector, but the biases are only used in training.

The following is the pseudo code of the multi-bias perceptron algorithm.

**Multi-Bias Perceptron with Margin**
**Input**: $(\mathbf{x}_{ij}, y_{ij})_{i=1..s,\ j=1..n}$, where $\mathbf{x}_{ij}$ is the $j$th English translation for Chinese sentence $i$, $y_{ij} \in \{1, -1\}$.
**Output**: $\mathbf{w}$
**Variables**: $b_1, b_2, ..., b_s$, biases for each Chinese sentence.
**Parameters:** $\tau, \eta$.

$\mathbf{w}^0 = 0; b_i^0 = 0; R = max_{\mathbf{x}_{ij}} ||\mathbf{x}_{ij}||;$
repeat
  foreach translation $\mathbf{x}_{ij}$
    if $y_{ij}(\mathbf{w}^t \cdot \mathbf{x}_{ij}) \leq \tau$
      $\mathbf{w}^{t+1} = \mathbf{w}^t + \eta y_{ij} \mathbf{x}_{ij};$
      $\underline{b_i^{t+1} = b_i^t + \eta y_{ij} R^2;}$
      $t = t + 1;$
    end if
  end for
until convergence

The line with underline is what is different from the traditional perceptron algorithm. In this algorithm, we only update the bias related to this translation. Now we will show that this algorithm converges after finite number of updating if the training samples are separable. We show it by modifying the proof for the perceptron with margin.

**Theorem 1** *The algorithm converges within at most* $2(s+1)((\frac{R}{\gamma})^2 + \frac{\tau}{\eta\gamma^2})$ *steps of updates, if each cluster of the training data is separable with margin* $\gamma$ *by a set of parallel hyperplanes defined on* $\mathbf{w}^*, b_1^*, ..., b_s^*$*, where* $||\mathbf{w}^*|| = 1,\ |b_i^*| \leq R$*.*

*Proof*: Suppose there are $s$ clusters and each cluster has $k$ samples. Let $\mathbf{I_i}$ = (0,...0,1,0...0) be a vector of $s$ elements, where the $i$th element is 1 and all others are 0. Through the proof we use the short-hand notations as follows.

$$\mathbf{z_{ij}} = (\mathbf{x_{ij}}', R\mathbf{I_i})',\ \mathbf{v} = (\mathbf{w}', \frac{1}{R}\mathbf{b}')'$$

Therefore there exist $\mathbf{w}^*$, $\mathbf{b}^*$ and margin $\gamma$ such that $||\mathbf{w}^*|| = 1,\ |b_i^*| \leq R$ and $y_{ij}(\mathbf{v}^* \cdot \mathbf{z_{ij}}) > \gamma > 0$, where $(\mathbf{x_{ij}}, y_{ij})$ is the $j$th sample of the $i$th cluster.

According to the algorithm, at the $t$th step of updating

$$\begin{aligned}
||\mathbf{v}^{t+1}||^2 &= ||\mathbf{v}^t + \eta y_{ij}\mathbf{z_{ij}}||^2 \\
&= ||\mathbf{v}^t||^2 + 2\eta y_{ij}(\mathbf{v}^t \cdot \mathbf{z_{ij}}) + ||\eta\mathbf{z_{ij}}||^2 \\
&\leq ||\mathbf{v}^t||^2 + 2\eta\tau + ||\eta\mathbf{z_{ij}}||^2 \\
&= ||\mathbf{v}^t||^2 + 2\eta\tau + ||\eta\mathbf{x_{ij}}||^2 + ||\eta R\mathbf{I_i}||^2 \\
&\leq ||\mathbf{v}^t||^2 + 2\eta\tau + 2\eta^2 R^2
\end{aligned}$$

Therefore,

$$||\mathbf{v}^T||^2 \leq 2T(\eta\tau + \eta^2 R^2). \tag{6.1}$$

On the other hand,

$$\begin{aligned}
\mathbf{v}^* \cdot \mathbf{v}^{t+1} &= \mathbf{v}^* \cdot \mathbf{v}^t + \eta y_{ij}(\mathbf{v}^* \cdot \mathbf{z_{ij}}) \\
&> \mathbf{v}^* \cdot \mathbf{v}^t + \eta\gamma \\
\mathbf{v}^* \cdot \mathbf{v}^T &> T\eta\gamma
\end{aligned} \tag{6.2}$$

Combining (6.1) and (6.2), we have

$$\begin{aligned}
T^2\eta^2\gamma^2 &< (\mathbf{v}^* \cdot \mathbf{v}^T)^2 \\
&\leq ||\mathbf{v}^*||^2||\mathbf{v}^T||^2 \\
&\leq (s+1)||\mathbf{v}^T||^2 \\
&\leq (s+1) * 2T(\eta\tau + \eta^2 R^2)
\end{aligned} \tag{6.3}$$

Therefore $T \leq 2(s+1)((\frac{R}{\gamma})^2 + \frac{\tau}{\eta\gamma^2})$, which means there are at most that much updates in the algorithm and then it converges. □

## 6.4 Experiments

We have designed two sets of experiments by using two different kinds of features. We first use the comprehensive features that we have described in the previous chapters. Then we use a rich feature set by using individual structures as features.

