Fundamental and New Approaches to Statistical Machine Translation

Lucia Specia

Research Group in Computational Linguistics
Unviersity of Wolverhampton
Stafford Street, Wolverhampton, WV1 1SB
L.Specia@wlc.ac.uk,
WWW home page: http://pers-www.wlv.ac.uk/ in1316/

1 Introduction

Statistical Machine Translation (SMT) is an approach to automatic text translation based on the use of statistical models and examples of translations. Although Machine Translation (MT) systems developed according to other paradigms are still in use, mainly rule-based or example-based MT, SMT dominates academic MT research and has gained significant commercial interest over the last two decades.

Machine Translation was conceived as one of the first applications of the newly invented electronic computers back in 1940's. Communications between Warren Weaver (director of the Natural Sciences Division of the Rockefeller Foundation) and his fellow researchers are often mentioned as the first attempt to use computers for translation, and can be also thought of as a pioneering idea for using statistical models for the task. According to [17, p.22], Warren Weaver proposed in 1947:

> "Recognizing fully, even though necessarily vaguely, the semantic difficulties because of multiple meanings, etc., I have wondered if it were unthinkable to design a computer which would translate. Even if it would translate only scientific material (where the semantic difficulties are very notably less), and even if it did produce an inelegant (but intelligible) result, it would seem to me worth while. Also knowing nothing official about, but having guessed and inferred considerable about, powerful new mechanized methods in cryptography - methods which I believe succeed even when one does not know what language has been coded - one naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say 'This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode'."

This view of the translation problem as a cryptography problem and a "decoding process" is strongly connected to the fundamental model of Statistical Machine Translation proposed much later, the *Noisy Channel Model* (Section

2). The formalization of such model was the basis for the first systems for SMT, based on word alignment techniques for the word-by-word translation (Section 3). Currently, the most successful systems exploit more elaborate models based on phrase translation (Section 4) and on a number of parameters to weight alternative phrases for ambiguous inputs and also weight different aspects of translation according to their relevance for a given language pair, translation task and type / genre / domain of text (Section 5 and 7). The search for the best translation among a number of possibilities is performed by a *decoder*, which uses a search strategy and heuristics to prune the search space (Section 6). A key aspect in the field of SMT is to automatically evaluate alternative outputs for improved system development and comparisons among different systems (Section 8). Some pointers to recent developments in the field are given in Section 9.

Disclaimer: This material gives a very basic overview on the topic of SMT. For broader reviews, see [29] and [21].

2 The Noisy Channel Model and the Main Components of an SMT system

The Noisy Channel Model was proposed in the field of Information Theory, by Claude Shannon in 1948 . Shannon's goal was to maximize the amount of information that could be transmitted over an imperfect (noisy) communication channel (like a noisy phone line). It has been used as underlying framework in different areas related to language processing, including Speech Recognition, Spell Checkers, Optical Character Recognition and Machine Translation. It assumes that the original text has been accidentally scrambled or encrypted (using a different alphabet, for example) and the goal is to find out the original text by "decoding" the encrypted/scrambled version, as depicted in Figure 1.

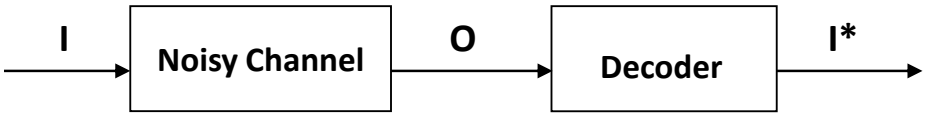

Fig. 1. The Noisy Channel Model: Message I is the input to the channel (e.g., text in native language). I gets encrypted into O using certain coding scheme (e.g., text in foreign language). The goal is to find a decoder that can reconstruct the input message as faithfully as possible into I^*

In a probabilistic framework, finding I^*, i.e., the closest possible text to I, can be stated as finding the argument that maximizes the probability of recovering

the original input given the noisy text, that is

$$\underset{text}{\operatorname{argmax}} P(text|noisy)$$

The Noisy Channel Model is formalized by using the Bayes Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

such that recovering the original text is done by modeling the probability of the noisy text given the input text, that is, $P(noisy|text)$. In the case of machine translation, this problem is usually exemplified by the task of translating from a foreign (or French) language sentence $\mathbf{f}$ into an English sentence $\mathbf{e}$. Given $\mathbf{f}$, we seek the $\mathbf{e}$ that maximizes $P(\mathbf{e}|\mathbf{f})$, that is, the most likely translation:

$$\underset{\mathbf{e}}{\operatorname{argmax}} P(\mathbf{e}|\mathbf{f})$$

Using the Bayes Theorem, this problem can be decomposed as:

$$\underset{\mathbf{e}}{\operatorname{argmax}} P(\mathbf{e}|\mathbf{f}) = \underset{\mathbf{e}}{\operatorname{argmax}} \frac{P(\mathbf{f}|\mathbf{e})P(\mathbf{e})}{P(\mathbf{f})}$$

Since the source text $\mathbf{f}$ is constant across any alternative translation $\mathbf{e}$, it can be disregarded, and therefore:

$$\underset{\mathbf{e}}{\operatorname{argmax}} P(\mathbf{e}|\mathbf{f}) = \underset{\mathbf{e}}{\operatorname{argmax}} P(\mathbf{f}|\mathbf{e})P(\mathbf{e})$$

The generative models resulting from the decomposition of $P(\mathbf{e}|\mathbf{f})$ produce the two of the three fundamental components of a basic SMT system: the **translation model** $P(\mathbf{f}|\mathbf{e})$ and the **language model** $P(\mathbf{e})$. The translation model conditions the search for the best translation on the input text, while the language model searches for the best translation regardless of the input text. These two components are usually seen as proxies to what is considered to constitute a good translation, respectively: adequacy and fluency. The third major component is the **decoder**, a module that performs the search for the best translation e given the space of all possible translations (or a subset of it) based on the probability estimates $P(\mathbf{e})$ and $P(\mathbf{e}|\mathbf{f})$ (see Section 6).

