PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 93 JANUARY 2010 17–26

# Sulis: An Open Source Transfer Decoder for Deep Syntactic Statistical Machine Translation

Yvette Graham

National Centre for Language Technology, Dublin City University, Ireland

**Abstract**

In this paper, we describe an open source transfer decoder for Deep Syntactic Transfer-Based Statistical Machine Translation. Transfer decoding involves the application of transfer rules to a SL structure. The N-best TL structures are found via a beam search of TL hypothesis structures which are ranked via a log-linear combination of feature scores, such as translation model and dependency-based language model.

## 1. Introduction

Deep Syntactic Transfer-Based Statistical Machine Translation (SMT) applies standard methods of Phrase-Based SMT (PB-SMT) (Koehn et al., 2003) to transfer between deep syntactic structures. For example, both Riezler and Maxwell (2006) and Bojar and Hajič (2008) use beam search decoding and a log-linear model to combine feature scores when transferring from source language (SL) deep syntactic structure to target language (TL) deep syntactic structure. Each uses different statistical models for transfer, however. Bojar and Hajič (2008) use the Synchronous Tree Substitution Grammar formalism, in which the probability of attaching pairs of dependency treelets into aligned pairs of frontiers given frontier state labels is used as a main feature function, as well as a bigram dependency-based language model. Riezler and Maxwell (2006), on the other hand, use a translation model computed from relative frequencies of automatically induced transfer rules and a trigram dependency-based language model. Riezler and Maxwell (2006) diverge somewhat from PB-SMT, however, by only applying the language model after decoding on the n-best decoder

 Corresponding author: ygraham@computing.dcu.ie
Cite as: Yvette Graham. Sulis: An Open Source Transfer Decoder for Deep Syntactic Statistical Machine Translation. The Prague Bulletin of Mathematical Linguistics No. 93, 2010, pp. 17–26. ISBN 978-80-904175-4-0. doi: 10.2478/v10108-010-0005-7.

output.[1] In this paper, we describe the Sulis transfer decoder that, like Riezler and Maxwell (2006), uses a translation model computed from relative frequencies of automatically induced transfer rules, but for language modeling, like Bojar and Hajič (2008), we use a dependency-based language model during transfer decoding to help decide the n-best TL structures. In addition, we increase the dependency-based language model from a bigram model, as in Bojar and Hajič (2008), to a trigram model.

Sulis forms part of a larger system consisting of several resources, most of which are open source and all of which are at least available to the research community: XLE (Kaplan et al., 2002) for parsing and generation, Giza++ (Och et al., 2000) and Moses (Koehn and Hoang, 2007) for automatic word alignment, RIA (Graham and van Genabith, 2009) for automatic transfer rule induction, Ariadne and SRILM (Stolcke, 2002) for dependency-based language modeling and ZMERT (Zaidan, 2009) for Minimum Error Rate Training (Och, 2003) (MERT). Graham et al. (2009) contains the most recently published evaluation of Sulis.

## 2. Deep Syntactic Transfer-Based SMT

Deep Syntactic Transfer-Based SMT is composed of three parts, (i) parsing to deep syntactic structure, (ii) transfer from SL structure to TL structure and (iii) generation of TL sentence. Step (ii) involves a statistical search for the n-best TL deep syntactic structures by means of a transfer decoder that constructs TL structures by applying transfer rules to the SL structure. Transfer rules are similar to phrases in PB-SMT: a bitext corpus is firstly word-aligned before all rules consistent with the word-alignment are extracted. They differ from SMT phrases, however, in their structure: they are in the form of dependency graphs with missing arguments replaced by variables as opposed to linear sequences of words. As in PB-SMT, for each SL input structure their exists a large number of possible TL output structures. We use beam search to manage the large search space. TL hypotheses are ranked using a log-linear model to combine several feature scores, such as dependency-based language model and translation model. MERT is carried out on a development set to optimize the weights used to combine feature scores.

### 2.1. Decoding

### 2.2. Translation Model

As in PB-SMT, a Transfer-Based SMT translation model can be defined as a combination of several feature functions combined using a log-linear model:

$$p(e|f) = exp \sum_{i=1}^{n} \lambda_i h_i(e, f)$$

[1] Through personal communication with John Maxwell.