Our first experiment is on the 12 baseline features. The training data for our perceptron algorithm contains 993 Chinese sentences. Each Chinese sentence has 1000 best translations generated by Och's system. We list the 1000 best translations with respect to delta BLEU scores. Then the top 30% of the translations are used as positive samples, and the bottom 30% of the translations are used as negative samples. The test data contains 878 Chnese sentences. Each one has four reference translations. The BLEU score on the test set is used for evaluation.

Then we use 12 baseline features plus 30 features that we have developed in the workshop, such as the features defined on IBM model 1, POS language model, etc. The train data and test data are the same as our experiment on the baseline features.

By using the baseline features, our algorithm achieves a BLEU of 30.9%. With the 30 extra features included, it achieves a BLEU score of 31.6%.

In the second group of experiments, we use individual structures as features. We first use aligned POS-tag sequences as features.

For each pair of the aligned templates, we first replace all the words with POS-tags. Then the POS-tag sequence on the Chinese side is used as a feature, and the sequence on the English is used as a feature too. Furthermore, pair of the corresponding sequences is also used as a features. Thus, each translation is represented by a vector defined on those POS-tag based features. The value of a feature is 1 if the corresponding POS-tag sequence or pair of sequences appears in this samples. Otherwise, the value of the feature is set to 0. We say the feature is active in this sample if its value is 1.

The feature space of the POS-tag features is about 30,000. However, the feature space is very sparse. There are about 31 active features for each sample in average. We use the same training and test data set as the previous two experiments. The BLEU score on the training set is 34.2%, and the BLEU score on the test set is 30.9%.

Our next experiment uses fragments in translation parse trees as features. For each translation, we first parse it with Collins' parser. Then we use subtrees in its parse tree as the features for that translation. We use the following subtrees in our experiment.

- Rules used in derivation, which is equal to (parent node, all child nodes) structure.
- (parent node, two adjacent child nodes)
- (parent node, three adjacent child nodes)

By parent node we mean the POS tag of the parent node. So do child node.

The feature space of the subtree features is about 65,000. There are about 100 active features for each sample in average. The BLEU score on the training set is 30.3%, and the BLEU score on the test set is 30.5%.

## 6.5 Analysis

Experiments on comprehensive features show slight improvement in BLEU score when more useful features are incorporated, however the improvement is not significant. We think this is due to the nature of the data set. With 1000 translations for each Chinese sentence, the reranking task here is more like a regression problem instead of a classification problem. Furthermore, the feature space is very limited so that the training data are inseparable by a linear classifier.

We know discriminative machine learning algorithm can be used a very high space, with which many useful individual structures can be used as features. This is the reason why we have designed the second group of features. However, since the training data contains only 997 Chinese features, we cannot use some useful features such as lexicalized structures. But the preliminary results are still convincing. The result of using only the POS-tags features is already as good as the well optimized baseline system.

We notice that features defined on aligned structures are more useful than features defined only on the translations, as they are shown in the second group of experiments. This to some extent explains why log-likelihood of translation parse is not a good feature in our previous experiments.

Although the performance of the reranking result of the perceptron algorithm is not convincing, this approach is still attractive. We need further our research in adapting discriminative algorithm to MT reranking. The Multi-Bias perceptron algorithm presented here is our first step in this direction. On the other hand, we will also explore the possibility of working a data set of different style, i.e. 100,000 Chinese sentences with 20 translations for each one, so that discriminative machine learning algorithms become more useful.

# Chapter 7

# Minimum Bayes Risk Search

## 7.1 Introduction

Statistical MT systems are being applied to a wide range of information processing tasks such as information retrieval from text archives in a foreign language or for speech-to-speech translation. In these applications, the overall system performance is not measured by the accuracy of the automatic translation, but through task dependent evaluation criteria. Furthermore, we might find applications in which the MT evaluation metric incorporates syntactic structure. Given that different performance metrics are used for different applications, it is useful to create MT systems tuned with respect to each individual criterion. In contrast, the maximum likelihood techniques that underlie the decision processes of most current MT systems do not take into account these application specific goals. *Minimum Bayes-Risk (MBR) Classification* is a promising approach in this direction, and allows building of automatic MT systems tuned for specific tasks. We will show how syntactic structure obtained from parse-trees of the source and target sentences, can be incorporated into statistical MT framework using MBR classifiers.

We first introduce the MBR framework in the context of automatic speech recognition (Goel and Byrne, 2000). We then present MBR classifiers for machine translation, and describe a hierarchy of translation loss functions that are based on different levels of syntactic information. We finally present the performance of MBR classifiers optimized for each loss function.

## 7.2 Minimum Bayes-Risk Classifiers

We now introduce the Minimum Bayes-Risk Classification framework in relation to Automatic Speech Recognition.

In ASR, an acoustic observation sequence $A = a_1, a_2, ..., a_T$ is to be mapped to a word string $W = w_1, w_2, ..., w_N$, where $w_i$ are words belonging to a vocabulary $\mathcal{V}$.

We assume that a language is known; it is a subset of the set of all word strings over $\mathcal{V}$. This language specifies the word strings that could have produced any acoustic data seen by the ASR system. We further assume that the ASR classifier selects its hypothesis from a set $\mathcal{W}$ of word strings, that forms a subset of the language. The ASR classifier can then be described as the functional mapping $\delta(A) : \mathcal{A} \rightarrow \mathcal{W}$.

Let $L(W, W')$ be a real valued loss function that describes the cost incurred when an utterance $W$ belonging to language $\mathcal{W}$ is mistranscribed as $W' \in \mathcal{W}$. An example loss function is Levenshtein distance (Levenshtein, 1965) which measures the minimum string edit distance (word error rate) between $W$ and $W'$. This loss function is defined as the minimum number of substitutions, insertions and deletions needed to transform one word string into another.