The rationale for decomposing the problem into two other (simpler) problems using Noisy Channel and Bayes Theorem can perhaps be more clearly understood in the context of Speech Recognition. In that scenario, directly modeling the probability of a written text from speech would be complicated as it would require a large number of examples of written texts corresponding to speech inputs. On the other hand, modeling the probability of speech given written text is a simpler problem, once it is easier to gather examples of that (one could simply get people to read out loud written texts). The second component, that is, the probability of a given written text regardless of the speech input is also simple to model, since one can use a large set of written texts as examples. In what follows we describe how to estimate the two groups of probabilities from data.

2.1 Language Model

The language model tries to estimate the likelihood of a given sentence translation in the target language: the more common the translation, the more likely it will be that it is a good translation, particularly in terms of fluency, since the source sentence is not taken into account. In SMT, this is done by counting the relative number of occurrences of the sentence in a (preferably very large) monolingual corpus of the target language.

Formally, the language model component $P(\mathbf{e})$ for a sentence with m words is defined as the joint probability of a sequence of all words in that sentence:

$$P(\mathbf{e}) = P(w_1, w_2, ..., w_m)$$

The *chain rule* can then be applied to decompose such joint probability in a series of conditional probabilities:

$$P(\mathbf{e}) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2)P(w_4|w_1w_2w_3)...P(w_m|w_1...w_{m-1})$$

Maximum Likelinood Estimates Each component in the $P(\mathbf{e})$ formula decomposed by chain rule, that is, the probability of a word w given a number of previous words, is estimated using Maximum Likelihood Estimation (MLE), that is, as the count of occurrences of the complete sequence, divided by the count of the conditional sequence. For example:

$$P(w_3|w_1w_2) = \frac{count(w_1w_2w_3)}{count(w_1w_2)}$$

N-gram Language Models Given the variability of human language, the chances of finding significant occurrences of a given new sentence to translate even in a very large corpus are very small. In fact, it is very possible that not a single occurrence of a new long sequence of words will have been seen in the corpus, and in that case $P(\mathbf{e})$ will be 0, and so will $P(\mathbf{e}|\mathbf{f})$, since its equation is a product of $P(\mathbf{e})$. Therefore, instead of looking for a complete sentence, one usually counts occurrences of parts of such sentence, more specifically, *n-grams*, or sequences of up to n words. The larger the n, the more information about the context of the specific sequence (larger discrimination). The smaller the n the more cases will have been seen in the training data, and therefore the better the statistical estimates (more reliability). In practice, n varies according to the size of the corpus: the larger the corpus, the longer the n-grams that can be reliably counted. Most current open-domain systems consider n between 3 and 7.

N-gram language models are formalized by applying the Markov assumption that one can approximate the probability of a word given its entire history by computing the probability of a word given the last few words. For example, a *bigram language model* would consider only one previous word:

$$P(\mathbf{e}) = P(w_1)P(w_2|w_1)P(w_3|w_2)P(w_4|w_3)...P(w_m|w_{m-1})$$

while a *trigram language model* would consider two previous words:

$$P(\mathbf{e}) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2)P(w_4|w_2w_3)...P(w_m|w_{m-2}w_{m-1})$$

Smoothing Even short sequences of words like bigrams or trigrams can be rare or inexistent in the monolingual corpus used to estimate the probabilities. To avoid 0-counts and a $P(\mathbf{e}) = 0$ as a consequence, probability estimates are usually *smoothed*. Several **smoothing** strategies can be used for guaranteeing non-zero probabilities even to previously unseen sentences. The simplest one is called *add-one smoothing* and consists in adding one to all the counts of n-grams. For example, the MLE counts for unigrams and bigrams become, respectively:

$$P(w_i) = \frac{count(w_i) + 1}{N + V}$$

and

$$P(w_j|w_i) = \frac{count(w_i w_j) + 1}{count(w_i) + V}$$

where N = number of tokens in the corpus, and V = vocabulary or number of types, that is, all different words seen in the corpus. Better strategies include *interpolation* and *backoff* models.

Perplexity If a given translation sentence for which $P(\mathbf{e})$ is being computed is a long one, $P(\mathbf{e})$ will be the product of many small numbers, resulting in a much smaller final number, which can be difficult to read and process due to underflow problems. A common way to compare language model scores for different translation sentences is to compute the *perplexity* of such sentence as:

$$-\frac{1}{N}\log_2(P(\mathbf{e}))$$

where N is the number of words in the translation sentence. By normalizing the value according to the size of the sentence, the perplexity of a given system remains more or less constant regardless of the different sizes of sentences to be translated. As $P(\mathbf{e})$ increases, perplexity decreases: the lower the perplexity, the better the language model.

Although recent work has tried to improve language models by considering linguistic information, for example, the syntax-based language models [10], the language model component is usually a simple, token-based model, in most SMT systems. Off-the-shelf language modeling toolkits like SRILM [40] are used by most SMT systems.