#### 2.2.1. Transfer Rule Probabilities

In PB-SMT the translation of an input sentence into an output sentence is modeled by breaking down the translation of the sentence into the translation of a set of phrases. Similarly, for Transfer-Based SMT, the transfer of the SL structure **f** into a TL structure **e** can be broken down into the transfer of a set of rules $\{\bar{f}, \bar{e}\}$:

$$p(\bar{f}_1^I|\bar{e}_1^I) = \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i)$$

We compute all rules from the training corpus and estimate the translation probability distribution by relative frequency of the rules:

$$\phi(\bar{f}, \bar{e}) = \frac{count(\bar{e}, \bar{f})}{\sum_{\bar{f}_i} count(\bar{e}, \bar{f}_i)}$$

This is carried out in both the source-to-target and target-to-source directions.

#### 2.2.2. Lexical Weighting

In PB-SMT, lexical weighting is used as a back-off since it provides richer statistics and more reliable probability estimates. Adapting this feature to deep syntax is straightforward. In PB-SMT the lexical translation probability of a phrase pair is calculated based on the alignment between the words in the phrase pair. For deep syntactic transfer, we simply calculate the same probability via the alignment of lexical items in the LHS and RHS of a transfer rule. The lexical translation probability of a RHS, $\bar{e}$, given the LHS, $\bar{f}$ and alignment $a$, is estimated as follows:

$$lex(\bar{e}|\bar{f}, a) = \prod_{i=1}^{length(\bar{e})} \frac{1}{|\{j|(i,j) \in a\}|} \sum_{\forall(i,j)\in a} w(e_i|f_j)$$

We use lexical weighting in both language directions.

#### 2.2.3. Transfer Rule Application

Decoding takes a single SL structure as input and involves a statistical search for the n-best TL structures. The current decoding algorithm works by creating TL solutions via a top-down application of transfer rules to the SL structure beginning at the root.[2] When the LHS of a rule unifies with the SL structure, the RHS produces a portion of TL structure. Figure 1 shows an example application of three rules to the dependency structure for the German sentence *Die Katze schläft gern* 'The cat likes to sleep' shown in Figure 1(a). Figure 1(b) shows the first transfer rule applied to the root node of the SL structure producing the TL structure portion shown in Figure 1(c). Transfer rule variables map arguments in the SL structure to the desired position

[2] In future work, we plan to extend the decoder by allowing rule application starting at any node in the SL structure.

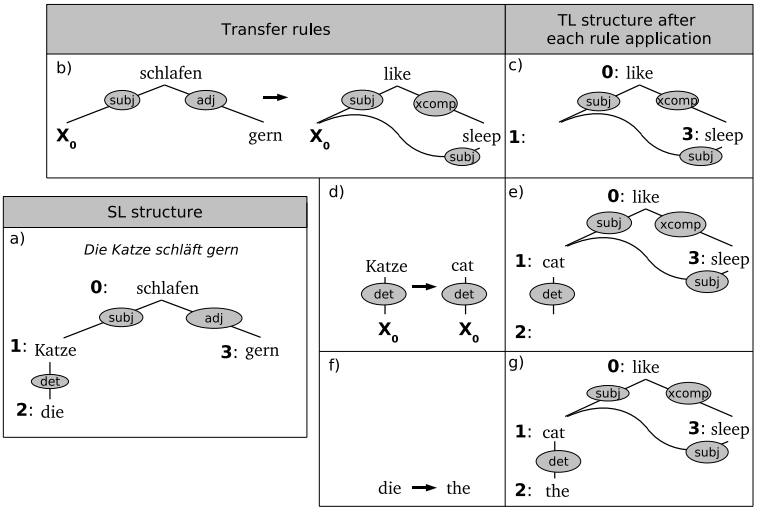


*Figure 1. Example top-down application of transfer rules*

when creating a TL solution. For example, variable $X_0$ in Figure 1(b) maps the *subject* of *schlafen* to the *subject* of *like* in the TL structure labeled with id number *1* shown in Figure 1(c). Next *Katze* in the SL structure is translated (Figures 1(d) and 1(e)), before finally *die* is translated (Figures 1(f) and 1(g)).

### 2.2.4. Beam Search

Partial translations (or translation hypotheses) are constructed by applying transfer rules to the SL structure. While TL translations are constructed, beam search manages the large search space by ranking translation hypotheses and pruning the search by dropping lower scoring hypotheses. A number of stacks are used to organize translation hypotheses into groups of comparable hypotheses, according to the portion of SL structure that has already been translated to produce each hypothesis, i.e. hypothesis stack *N* stores TL translation hypotheses with *N* nodes covered in the SL structure. For example, Figure 2(a) shows the hypothesis stacks for decoding the dependency structure of *Die Katze schläft gern* containing 4 nodes and therefore requiring stacks 1-4 for decoding, each stack storing translation hypotheses for solutions covering one to four nodes, respectively.