Suppose the true distribution $P(W, A)$ of speech and language is known. It would then be possible to measure the performance of a classifier $\delta$ as $R(\delta(A)) = E_{P(W,A)}[L(W, \delta(A))]$. This is the expected loss when $\delta(A)$ is used as the classification rule for data generated under $P(W, A)$. Given a loss function and a distribution, the classification rule that minimizes $R(\delta(A))$ is given by (Bickel and Doksum, 1977)

$$\delta_R(A) = \underset{W' \in \mathcal{W}}{\operatorname{argmin}} \sum_{W \in \mathcal{W}} L(W, W')P(W|A). \tag{7.1}$$

We shall refer to the sum $\sum_{W \in \mathcal{W}} L(W, W')P(W|A)$ in Equation 7.1 as *conditional risk* and classifier given by this equation as the *Minimum Bayes-Risk* (MBR) classifier.

Our treatment so far assumes that the true distribution $P(W|A)$ is available, however this is not the case in practice. This distribution is obtained by applying Bayes rule $P(W|A) = P(W)P(A|W)/P(A)$, where the component distributions are approximated by models. As is commonly done, $P(W)$ is approximated as the language model and $P(A|W)$ is obtained from a hidden Markov model acoustic likelihoods.

The conventional Maximum A-posteriori Probability Classifier (MAP) can be derived as a special case of the MBR classifier by considering a loss function that assigns a equal cost (say 1) to all misclassifications. Under the 0/1 loss function on word strings,

$$L(W, W') = \begin{cases} 0 \text{ if } W = W' \\ 1 \text{ otherwise} \end{cases} \tag{7.2}$$

the classifier of Equation 7.1 reduces to the MAP classifier

$$\delta_{\text{MAP}} = \underset{W \in \mathcal{W}}{\operatorname{argmax}} P(W'|A) \tag{7.3}$$

where

$$P(W'|A) = \sum_{W \in \mathcal{W}: L(W,W')=0} P(W|A). \tag{7.4}$$

This illustrates why we are interested in MBR decoders based on other loss functions: the MAP decoder is optimal with respect to a loss function that is overly harsh. It does not distinguish between different types of recognition errors and good sentences receive the same penalty as poor sentences.

## 7.3 MBR Classifiers for SMT

Statistical Machine Translation can be described as a classification task that maps a word sequence $\mathbf{f} = f_1^J$ in a source language (e.g. Chinese) to a word sequence $\mathbf{e} = e_1^I$ in a target language (e.g. English), where $f_i$ are words in the source vocabulary, and $e_i$ are words in the target vocabulary.

We now show the formulation of MBR classifiers for statistical machine translation, and present various loss functions that measure translation performance. In addition we will demonstrate that syntactic structure can be incorporated into the loss functions by considering parse-trees for the source sentence and its translation.

Suppose we have a loss function that measures the quality of a given translation using information from the word sequences, alignments and parse-trees. Given a foreign sentence $\mathbf{f}$ and its parse-tree $T(\mathbf{f})$, its correct translation $\mathbf{e}$ with word alignment $\mathbf{a}$ and parse-tree $T$, the loss function $L((\mathbf{e}', \mathbf{a}', T'), (\mathbf{e}, \mathbf{a}, T); \mathbf{f}, T(\mathbf{f}))$ should reflect the quality of a candidate translation $\mathbf{e}'$ with word alignment $\mathbf{a}'$ and parse-tree $T'$.

Suppose we have the true distribution $P(\mathbf{e}, \mathbf{a}, \mathbf{f})$ that describes translations of human quality. We can then measure the performance of the classifier $\delta(\mathbf{f})$

using the Bayes-Risk Bayes-Risk $R(\delta(\mathbf{f})) = E_{P(\mathbf{e},\mathbf{a},\mathbf{f})}[L((\mathbf{e},\mathbf{a},T),\delta(\mathbf{f}))]$. Given a loss function and a distribution, the classifier that minimizes $R(\delta(\mathbf{f}))$ is given by (Bickel and Doksum, 1977)

$$\delta(\mathbf{f}) = \underset{\mathbf{e}',\mathbf{a}',T'}{\operatorname{argmin}} \sum_{\mathbf{e},\mathbf{a},T} L((\mathbf{e}',\mathbf{a}',T'),(\mathbf{e},\mathbf{a},T);\mathbf{f},T(\mathbf{f}))P((\mathbf{e},\mathbf{a}|\mathbf{f})). \tag{7.5}$$

This optimal decoder has the usual difficulties of search (minimization) and computing the expectation under the true distribution. In practice, we will consider the space of translations to be an N-best list of translation alternatives generated under a translation model. We will also approximate the true distribution by a statistical translation model.

### 7.3.1 Translation Loss functions

We will now present a three-level hierarchy of loss functions for translation which have the general form $L((\mathbf{e}',\mathbf{a}',T'),(\mathbf{e},\mathbf{a},T);\mathbf{f},T(\mathbf{f}))$, and make use of different levels of lexical and syntactic information.

#### Tier 1 Loss functions

The first tier of loss functions has no information about word alignments or parse-trees and can be reduced to $L(\mathbf{e},\mathbf{e}')$. Examples of loss functions in this category are Levenshtein distance that measures word-error rate (WER), position-independent word-error rate (PER) (Och and Ney, 2002) and the BLEU score (Papineni et al., 2001). A loss function of this type depends only on information from word strings.