2.2 Translation Model - Word-Based Models

Given the input sentence $\mathbf{f}$, the translation model aims to estimate a general model of $P(\mathbf{f}|\mathbf{e})$ by looking at a parallel corpus with examples of translations from $\mathbf{f}$ to $\mathbf{e}$. We assume here that such corpus has already been aligned at the sentence level by using some standard algorithm like the one in [15]. Extracting probability estimates $P(\mathbf{f}|\mathbf{e})$ for the whole sentences $\mathbf{f}$ and $\mathbf{e}$ is not feasible, for the same sparsity reasons as explained for language models, that is, it is unlikely

that we can find enough occurrences of complete sentences aligned to each other. We therefore need to work on shorter portions of the sentences.

We start with a simple translation model that translates words in isolation, that is, a word-by-word translation model. This can be done by using word alignment methods that will be described in Section 3. Here we assume we have the basic component produced by a word alignment method: a translation table, or **t-table**, which shows, for a given word in **f**, all its possible translations in **e** with their corresponding probability estimates, extracted from a parallel sentence-aligned corpus. For example, Table 1 shows some hypothetical entries for the English source word *take* with its lexical translations into Portuguese and corresponding translation probabilities.

Table 1. Example of t-table entries for the English-Portuguese translation of *take*

e	$P(e\|f)$
pegar	0.4
tomar	0.2
levar	0.1
receber	0.15
aceitar	0.075
tirar	0.05
tomada	0.001
...	...

Ideally, given sentence **f** to be translated, each of its single words will have at least one entry in the translation table (usually many). According to the simplest *translation model*, which actually consists of the *word alignment* model **IBM Model 1**, the best translation **e** will be the one that maximizes the lexical alignment **a** between $\mathbf{f} = (f_1, ..., f_J)$ and $\mathbf{e} = (e_1, ..., e_I)$, that is:

$$P(\mathbf{f}|\mathbf{e}) = \prod_{j=1}^{J} \sum_{a_{j=0}}^{I} P(a_j) P(f_j|e_{a_j})$$

where each source word f_j corresponds to one and only one target word (or $NULL$) and the target words are unconstrained, that is, each can link to an arbitrary number of words (including zero). The alignment variable **a** is assigned a value a_1^J, where each element a_j represents the position of the target word e_{a_j} to which f_j corresponds. By make the assumption that each variable a_j is independent, the optimal alignment **a** is given by:

$$\mathbf{a} = \underset{a_1^J}{\operatorname{argmax}} \prod_{j=1}^{J} P(a_j) P(f_j|e_{a_j})$$

IBM Model 1 can thus be used as a translation model, but it does not produce very good translations. In the next section we will see better word alignment models. As it will be discussed in Section 4, in the state-of-the-art SMT systems are based on phrase translation where phrases are extracted from word alignment models that build on IBM Model 1.

As we will see in Section 5, although the Noisy Channel Model was used as main underlying framework in the first SMT systems, in most recent systems the translation problem is modeled using probabilities extracted from a parallel corpus in both directions, that is, one can use both the probability of the source text given the target text and the probability of the target text given source text: $P(\mathbf{e}|\mathbf{f})$ and $P(\mathbf{f}|\mathbf{e})$, and these can be extracted using the same procedures.

3 Word Alignment

Word alignment constitutes the basis for SMT. Five basic models are normally used for this purpose, they are commonly called IBM Models 1-5, as they were proposed by IBM researchers [5, 6].

3.1 IBM Model 1

The simplest model, IBM Model 1, was outlined in Section 2.2. We now outline how the actual lexical translation probabilities are learned from sentence-aligned parallel text. This is done by using the **Expectation Maximization** (EM) algorithm. The EM algorithm works as follows [21, p.88]:

1. Initialize the model, typically with uniform distributions. This means that each input word f in the parallel corpus is assumed to be translated with equal probability into any output word e.
2. Apply the model to the data (**expectation** step). This means applying the current probability distribution to align the words.
3. Learn the model from the data (**maximization** step). This means collecting counts given the current model applied to the data to estimate an improved model, that is, a model considering the actual frequencies of different possible alignments.
4. Iterate steps 2 and 3 until convergence.

Details on the mathematical framework of the use of EM for word alignment are given in [21]. As a simple example to illustrate how EM works, consider the following parallel corpus of three sentence pairs when translating from German to English [21, p.91]: (*das haus, the house*),(*das buch, the book*),(*ein buch, a book*). Table 2 shows the application of IBM Model 1 EM training.

IBM Model 1 is a model of *lexical translation* and has many flaws, mainly because it is very week in deadline with reorderings (all reorderings are equally likely), or adding or dropping words. The other IBM models present the following advances on top of IBM Model 1:

Table 2. Example of application of IBM Model 1 EM training, taken from [21, p.91]

e	f	Initial	1st iter.	2nd iter.	3rd iter.	...	Final
the	das	0.25	0.5	0.6364	0.7479	...	1
book	das	0.25	0.25	0.1818	0.1208	...	0
house	das	0.25	0.25	0.1818	0.1313	...	0
the	buch	0.25	0.25	0.1818	0.1208	...	0
book	buch	0.25	0.5	0.6364	0.7479	...	1
a	buch	0.25	0.25	0.1818	0.1313	...	0
book	ein	0.25	0.5	0.4286	0.3466	...	0
a	ein	0.25	0.5	0.5714	0.6534	...	1
the	haus	0.25	0.5	0.4286	0.3466	...	0
house	haus	0.25	0.25	0.5714	0.6534	...	1

- IBM Model 2: adds absolute alignment model;
- IBM Model 3: adds fertility model;
- IBM Model 4: adds relative alignment model;
- IBM Model 5: fixes deficiency.