Transfer rules are indexed by root node so that they can be retrieved quickly to translate SL structure nodes. For example, in Figure 2(a) the rules rooted at node *Katze* are stored together. Since rules are applied top-down to the SL structure (see

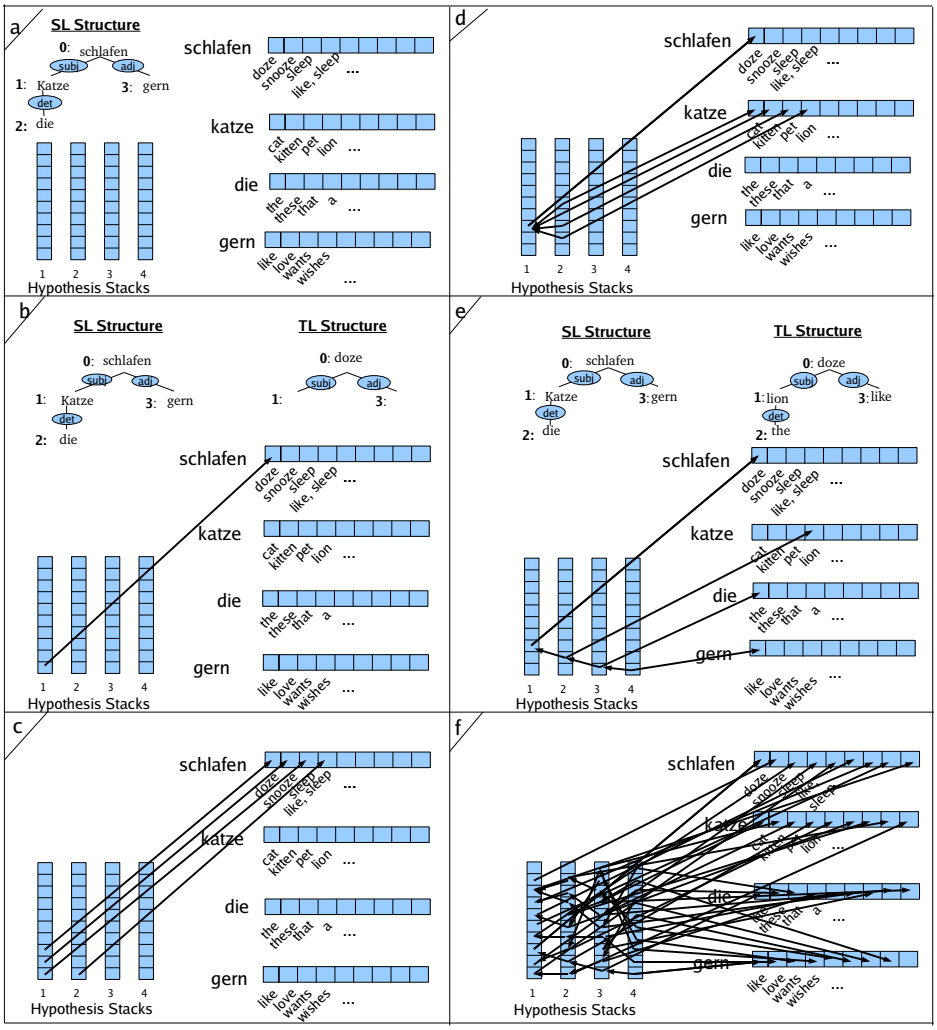


*Figure 2. Beam Search Decoding of Example German Deep Syntactic Structure*

Section 2.2.3) rules beginning at the root node of the SL structure are first used to construct hypotheses. For example, in Figure 2(b) the rule that translates the root node of the SL structure *schlafen* as *doze* is first used to construct a hypothesis and since it covers one SL node it is stored in hypothesis stack 1. Figure 2(c) shows the next three hypotheses that are constructed: *snooze*, *sleep* and *like sleep*. Hypotheses are ordered within each stack according to their score, high-to-low from bottom-to-top. We currently use histogram pruning. When a stack becomes full, lower scoring solutions are pruned by being popped off the top of the stack.