PER measures the minimum number of edit operations needed to transform a word string to any permutation of the other word string. The PER score (Och and Ney, 2002) is then computed as a ratio of this distance to the number of words in the reference word string. BLEU score (Papineni et al., 2001) measures the precision of unigrams, bigrams trigrams and four-grams in the hypothesis word string with respect to the reference word string, and includes a length penalty ($\gamma$) if the hypothesis is shorter than the reference. The score is computed as a geometric mean of these four precisions, weighted by the length penalty.

$$BLEU(W,W') = exp(\frac{1}{4}\sum_{i=1}^{4} log(p_i(W,W'))) * \gamma(W,W'). \tag{7.6}$$

Therefore, BLEU measures accuracy, rather than an error rate.

**Tier 2 Loss functions**

The second tier of translation loss functions uses information from word strings and parse-trees only, and can be written as $L((\mathbf{e}, T), (\mathbf{e}', T')$. This loss function has no access to any information from the source sentence or the word alignment. An example of such a loss function is a tree-kernel (Collins and Duffy, 2002) that measures the number of common subtrees between any two parse-trees $T$ and $T'$. This metric measures structural similarity between the parse-trees of the hypothesis and the reference translation. To measure this similarity, we consider a representation of parse-trees that tracks all sub-trees seen in training data. This is done by enumerating (implicitly) all tree fragments in the training data $1, ..., d$, where $d$ is huge. Each tree $T$ is then represented as a $d$ dimensional vector $h(T) = [h_1(T), ..., h_d(T)]$ , where the $i^{th}$ component $h_i(T)$ counts the number of occurrences of the $i^{th}$ tree fragment in tree $T$. Given two trees $T$ and $T'$, the tree-kernel computes an inner-product of the two structures using an efficient recursion whose computational complexity is independent of $d$.

$$\text{Tree-Kernel}(W, W') = h(T) \cdot h(T') = \sum_{k=1}^{d} h_k(T) h_k(T'). \tag{7.7}$$

The Tree-Kernel is a measure of accuracy; a larger score implies greater similarity between the two trees and vice-versa.

**Tier 3 Loss functions**

The third tier of loss functions is trans-lingual and uses information from word strings, alignments and parse-trees in both languages , and can be written in the most general form $L((\mathbf{e}, \mathbf{a}, T), (\mathbf{e}', \mathbf{a}', T'); \mathbf{f}, T(\mathbf{f}))$. We will now describe an example of such a loss function. We first assume that each node $n$ in the source tree $T(\mathbf{f})$ can be mapped to a node $m$ in $T$ (and a node $m'$ in $T'$) using word alignment $a$ (and $a'$). We also denote $t_m$ to be the subtree of $T$ rooted at node $m \in T$ and $t'_{m'}$ to be the subtree of $T'$ rooted at node $m' \in T'$.

Such a node-to-node alignment between nodes in the source and the target trees can be obtained using MT word-to-word alignments. For each node $n$ in the source tree $T(\mathbf{f})$ we first obtain the source word sequence that corresponds to the leaves of the subtree rooted at $n$. We next consider the subset of words in the target sentence that are aligned to any word in this source word string, and select the leftmost and rightmost word from this set. We then obtain the closest common ancestor $m$ of the leaf nodes corresponding to these two words in the

target language parse tree. This procedure allows us to obtain a node-to-node correspondence between nodes $n \in T(\mathbf{f})$ and $m \in T$.

The loss function can now be computed as

$$\text{BiTreeLoss}((\mathbf{e}, \mathbf{a}, T), (\mathbf{e}', \mathbf{a}', T'); \mathbf{f}, T(\mathbf{f})) = \sum_{n \in T(\mathbf{f})} d(t_m, t'_{m'}), \qquad (7.8)$$

where $d(t, t')$ is a distance measure (e.g. 0/1 loss) between sub-trees $t$ and $t'$. The *BiTree Error Rate* is then computed as a ratio of this loss to the number of nodes in the source tree $T(\mathbf{f})$. An example of a tree-to-tree mapping between a source (Chinese) parse-tree and parse-trees of two competing candidate English translations is shown in Figure 7.1. The figure shows node-to-node mappings between some of the nodes in the source tree and the two target trees.

Among these loss functions, BLEU directly takes into account multiple reference translations. In case of the other loss functions, we consider multiple references in the following way. For each sentence, we compute the error rate of the hypothesis translation with respect to the most similar reference translation under each loss function.

## 7.4 Experiments

We performed our MT experiments on the NIST 2002 MT-eval set consisting of 878 sentences. The baseline translation model was used to generate 1000-best translation hypotheses for each sentence in the test set. The N-best lists were rescored using the different translation loss functions described in Section 7.3. The performance of the MBR decoder under the various loss functions is presented in Table 7.1. We say we have a matched condition when the same loss function is used in both the error rate and the decoder design.

We observe that the MBR decoder optimized for each loss function performs the best under the corresponding error metric. We also notice some affinity among the loss functions. The MBR decoding under the Bitree Loss function obtains a significant WER reduction. The Parse-Kernel metric assigns a higher score for translations which are parsable; the MBR decoder under this error does poorly with respect to all metrics other than parse-kernel.