3.2 IBM Model 2

According to IBM Model 1, the translation probabilities of the target words in any order are all the same. Alignment models from IBM Model 2 deal with **reorderings** in a better way, by assuming that words that follow each other in the source language will also follow each other in the target language. This is done by using an explicit model for alignment based on the **positions** of the source and target words. The translation of a foreign source word in position i to a target word in position j is modeled by an alignment probability distribution

$$a(i|j, l_e, l_f)$$

where l_f is the length of the input sentence $\mathbf{f}$ and l_e is the length of the target sentence $\mathbf{e}$. The translation under IBM Model 2 is a two-step process with a lexical translation step (IBM Model 1) and an alignment step.

3.3 IBM Model 3

IBM Model 3 introduces the notion of **fertility** to cover for the fact that source words may be aligned to a specific number of target words. Often one word in the source is aligned to one word in the target (fertility = 1) , but it also happens that one source word can be aligned to n multiple target words (fertility = n), or even zero target words (fertility = 0). The fertility of source words is modeled directly with a probability distribution:

$$n(\phi|f)$$

for each foreign word f, this probability distribution indicates how many $\phi = 0, 1, 2...$ target words it usually translates to.

Fertility deals explicitly with dropping source words by allowing $\phi = 0$. The $NULL$ token is introduced to allow accounting for words in the target that have no correspondent in the source. One could model the fertility of the $NULL$ token for **insertion** of words by the conditional distribution $n(\phi|NULL)$. However, the number of inserted words depends on the sentence length, so the $NULL$ insertion is modeled in a separate step. This results in four steps for the translation process according to IBM Model 3:

- Fertility is modeled by the distribution $n(\phi|f)$;
- $NULL$ insertion is modeled by the probabilities p_1 (one $NULL$ token after each word) and $p_0 = 1 - p_1$ (no $NULL$ token);
- Lexical translation is handled by IBM Model 1;
- **Distortion** is modeled in a similar way as the reordering in IBM Model 2, with a probability distribution $d(j|i, l_e, l_f)$.

IBM Model 3 is already a powerful model for word-by-word translation, as it accounts for translation of words (t-table), reordering (distortion), insertion of words ($NULL$ insertion), dropping of words (words with fertility 0) and one-to-many translations (fertility > 0).

3.4 IBM Model 4

IBM Model 4 further improves IBM Model 3 by providing a better formulation of the distortion probability distribution $d(j|i, l_e, l_f)$. In IBM Model 3, for large source and target sentences (therefore large l_e and l_f), the estimates for movement will be sparse and not very realistic. In the translation process, large phrases tend to move together; words that are adjacent in the source tend to be next to each other in the target. IBM Model 4 introduces a **relative distortion** model. In such model, the placement of the translation of a source word is based on the placement of the translation of the preceding word. See [21, p.107-110].

3.5 IBM Model 5

According to IBM Models 3-4, it is possible for multiple target words to be placed in the same position, that is, nothing prohibits the placement of a target word into a position that has already been filled. In other words, some impossible alignments have positive probabilities. IBM Model 5 fixes this problem, i.e., eliminates **deficiency**. It does so by keeping track of the number of vacant word positions and allowing for placement only into these positions. The distortion model is therefore based on such vacancies.

In practice all IBM Models are relevant for SMT, since the final word alignment can be produced iteratively starting from Model 1 and finishing with Model 5. For more details about the IBM Models, see [19]. All IBM models are implemented in Giza++ [32], a word alignment toolkit used by most SMT systems. Several extensions of the IBM models have been proposed, particularly popular ones are based on Hidden-Markov Models [43].

4 Phrase-Based Models

When translating a sentence, it is common for contiguous sequences of words to translate as a unit. To account for that, current SMT systems are not based on word-by-word translation, but on the translation of phrases. The translation model is therefore based on phrases as units, as opposed to words [24]. A **phrase** is simply a contiguous sequence of words, usually not linguistically motivated. The notions of $NULL$ translation and fertility are not necessary anymore: each source phrase is non-empty an translates to exactly one non-empty target phrase. However, the phrases are not necessarily equal in length.

This model allows taking into account the local context of the words being translated. The more parallel training data available, the longer phrases one can model, and therefore the higher the chances of good quality translations.

A phrase-based translation model yields much better translations than word-based models. The basic model is yet a very simple one, consisting of three main steps:

1. The source sentence is split into *phrases*.
2. Each phrase is translated in isolation based on probabilities estimates for phrases.
3. The translated phrases are permuted into their final order.

The source sentence can be split in many different phrases. The choice of phrases is guided by the existing phrases in a **phrase translation table**. In what follows we describe a simple phrase-based translation model consisting of phrase probabilities extracted from corpus and a basic reordering model, and how to extract the phrases to build a phrase-table.

4.1 Translation Model - Phrase Translation Models

The mathematical formulation for phrase-based models is the same as that for word-based models, that is:

$$\underset{\mathbf{e}}{\operatorname{argmax}}\, P(\mathbf{e}|\mathbf{f}) = \underset{\mathbf{e}}{\operatorname{argmax}}\, P(\mathbf{f}|\mathbf{e})P(\mathbf{e})$$

However, for the phrase-based models, $P(\mathbf{f}|\mathbf{e})$ is further decomposed into:

$$P(\bar{f}_1^I|\bar{e}_1^I) = \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i) d(\text{start}_i - \text{end}_{i-1} - 1)$$

Where the source sentence $\mathbf{f}$ is broken up into I phrases $\bar{f}_i$, each $\bar{f}_i$ is translated into a target phrase $\bar{e}_i$ and the phrase probability is represented by $\phi(\bar{f}_i|\bar{e}_i)$.