For efficiency, each partial translation is only stored once in memory even though it may be part of several different future hypotheses. For example, hypothesis stack 2 in Figure 2(d) contains four translations constructed by expanding hypothesis *doze* by four different rules, each translating the word *Katze* into a different TL word. These new hypotheses are represented by a reference to the most recently applied transfer rule (rules translating *Katze*) and a reference back to the previous hypothesis.

Figure 2(e) shows an example of how per single completed translation, the structure for *the lion likes to doze*, is represented in the hypothesis stacks and Figure 2(f) shows all hypotheses represented when the decoder has completed translating a single SL input structure. The n-best translated structures can be retrieved from the final stack.

### 2.2.5. Efficient Dependency-Based Language Modeling

Although the search space is limited by beam search, during decoding large numbers of TL hypothesis structures need to be ranked. At each expansion of a translation hypothesis (via joining of an existing hypothesis with a transfer rule) a language model score for the newly created hypothesis needs to be calculated. Since this is carried out very many times per single decoding run, it is vital that the method of calculating this score is highly efficient.

In our system, we pre-compute a dependency-based language model score for each transfer rule prior to beam search. This score is calculated only once for each rule even though a single rule may be part of several translation hypotheses. Then during decoding, when a translation hypothesis is expanded by adding a new rule, the new hypothesis score can be calculated quickly by combining the score of the old hypothesis, the rule score and a score calculated based on the probabilities of n-grams where the old hypothesis and rule join together. The probability of a TL hypothesis, $h_n$, that was produced by combining hypothesis $h_{n-1}$ and rule $r$ can be calculated as follows:

$$\mathtt{hyp\_score}(h_n) = \mathtt{hyp\_score}(h_{n-1}) * \mathtt{join\_score}(h_{n-1}, r) * \mathtt{rule\_score}(r)$$

Since $\mathtt{hyp\_score}(h_{n-1})$ and $\mathtt{rule\_score}(r)$ are already computed, only $\mathtt{join\_score}(h_{n-1}, r)$ needs to be computed to compute $\mathtt{hyp\_score}(h_n)$.

Figure 3 shows how the language model scores are efficiently calculated when decoding the dependency structure for the German sentence *Die Werbung spiegelt die Vielfalt der britischen Universität wider* 'The advertisement reflects the diversity of the