Our second set of experiments were performed on top of the best system (as of $4^{th}$ week of the workshop) that was trained with the max-BLEU training with syntactic feature functions. The results are summarized in Table 7.2

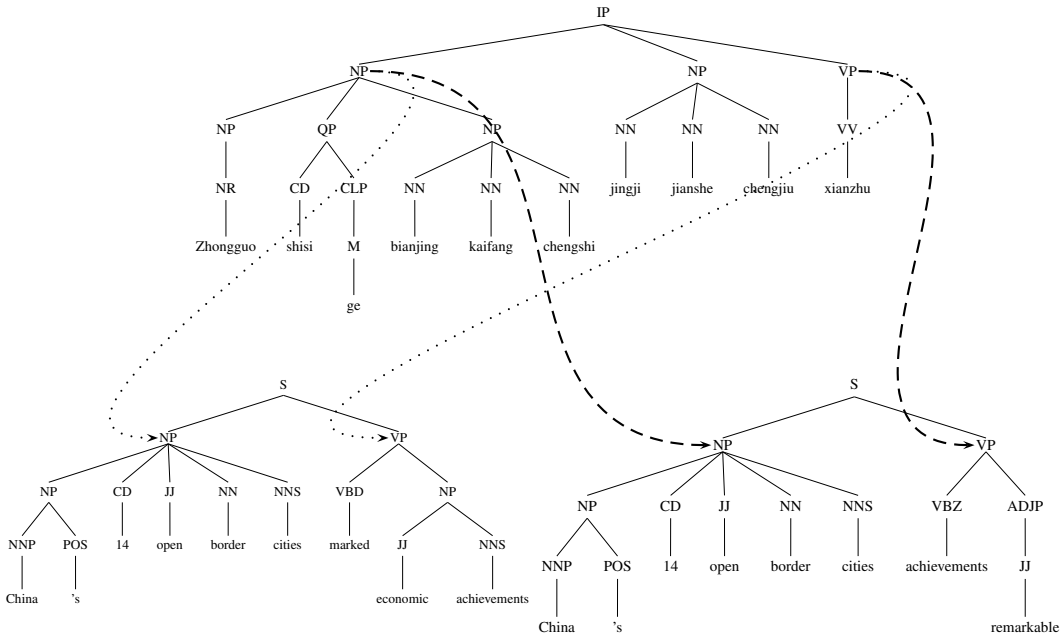



Figure 7.1: An example showing a parse-tree for a Chinese sentence and parse-tree for two of the candidate English translations. Node-to-node alignments between some of the nodes of the Chinese tree and the two English trees are displayed.

We observe that the MBR decoder under the BLEU loss function improves over the best system trained with the max-BLEU training.

## 7.5 Conclusion

We have described the formulation of Minimum Bayes-Risk classifiers for machine translation. We have described several translation loss functions that incorporate varying degrees of syntactic structure from Chinese and English parse-trees and word alignments. This can accomplished without building detailed models of these linguistic features and retraining models from scratch. However, we emphasize that the MBR framework does not preclude the construction of complex models of syntactic structure. Our second set of experiments show that models that have been trained with linguistic features could still benefit by the application

| | Performance Metrics | | | | |
|---|---|---|---|---|---|
| | BLEU (%) | mWER(%) | mPER (%) | mParseKernel | mBiTree Error(%) |
| MAP(baseline) | 31.6 | 62.4 | 39.3 | 1002.2 | 46.56 |
| MBR Decoder | | | | | |
| BLEU | **31.9** | 62.5 | 39.2 | 1113.1 | 46.75 |
| WER | 31.8 | **61.8** | 38.8 | 1016.4 | 46.16 |
| PER | 31.7 | 62.2 | **38.5** | 835.5 | 46.48 |
| Parse-Kernel | 29.9 | 68.5 | 43.2 | **4478.1** | 50.59 |
| BiTree Loss | 31.1 | 61.6 | 39.2 | 840.5 | **45.28** |

Table 7.1: Translation performance of the MBR decoder under various loss functions. For each metric, the performance under a matched condition is shown in bold. BLEU and mParseKernel represent similarities while other metrics measure error rates.

| | Performance Metrics | | | |
|---|---|---|---|---|
| | BLEU | mWER | mPER | NIST |
| Best System | 32.9 | 61.7 | 38.3 | 9.40 |
| MBR-BLEU | **33.2** | 61.7 | 38.3 | 9.65 |

Table 7.2: MBR Decoding Under BLEU loss function on top of the best system at the end of $4^{th}$ week. Matched condition is indicated in bold.

of MBR decoding procedures.

Each of these loss functions could be valuable in describing some aspect of system performance. We have shown through our experiments that MBR classifiers can allow the decoding procedure to be tuned for specific loss functions. We have also performed experiments to show that MBR decoding under the BLEU loss function can obtain further improvements on top of a baseline system trained under max-BLEU training.

In future we plan to investigate translation loss functions based on alignments obtained using the tree-to-string and the tree-to-tree translation models. We expect these alignments to improve upon the tree-to-tree alignments (Section 7.3) that were obtained using the MT word alignments. We also plan to extend the search space of the MBR decoder to word lattices generated by the translation system, and by considering more hypotheses, obtain improved performance.

# Chapter 8

# Conclusions

## 8.1 Summary

The goal of the workshop has been to integrate better models of syntactic structure into statistical models for machine translation.