In the formula, $d(\text{start}_i - \text{end}_{i-1} - 1)$ represents a **distance-based reordering model**. In such reordering model, the reordering of a phrase is relative to the previous phrase: start_i is the position of the first word of the source phrase that translates into the ith target phrase; end_i is the position of the last word of that source phrase.

The reordering distance, computed as ($\text{start}_i - \text{end}_{i-1} - 1$), is the number of words skipped (forward or backward) when taking source words out of sequence. For example, if two contiguous source phrases are translated in sequence, than start_i=$\text{end}_{i-1} + 1$, that is, the position of the first word of phrase i is the next after the position of the last word of the previous phrase. In that case, the reordering cost will be zero, i.e., a cost os $d(0)$ will be applied to that phrase. This model therefore penalizes movements of phrases over large distances.

While the phrase translation probabilities are learned from data (see Section 4.2), instead of estimating the reordering probability from data as well, the reordering is handled by a pre-defined model: an exponentially decaying cost function:

$$d(x) = \alpha^{|x|}, \text{ in this case: } d = \alpha^{|\text{start}_i - \text{end}_{i-1} - 1)|}$$

with a value for the parameter $\alpha \in [0, 1]$.

4.2 Learning a Phrase Translation Table

Similar to the *t-table* in word-based models, the phrase-based models use a probabilistic dictionary extracted from parallel corpora. However, in phrase-based models, the entries in such dictionary are not only word pairs, but also phrase pairs of (in principle) any length. For example, Table 3 shows some entries for the English phrase *of course* and its translations into Portuguese.

Table 3. Example of phrase-table entries for English-Portuguese translation of *of course*

e	$P(e\|f)$
certamente	0.25
naturalmente	0.25
, certamente ,	0.15
, naturalmente ,	0.1
como era de se esperar	0.02
...	...

The most common way to extract such phrases from corpora, together with their translation probabilities, is by applying a set of heuristics to word-aligned parallel corpora. The word-alignment can be produced using IBM Models as explained in Section 3. The heuristics try to extract **phrase pairs** which are **consistent** with the word alignment. For example, consider the word alignments produced for the English-German sentence pair in Figure 2, in both directions and then their intersection/union.

Since the parallel corpus can be handled in both directions ($\mathbf{f} \rightarrow \mathbf{e}$ and $\mathbf{e} \rightarrow \mathbf{f}$), it is trivial to generate word alignments in both directions. By intersecting such two alignments, one can get a high-precision alignment with high-confidence

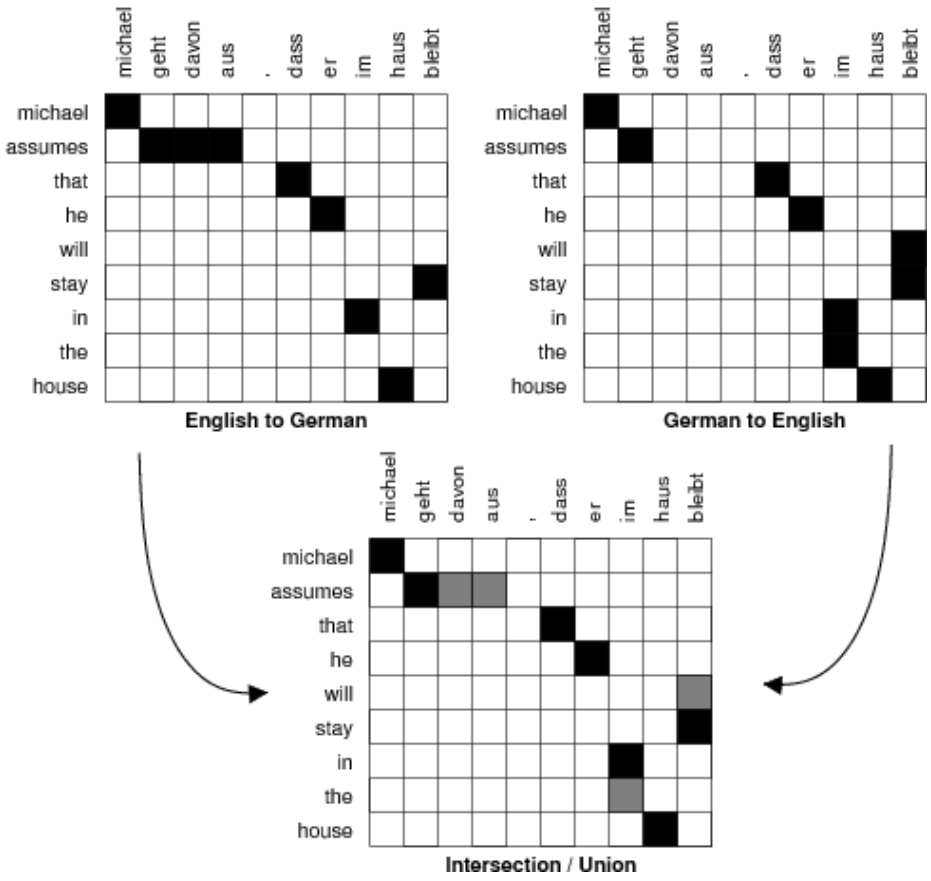

Fig. 2. Word alignments produced for the English-German sentence pair in both directions and their intersection (black points) and union (additional grey points) as taken from the Moses SMT system description [23]: http://www.statmt.org/moses/?n=Moses.Background.

alignment points. By taking the union of the two alignments, one can get a high-recall alignment with additional alignment points.