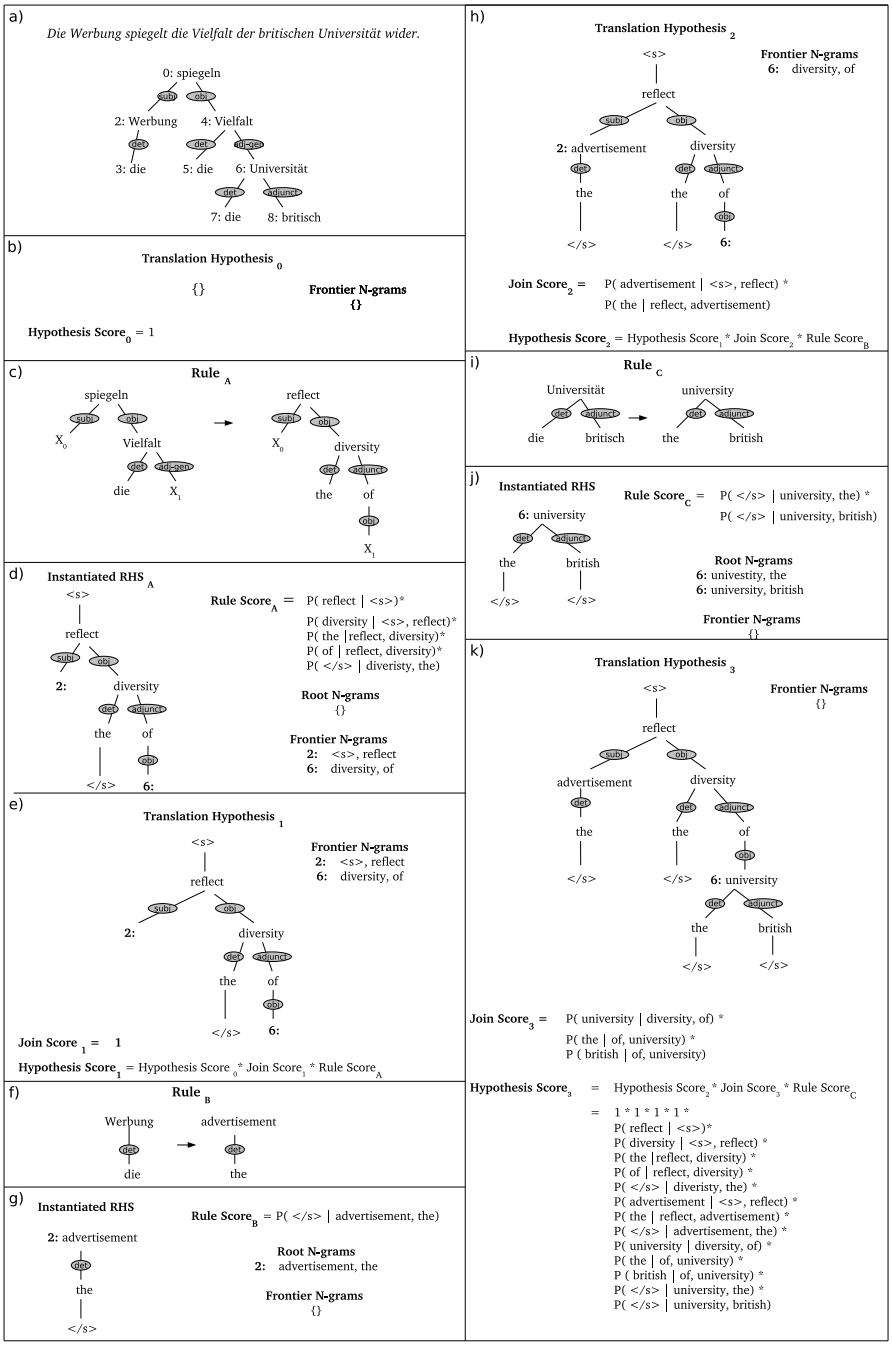




*Figure 3. Efficient Dependency-based Language Modeling*

British university'. We begin with the German dependency structure graph shown in Figure 3(a) with nodes labeled by id numbers. Figure 3(b) shows the initial empty translation hypothesis that has probability 1.

Figures 3(c), 3(f) and 3(i) show example transfer rules that can be applied to the German dependency structure. Dependency-based language model scores are precomputed for each rule by identifying all trigrams within the RHS structure and calculating the product of their individual probabilities; we call this the rule_score (see Figure 3(d) for $\mathrm{Rule}_A$, Figure 3(g) for $\mathrm{Rule}_B$ and Figure 3(j) for $\mathrm{Rule}_C$). In addition, for each rule, n-grams located at the RHS root node and frontier nodes are recorded. For example, $\mathrm{Rule}_B$ in Figure 3(g) has a single root node bigram *advertisement the* located at node 2, and $\mathrm{Rule}_A$ in Figure 3(d) has two frontier bigrams < *s* >, *reflect* and *diversity, of* located at nodes 2 and 6, respectively. This information is used to calculate the score of joining a rule and a hypothesis.

Figure 3(e) shows the translation hypothesis established by applying $\mathrm{Rule}_A$ to the German structure. The language model score for the structure is computed by combining the score of the previous hypothesis (since this is the first rule for this hypothesis, the previous hypothesis is the empty hypothesis and is therefore 1), the join score (since we are joining the rule with the empty hypothesis this score is also 1) and the rule score of $\mathrm{Rule}_A$ (see Figure 3(d)).

Figure 3(h) shows the translation hypothesis created by expanding $\mathrm{Hypothesis}_1$ by $\mathrm{Rule}_B$. Since this expansion involved adding a rule at node 2 in the TL structure, the joining trigrams are derived by creating lists of words via all possible combinations of the frontier bigrams belonging to $\mathrm{Hypothesis}_1$ labeled 2 and the root bigrams of $\mathrm{Rule}_B$, also labeled 2 (see root n-grams in Figure 3(g)). For this example, this results in a single word sequence <*s* >*reflect advertisement the* which forms two trigrams <*s* >-*reflect-advertisement* and *reflect-advertisement-the*. The score for $\mathrm{Hypothesis}_2$ is then calculated by combining the hypothesis score for $\mathrm{Hypothesis}_1$, this join score and the precomputed rule score for $\mathrm{Rule}_B$.