The workshop started with a very strong baseline – the alignment template statistical machine translation system that obtained the best results in the 2002 and 2003 DARPA MT evaluations. During the workshop we developed more than 450 different feature functions with the goal to improve the syntactic well-formedness of the machine translation output of the alignment template baseline system. We break our feature functions into two broad categories: implicit syntactic feature functions and explicit syntactic feature functions. The implicit syntactic feature functions just try to exploit (better than the baseline system) the translations provided in the training and development corpora without using additional annotated data and explicit linguistic modeling. Prominent examples of this type of feature function are the multi-sequence alignment of hypotheses, the Model 1 score or the specific word penalty feature functions. A major goal during the workshop was to find out if we can exploit annotated data, especially treebanks for Chinese and English, by using tools like POS tagger, parser or shallow analysis tools. Prominent examples of this type of feature function are the parser probabilities of the tree-to-string and tree-to-tree alignment models.

Most of the experiments were performed using a 1000-best list experiment. Due to the limitations of this relatively small list there is an estimated upper bound of 35.7% BLEU score and a theoretical upper bound of 45.25% BLEU score if the optimal translations with respect to the four reference translations are picked

(Section 1.4).

The best single feature functions developed during the six week workshop are in the category 'implicit syntax', most prominently the Model 1 feature function. Most other feature functions and especially the feature functions from the categories shallow, deep and tricky did not obtain statistically significant improvements. A major problem is that many of the results are not statistically significantly better than the baseline. It would be necessary to measure the improvement of single feature functions on a much larger test corpus. Yet, the overall improvement by performing feature combination was 1.6% from 31.6% to 33.2%, which is statistically significant. If we contrast that to the estimated upper bound on that test corpus of 35.7%, our conclusion is that within the boundary conditions this is actually a considerable improvement over the baseline system.

During the workshop we created a large number of resources that we expect to be useful for various follow-up research projects. First of all the 16384-best lists of the development and test corpora created before the workshop are a valuable resource for various language modeling experiments. Both, the Chinese input sentences and the best 1000 translations of the development and test corpora have been parsed. The creation of this resource took over 3000 hours of computing time. This resource will be useful for any research to experiment with additional feature functions making use of parse trees. The more than 450 developed feature functions are all precomputed and are a valuable resource for conducting machine learning research to develop better discriminative training techniques useful in machine translation. All three major resources: $n$-nbest lists, bilingual parse annotation and feature function files are unique in the field of machine translation. We expect that these resources will stimulate a large amount of useful MT research.[1]

## 8.2 Outlook

Here is a list of interesting research problems which can be tackled in the future:

### Better evaluation metrics

During the workshop, we typically used the BLEU evaluation metric for error analysis, training and assessment of results because according to the experience

[1] Unfortunately, due to copyright problems it is not possible to make all of these resources freely available at this point in time.

of team members this metric seems to be better than other available metrics in assessing relatively small improvements with respect to word order changes.

One problem with existing metrics is that they have a very poor correlation with translation quality when used to score single single sentences (Melamed, Green, and Turian, 2003). This affects the quality of the produced oracle translations that are used in our contrastive error analysis. In addition, certain types of improvements (like reducing the number of omitted content words) result in a subjectively too small improvement in the BLEU score.

## Improved discriminative training for MT

The use of discriminative training for machine translation directly related to the evaluation criterion has shown to be a very effective method to boost MT performance. The technique of (Och, 2003), which was used at the workshop, works well for a small number of parameters (less than 10). Yet, it seems that for larger number of parameters the training technique is not reliable and it is especially not able to combine a large number of very small improvements into one large improvement. This obviously depends directly on the size of the development corpus used. Hence, one solution would be to use a larger development corpus for discriminative training which would allow to estimate larger number of parameters. An alternative promising approach would be the use of smoothed discriminative training criteria related to the evaluation criterion as is commonly done in the speech community (Beyerlein, 1997; Schlüter and Ney, 2001).

## Bilingual syntactic analysis

The experiments in Section 5.1 on the 'cross-lingual constituent alignment features' show that often certain constituents do not align nicely between Chinese and English parse trees. This has also been observed in (Hwa et al., 2002) and for French–English in (Fox, 2002). The alignment mismatch that we observe is partially due to alignment and parser errors but partially also due to systematic language differences. A systematic approach to deal with this problem could be to analyze the source language sentence in such a way that the resulting entities can easily be transferred into the target language. This bilingual syntactic analysis would be specific for the considered language pair.

## Parser as language model for MT

During the workshop large amounts of English machine translation output has been parsed. The majority of these hypothesized sentences are not syntactically well-formed. However, our experiments using the statistical parser as an additional language model were not successful. The results described in Section 4.1 show that the parser prefers the 'bad' machine translation output in more than fifty percent of the cases over the presumably syntactically well-formed reference translations!

The experiments performed in the workshop used a parser trained off of the Penn treebank with about one million words. The language model used in the baseline system is a trigram language model trained on about 250 million words. A promising idea is to parse this language model training data and to retrain the parser on that data. If this is done, the statistical parser will presumably perform better. In addition, it would be interesting to examine the effect of the change in domain from the Penn treebank to the MT training corpus on the highly lexicalized parser models that were used, and to evaluate whether reducing lexical dependencies could result in more robust syntactic modeling.