Phrase pairs which are **consistent** with such word alignments can then be extracted. A phrase pair $(\bar{f}, \bar{e})$ is consistent with an alignment A if all words $f_1, ..., f_n$ in $\bar{f}$ that have alignment points in A have such alignment points with words $e_1, ..., e_n$ in $\bar{e}$ and vice-versa, that is:

$$
\begin{aligned}
(\bar{e}, \bar{f}) \text{ consistent with } A \quad \Leftrightarrow \\
\forall e_i \in \bar{e} : (e_i, f_j) \in A \Rightarrow f_j \in \bar{f} \\
\text{AND } \forall f_j \in \bar{f} : (e_i, f_j) \in A \Rightarrow e_i \in \bar{e} \\
\text{AND } \exists e_i \in \bar{e}, f_j \in \bar{f} : (e_i, f_j) \in A
\end{aligned}
$$

The heuristics described here are those implemented in the Moses SMT system. The heuristics start with the intersection of the bidirectional alignments and then add additional alignment points. The idea is to loop over all possible target phrases and find the minimal source phrase that matches each of them in a consistent way.

Given an alignment matrix like the one shown in Figure 2, starting from the intersection of the two certain word alignments, new alignment points that exist in the union of two word alignments can be added. A new alignment point must connect at least one previously unaligned word. First, the algorithm expands to only directly adjacent alignment points. It checks for potential points starting from the top right corner of the alignment matrix, checking for alignment points for the first target word, then continues with alignment points for the second target word, and so on. This is done iteratively until no alignment point can be added anymore. In a final step, it adds non-adjacent alignment points, with otherwise the same requirements.

For example, given the intersection/union of the English $\leftrightarrow$ German alignment shown in Figure 2, the phrases in Figure 3 would be extracted [21, p. 134].

Fig. 3. Phrase pairs that can be extracted from the alignment in Figure 2 [21, p. 134].

michael – michael
michael assumes – michael geht davon aus ;
michael assumes – michael geht davon aus ,
michael assumes that – michael geht davon aus , dass
michael assumes that he – michael geht davon aus , dass er
michael assumes that he will stay in the house
– michael geht davon aus , dass er im haus bleibt
assumes – geht davon aus ;
assumes – geht davon aus ,
assumes that – geht davon aus , dass
assumes that he – geht davon aus , dass er
assumes that he will stay in the house – geht davon aus , dass er im haus bleibt
that – dass ;
that – , dass
that he – dass er ;
that he – , dass er
that he will stay in the house – dass er im haus bleibt ;
that he will stay in the house – , dass er im haus bleibt,
he – er
he will stay in the house – er im haus bleibt ;
will stay – bleibt
will stay in the house – im haus bleibt
in the – im
in the house – im haus
house – haus

Estimating Phrase Translation Probabilities Once all phrases consistent with word alignments are extracted from all sentence pairs in the parallel corpus, the phrase translation probabilities $\phi(\bar{f}_i|\bar{e}_i)$ are learned from the word-aligned parallel corpus using Maximum Likelihood Estimation (MLE), that is, counts of such phrase pairs in the corpus:

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

The phrase-based SMT models described so far (see Section 4) consist of three factors:

- the phrase translation table $\phi(\bar{f}|\bar{e})$: the foreign phrase $\bar{f}$ matches the English phrase $\bar{e}$;
- the reordering or distortion model d: the phrases are reordered appropriately; and
- the language model $P(\mathbf{e})$: the output is fluent English.

The product of these three model components forms a standard **phrase-based SMT model**:

$$\underset{\mathbf{e}}{\operatorname{argmax}} P(\mathbf{e}|\mathbf{f}) = \underset{\mathbf{e}}{\operatorname{argmax}} \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i) d(\text{start}_i - \text{end}_{i-1} - 1) \prod_{i=1}^{|\mathbf{e}|} P(e_i|e_1 ... e_{i-1})$$

5 Log-linear Models

State-of-the-art SMT systems like Moses [23] use an extended version of the phrase-based SMT models described in Section 4, namely a **log-linear model**. This extension allows weighting each of the standard phrase-based SMT model components according to their relevance: with uniform weights, we may produce adequate phrases but using an inadequate or non-fluent order, or vice-versa. Therefore, for a certain language-pair, text domain, etc., it might be better to give more weight to a specific component. Formally, this can be done by introducing **weights** $\lambda_\phi, \lambda_d, \lambda_{LM}$ to the phrase table, distortion and language model components, respectively:

$$\underset{\mathbf{e}}{\operatorname{argmax}} P(\mathbf{e}|\mathbf{f}) =$$

$$\underset{\mathbf{e}}{\operatorname{argmax}} \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i)^{\lambda_\phi} d(\text{start}_i - \text{end}_{i-1} - 1)^{\lambda_d} \prod_{i=1}^{|\mathbf{e}|} P(e_i|e_1 ... e_{i-1})^{\lambda_{LM}}$$

Log-linear models are very well known models in the Machine Learning community, and they have the following general form:

$$p(x) = \exp \sum_{i=1}^{n} \lambda_i h_i(x)$$

We can see that our phrase-based SMT model is an instance of such general model where:

- random variable $x = (e,\ f,$ start, end)
- $n = 3$ (number of feature functions h)
- feature function $h_1 = \log\ \phi$
- feature function $h_2 = \log\ d$
- feature function $h_3 = \log\ LM$

For further clarification, we can rewrite the phrase-based SMT model equation as:

$$P(e,a|f) = \exp\left[\lambda_\phi \sum_{i=1}^{I} \log \phi(\bar{f}_i|\bar{e}_i) + \lambda_d \sum_{i=1}^{I} \log d(a_i - b_{i-1} - 1) + \lambda_{LM} \sum_{i=1}^{|\mathbf{e}|} \log P(e_i|e_1 ... e_{i-1})\right]$$

In such framework, each sentence pair is seen as a vector of features, and the model as a set of corresponding feature functions. The feature functions are trained separately and then combined assuming that they are independent of each other (see Section 7). Besides the advantage of providing weights for different components, log-linear models allow a natural way to include additional model components in the form of feature functions. Additionally to the already described components, some of the commonly used feature functions in most standard phrase-based SMT systems are given in what follows.