## 3. Using Sulis Decoder

### 3.1. Input Format

To use Sulis, go to *http://www.computing.dcu.ie/~ygraham/software.html* and follow instructions. The input to the decoder is a text file containing transfer rules and information for computing feature scores. Figure 4 shows an example rule entry for the decoder. The first two lines in Figure 4 gives the id number of the SL structure, 0, and the id number of the node in the SL structure where this rule is applied, also 0. The subsequent lines of the file are used for dependency-based language modeling and are paths of lexical items associated with nodes in the TL structure beginning at the frontier nodes back to the root node of the rule. For example, the rule in Figure 4 has three frontier nodes labeled *1*, *7* and *12*, and therefore the file contains three paths

```
rule: 0
start: 0
1  agree  0  <s>  -1
7  agree  0  <s>  -1
12  agree  0  <s>  -1
cf(1,eq(attr(var(0),'COMP'),var(12)))
cf(1,eq(attr(var(0),'PASSIVE'),-))
cf(1,eq(attr(var(0),'SUBJ'),var(1)))
cf(1,eq(attr(var(0),'PRED'),semform(agree,_17367,[var(1),var(12)],[])))
cf(1,eq(attr(var(0),'TENSE'),pres))
cf(1,eq(attr(var(0),'ADJUNCT'),var(6)))
cf(1,in_set(var(7),var(6)))
lhs_vars: 0
num_rhs: 1
str: 8.204571144249204
tsr: 8.519636252843213
stl: 7.730179751898788
tsl: 8.799261640333976
```

*Figure 4. Example Rule Entry of Input File: rule translating German structure word* meinen *as* agree*.*

from each of these nodes back to the root.[3] Next is the RHS of the rule.[4] Following that, is a list of SL nodes covered by the LHS of the rule. In the example in Figure 4 a single node, labeled *0* is covered by the rule. In addition, the number of TL lexicalized nodes produced by the rule is given. Finally, the source-to-target and target-to-source relative frequencies are given in the form of positive log probabilities, as well as the source-to-target and target-to-source lexical weights, also as positive log probabilities.

In addition to the rules for each structure, the decoder takes in a weights file for combining feature scores. The file should be in the format of Zaidan (2009) Z-MERT tool. For language modeling, the tool expects a dependency-based language model in ARPA format. To compute such a model, Ariadne open source tool in conjunction with the SRILM toolkit (Stolcke, 2002) can be used.

### 3.2. Output Format

The decoder outputs the n-best TL structures in the form of the union of the RHS equations of transfer rules used to construct it, as well as a list of feature scores and the total combined score.[5]

[3] Each lexical item in the path is labeled with its node id number, which is used to verify that no single trigram is counted more than once.

[4] In Figure 4, this is in the form of LFG F-structure Prolog equations, but can in fact be in any format, as it is not interpreted by the program code, but simply remains as a string of characters to be output if this rule forms part of a solution.

[5] These scores are needed for MERT.

## 4. Conclusion

In this paper, we present an open source transfer decoder for Deep Syntactic Transfer-Based SMT. The decoder applies standard methods of PB-SMT to deep syntactic transfer.

## Bibliography

Bojar, Ondřej and Jan Hajič. Phrase-Based and Deep Syntactic English-to-Czech Statistical Machine Translation. In *Proceedings of the third Workshop on Statistical Machine Translation*, Columbus, Ohio, June 2008.

Graham, Yvette and Josef van Genabith. An open source rule induction tool for transfer-based smt. *The Prague Bulletin of Mathematical Linguistics Special Issue: Open Source Tools for Machine Translation*, pages 37–46, 2009.

Graham, Yvette, Josef van Genabith, and Anton Bryl. F-structure transfer-based statistical machine translation. In *Proceedings of Lexical Functional Grammar Conference 2009*, Cambridge, July 2009.

Kaplan, Ronald M., Tracy H. King, and John T. Maxwell. Adapting existing grammars: the XLE experience. In *Proceedings of COLING 2002*, Taipei, Taiwan, 2002.

Koehn, Philipp and Hieu Hoang. Factored Translation Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, page 868–876, Prague, June 2007.

Koehn, Philip, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 48–54, Edmonton, Alberta, 2003.

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

Och, Franz Josef, Christoph Tillmann, and Hermann Ney. Improved alignment models for statistical machine translation. In *Proceedings of the 1999 Conference on Empirical Methods in Natural Language Processsing (EMNLP 99*, pages 20–28, College Park, MD, 2000.

Riezler, Stefan and John Maxwell. Grammatical Machine Translation. In *Proceedings of HLT-ACL*, pages 248–255, New York, 2006.

Stolcke, Andreas. Srilm - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, Denver, Colorado, September 2002.

Zaidan, Omar. Z-mert: A fully configurable open source tool for minimum error rate training of machine translation systems. pages 79–88, 2009.