## Confident Parsers

Today's statistical parsers do not provide confidence information on the quality of the produced analysis. Currently, the parser hallucinates parse trees for completely garbled sentences which have no syntactic structure at all. Ideally, we would like the parser to provide information on which parts of the syntactic analysis should be presumed to be correct, and hich parts are likely to be wrong. This information could be used to develop feature functions that rely only on those parts of the parse tree in which there should be a high level of confidence.

# Appendix A

# Contrastive Error Analysis

## A.1 Human Evaluation

During the workshop the members of the team conducted an error analysis contrasting the Oracle-BLEU translations in the $N$-best lists with the sentences preferred by the baseline MT system.

The categories are listed below:

0: deletion: missing articles, prepositions, and other function words: incoherent

1: deletion: missing articles, prepositions, and other function words: changing meaning significantly

2: deletion: missing content words

3: insertion: hallucinated content words

4: substitution: word choice (WSD wrong)

5: substitution: wrong tense

6: word order: VP internal structure

7: word order: NP internal structure

8: word order: PP internal structure

9: messed up puncutation

10: named entities wrong

11: wrong single/plural cases

12: wrong noun dependency (ie, noun incorrectly moves from subject to object, etc)

13: insertion, deletion: pronouns wrong

For 100 sentences which were randomly selected for each team member, they

were asked to make a binary decision as to whether the problem existed in the BLEU-Oracle translation of that sentence, and in the baseline systems output. Two native Chinese speakers, and two native English speakers double-annotated the same set of English hypotheses.

The results of this error analysis were inconclusive. In many cases the BLEU Oracle sentence appeared to have more problems than the baseline-produced sentence. Because the sentences were selected at random, the quality of both sentences examined was often quite low, and humans have difficulty analyzing hypotheses with many problems. In the future it would be interesting to repeat this analysis by first looking for Oracle sentences which are more likely to be correct, and then conducting the contrastive analysis. In addition, techniques for artificially improving the output to make it easier for human annotators to analyze should be examined, though of course great care must be taken to avoid distortion of the analysis.

# Appendix B

# Used Symbols

$p(\cdot)$: generic symbol for modeled probabilities
$Pr(\cdot)$: generic symbol for 'true' probabilities
$C(\cdot)$: class function for mapping words to word classes

$f_1^J = \mathbf{f}$: source language symbols (Chinese)
$\tilde{f}_1^K$: sequence of source language phrases
$e_1^I = \mathbf{e}$: target language symbols (English)
$\tilde{e}_1^K$: sequence of target language phrases
$a_1^J = \mathbf{a}$: word alignment vector (as produced by IBM alignment models)
$\tilde{a}_1^K$: phrase alignment
$\phi_1^I$: fertility of target language words
$B_1^I$: inverted word alignment
$B_{i,j}$: $j$-th element of $B_i$ in ascending order

$j, J$: index and length for source language sentence
$i, I$: index and length for target language sentence
$k, K$: index and length for segmented source language sentence
$m, M$: index for feature function and number of feature functions for maximum entropy modeling

$c_1^J = \mathbf{c}$: coverage vector
$z$: alignment template
$Z = (z, j)$: alignment template instantiation (a specific alignment template starting used in a sentence starting at position $j$)

$P_e(\cdot)$: preprocessing function for target language sentence

$P_f(\cdot)$: preprocessing function for source language sentence
$P_e^{-1}(\cdot)$: postprocessing function for target language sentence

$\theta$: set of all parameters for translation model
$\gamma$: set of parameters for language model
$M$: number of parameters/feature functions in log-linear model
$\lambda_1^M$: parameters of log-linear model
$h_1^M(e_1^I, f_1^J)$: feature functions
$\mathbf{f}_1^S, \mathbf{e}_1^S$: training corpus sentences

$R_s$: number of reference translations for sentence $s$
$\mathbf{e}_{s,r}$: $r$-th reference translation of the $s$-th sentence ($r = 1, \ldots, R_s$)

$T(\mathbf{e}), T(\mathbf{f})$: parse tree of $\mathbf{e}$ or $\mathbf{f}$
$\mathrm{POS}(\mathbf{e}), \mathrm{POS}(\mathbf{f})$: parts-of-speech sequence of $\mathbf{e}$ or $\mathbf{f}$
$\mathrm{CHK}(\mathbf{e}), \mathrm{CHK}(\mathbf{f})$: chunk-tagged sequence of $\mathbf{e}$ or $\mathbf{f}$
$T_{\mathbf{e}}, T_{\mathbf{f}}$: arbitrary source/target language parse tree
$\mathrm{POS}_{\mathbf{e}}, \mathrm{POS}_{\mathbf{f}}$: arbitrary source/target language parts-of-speech sequence
$\mathrm{CHK}_{\mathbf{e}}, \mathrm{CHK}_{\mathbf{f}}$: arbitrary source/target language chunk segmentation

**References**

Berger, Adam L., Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–72, March.

Beyerlein, P. 1997. Discriminative model combination. In *Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 238–245, Santa Barbara, CA, December.

Bickel, P. J. and K. A. Doksum. 1977. *Mathematical Statistics: Basic Ideas and Selected topics*. Holden-Day Inc., Oakland, CA, USA.

Bies, Ann, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. Bracketing guidelines for treebank II style. Penn Treebank Project, January.

Bikel, Daniel and David Chiang. 2000. Two statistical parsing models applied to the chinese treebank. In *Proceedings of the Second Chinese Language Processing Workshop*, pages 1–6, Hong Kong.

Bikel, Daniel M. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Human Language Technology Conference (HLT)*.