5.1 Bidirectional Phrase Translation Probabilities

The use of the Noisy Channel Model results in considering the inverted conditional translation probability, that is, $P(\mathbf{f}|\mathbf{e})$ when translating from $\mathbf{f}$ into $\mathbf{e}$. However, in the training data it might happen that an unusual source phrase $\bar{f}$ is mistakenly aligned to a common target phrase $\bar{e}$, yielding a very high $\phi(\bar{f}|\bar{e})$. Whenever $\bar{f}$ is found in the test data, it is highly likely that it will be translated as $\bar{e}$, because it has a high translation probability and also a high language model probability, since it is a common target phrase.

In order to avoid such mistakes, a common strategy is to use the conditional probability of the phrases in the direct direction, i.e., $\phi(\bar{e}|\bar{f})$. In fact, most systems use both translation directions as feature functions and this usually outperforms using a single direction. The probability estimates $\phi(\bar{e}|\bar{f})$ are obtained in the very same way as for $\phi(\bar{f}|\bar{e})$, but using word alignments in the opposite direction.

5.2 Bidirectional Lexical Probabilities

Rare phrase pairs will have very high phrase translation probability: if they are seen only once or a few times, but are not aligned with anything else, then

$$\phi(\bar{e}|\bar{f}) = \phi(\bar{f}|\bar{e}) = 1$$

This often overestimates how reliable rare phrase pairs are, and can thus be problematic, especially if the phrases were extracted from noisy data.

In order to judge whether phrase pairs are reliable, one can decompose them into their word translations, that is, consider the **lexical weighting** of the phrases. Essentially, we can use the probability distributions for lexical translations, as in word-based translation models, for which the statistics are usually more reliable. Since phrases are actually extracted from word-alignments, we can easily consider the alignment of words within phrases.

Most lexical weighting feature functions therefore rely on word-based IBM models. Based on the word alignment within a phrase, we can compute the lexical translation probability of a phrase $\bar{e}$ given the phrase $\bar{f}$ by [21, p.139]:

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

In this feature function, each target word e_i is generated by aligned source words f_j with the word translation probability $w(e_i|f_j)$. If e_i is aligned to multiple source words, the average of the corresponding translation probabilities is taken. If e_i is not aligned to any source word, it is aligned to $NULL$, which is also factored in as a word translation.

Similar to the phrase translation probabilities, both translation directions can be considered: $lex(\bar{e}|\bar{f})$ and $lex(\bar{f}|\bar{e})$.

5.3 Word Penalty

To control the translation length in terms of the source text length and compensate for the bias of the language model towards shorter phrases (the fewer the n-grams, the higher the global language model score), it is common to introduce an explicit word penalty ω, where $\omega < 1$ increases the scores of shorter translations, and $\omega > 1$ increases the scores of longer translations.

5.4 Phrase Penalty

When translating a sentence, the first step, as previously discussed, is to segment the sentence into phrases. There are usually many possible segmentations (according to the multiple possible phrases in the phrase-table) and all segmentations are in principle equally likely: only the chosen phrase translations with their translation, reordering, and language model scores can (indirectly) determine the choices for the input sentence segmentation.

Depending on the data, longer (and fewer) phrases or shorter (and more) phrases may be more appropriate. A way to bias the choice towards longer or shorter phrases is to introduce a phrase penalty factor ρ. Likewise the word penalty, $\rho < 1$ increases the scores of fewer and longer phrases, and $\rho > 1$ increases the scores of shorter and more phrases. Intuitively, in general one prefers longer and fewer phrases, since they include more context and therefore are more likely to be correct. On the other hand, longer phrases are less frequent and can thus be less statistically reliable. In practice, the value for phrase penalty function is usually defined beforehand and its weight is learned according to the actual data.

5.5 Lexicalized Reordering

The standard distance-based reordering model uses a cost that is linear to the reordering distance, that is, skipping over two words will cost twice as much as skipping over one word. Such model generally penalizes movement: unless the language model prefers translations with placement of words in a different order in the translation, very little reordering is done in practice. While this model works well for close language pairs like French and English, it is not appropriate for distant pairs like Chinese and English.

More elaborate reordering models conditioned on specific factors can also be used, and different reordering probabilities for each phrase pair can also be learned from data, since some phrases are reordered more frequently than others. A **lexicalized reordering model** conditions reordering on the actual phrases. [21] proposes three reordering types:

- monotone order (m);
- swap with previous phrase (s); and
- discontinuous (d).

Formally, they introduce a reordering model p_0 that predicts an orientation type, m, s or d, given the phrase pair currently used in the translation:

$$\text{orientation} \in \{\text{m,s,d}\}$$

$$p_0(\text{orientation}|\bar{f}, \bar{e})$$

The probability distribution can be learned from the data using the word-alignment information: whenever a phrase pair is extracted, the orientation type of that specific occurrence is also extracted. The number of times each phrase pair is found with each of the three alignment orientation types is thus counted. The probability distribution p_0 is estimated based on this counts using MLE:

$$p_0(\text{orientation}|\bar{f}, \bar{e}) = \frac{count(\text{orientation}, \bar{e}, \bar{f})}{\sum_o count(o, \bar{e}, \bar{f})}$$

where o assumes the three orientation types.