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Charniak, E. 2000. A maximum-entropy-inspired parser. In *Proc. of NAACL 2000*.

Collins, M. and N. Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the weighted perceptron. In *Proceedings of the Conference on Empirical Methods in Natural Language Pr ocessing*, Philadelphia, PA, USA.

Collins, Michael. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the 7th International Conference on Machine Learning*.

Collins, Michael and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of ACL 2002*.

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

Fei Xia. 2000. The part-of-speech tagging guidelines for the penn chinese treebank. Technical Report IRCS-00-07, IRCS, University of Pennsylvania.

Fox, Heidi J. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 304–311, Philadelphia, PA.

Gildea, Daniel. 2003. Loosely tree-based alignment for machine translation. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sapporo, Japan.

Goel, V. and W. Byrne. 2000. Minimum Bayes-risk automatic speech recognition. *Computer Speech and Language*, 14(2):115–135.

Gusfield, Dan. 1997. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, Cambridge, England.

Hajič, Jan, Martin Čmejrek, Bonnie Dorr, Yuan Ding, Jason Eisner, Daniel Gildea, Terry Koo, Kristen Parton, Gerald Penn, Dragomir Radev, and Owen Rambow. 2002. Natural language generation in the context of machine translation. Technical report, Center for Language and Speech Processing, Johns Hopkins University, Baltimore. Summer Workshop Final Report.

Hwa, R., P. Resnik, A. Weinberg, and O. Kolak. 2002. Evaluating translational correspondence using annotation projection. In *the Proceedings of the 40th Annual Meeting of the ACL*, Philadelphia, PA.

Levenshtein, V. I. 1965. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information transmission*, 1(1):8–17.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330, June.

Melamed, I. Dan, Ryan Green, and Joseph P. Turian. 2003. Precision and recall of machine translation. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL)*, Edmonton, Canada.

Ney, Hermann. 1995. On the probabilistic-interpretation of neural-network classifiers and discriminative training criteria. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(2):107–119, February.

Ney, Hermann, M. Generet, and Frank Wessel. 1995. Extensions of absolute discounting for language modeling. In *Proc. of the Fourth European Conf. on Speech Communication and Technology*, pages 1245–1248, Madrid, Spain, September.

Ngai, Grace and Radu Florian. 2001. Transformation-based learning in the fast lane. In *Proceedings of the 39th ACL Conference*.

Nianwen Xue and Fei Xia. 2000. The bracketing guidelines for the penn chinese treebank. Technical Report IRCS-00-08, IRCS, University of Pennsylvania.

Och, Franz Josef. 2002. *Statistical Machine Translation: From Single-Word Models to Alignment Templates*. Ph.D. thesis, Computer Science Department, RWTH Aachen, Germany, October.

Och, Franz Josef. 2003. Minimum error rate training in statistical machine translation. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan, July.

Och, Franz Josef and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 295–302, Philadelphia, PA, July.

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

Och, Franz Josef, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *Proc. of the Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28, University of Maryland, College Park, MD, June.

Papineni, Kishore A., Salim Roukos, and R. T. Ward. 1997. Feature-based language understanding. In *European Conf. on Speech Communication and Technology*, pages 1435–1438, Rhodes, Greece, September.

Papineni, Kishore A., Salim Roukos, and R. T. Ward. 1998. Maximum likelihood and discriminative training of direct translation models. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 189–192, Seattle, WA, May.

Papineni, Kishore A., Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. Technical Report RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY, September.

Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 2002. *Numerical Recipes in C++*. Cambridge University Press, Cambridge, UK.

Radev, Dragomir R., Vasileios Hatzivassiloglou, and Kathleen R. McKeown. 1999. A description of the CIDR system as used for TDT-2. In *DARPA Broadcast News Workshop*, Herndon, VA, February.

Ratnaparkhi, Adwait. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142, University of Pennsylvania, May. ACL.

Santorini, Beatrice. 1990. Part-of-speech tagging guidelines for the Penn Treebank project. 3rd revision, 2nd printing.

Schlüter, Ralf and Hermann Ney. 2001. Model-based MCE bound to the true Bayes' error. *IEEE Signal Processing Letters*, 8(5):131–133, May.

Shen, L., A. Sarkar, and A. K. Joshi. 2003. Using LTAG based features in parse reranking. In *Proc. of EMNLP 2003*.

Tjong Kim Sang, Erik F. and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal.

Ueffing, Nicola, Franz Josef Och, and Hermann Ney. 2002. Generation of word graphs in statistical machine translation. In *Proc. Conference on Empirical Methods for Natural Language Processing*, pages 156–163, Philadelphia, PA, July.

Vogel, Stephan, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *COLING '96: The 16th Int. Conf. on Computational Linguistics*, pages 836–841, Copenhagen, Denmark, August.

Wahlster, Wolfgang, editor. 2000. *Verbmobil: Foundations of speech-to-speech translations*. Springer Verlag, Berlin, Germany.

Xia, Fei. 2000. The part-of-speech guidelines for the penn chinese treebank $(3.0)$. Technical Report IRCS Report 00-07, University of Pennsylvania, Pennsylvania, PA.

Xue, Nianwen, Fu-Dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated chinese corpus. In *Proceedings of the 19th. International Conference on Computational Linguistics (COLING 2002)*, Taipei, Taiwan.

Yamada, K. and K. Knight. 2002. A decoder for syntax-based MT. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yamada, Kenji and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 523–530, Toulouse, France, July.