A number of variations of this lexicalized reordering model can be considered. For example, one can combine the orientation information with:

- whether the model is conditioned on the foreign phrase $\bar{f}$ or on both $\bar{f}$ and $\bar{e}$; and
- whether the ordering is unidirectional or bidirectional: for each phrase, the ordering of itself with respect to the previous phrase is considered. In bidirectional models, the ordering of the next phrase with respect to the current phrase is also modeled.

The number of features created with a lexical reordering model depends on the types of combinations considered. For example, in Moses a "msd" model is implemented using three features, one each for the probability that the phrase is translated monotone, swapped, or discontinuous. If bidirectional models are used, then the number of features doubles - one for each direction.

6 Decoding

Given any type of SMT model (word- or phrase-based), the component responsible for finding the best scoring translation (the "argmax" in the equations previously given) is called **decoder**.

Decoding is a hard problem, as there is an exponential number of options. In fact, an exhaustive search in the entire search space would make the problem a NP-complete one [20]. In practice, most SMT systems implement heuristic search methods. With such methods, there is no guarantee that the "best" translation will be found, although usually the one found is either the best or close to it. The decoder in most SMT systems is based on implementations of a beam search process, similar to that proposed for speech recognition [18]. The target sentence is generated left to right in form of **hypotheses**. In what follows we describe a standard decoder used by systems like Moses. The description is strongly based on that by [21].

6.1 Decoding by Hypothesis Expansion

Given a source sentence, a number of phrase translations could be applied to translate it. Each such applicable phrase translation is called a **translation option**. This is illustrated in Figure 4, where a number of phrase translations for the Spanish source sentence "*Maria no daba uma bofetada a la bruja verde*" are given.

The search starts with an initial state where no foreign source words f are translated/covered and no English target words e have been generated. New states are created in the graph by extending the English output with a phrasal translation that covers some of the source words not yet translated. At every expansion, the current cost of the new state is the cost of the original state multiplied with the feature functions under consideration, for example, translation, distortion and language model costs of the added phrasal translation (a high cost is associated with a low probability). Final states in the search are hypotheses that cover all source words. Among these, the hypothesis with the lowest cost (highest probability) is selected as best translation.

Fig. 4. Translation options for the Spanish source sentence "*Maria no daba uma bofetada a la bruja verde*" into English, as taken from the Moses SMT system description: http://www.statmt.org/moses/?n=Moses.Background.

The initial decoding steps for the Spanish source sentence "*Maria no daba uma bofetada a la bruja verde*" into English, given the translation options shown in Figure 4, are illustrated in Figure 5.

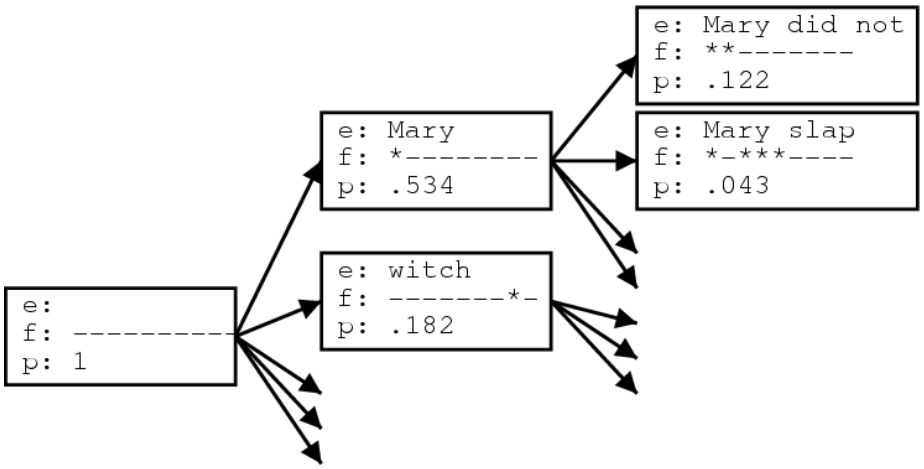

Fig. 5. Partial illustration of the search process for the Spanish source sentence "*Maria no daba uma bofetada a la bruja verde*" into English, given the translation options shown in Figure 4, as taken from the Moses SMT system description: http://www.statmt.org/moses/?n=Moses.Background.

In the search graph in Figure 5, starting from the initial hypothesis, the first expansion is the source word *Maria*, which is translated as *Mary*. This word is then marked as translated (with an asterisk). The initial hypothesis may also be expand by translating the source word *bruja* as *witch*. New hypotheses are then generated from the expanded hypotheses. For example, given the first expanded hypothesis once can generate a new hypothesis by translating *no* as *did not*. The first two foreign words *Maria* and *no* are marked as being already cov-

ered. Following the back pointers of the hypotheses we can read of the (partial) translations of the sentence.

The algorithm described so far can be used for exhaustively searching through all possible translations. However in practice it is optimized by discarding hypotheses that cannot be part of the path to the best translation and by using the concept of *comparable states* that allow defining a **beam** of good hypotheses and pruning out hypotheses that fall out of this beam.

6.2 Hypothesis Recombination

Hypothesis Recombination is a strategy for optimizing search that takes advantage of the fact that a given translation can be reached by different paths in the search graph. When that happens, the more costly hypothesis can be disregarded and excluded from the search graph.

Two hypotheses can be recombined not only if they are identical, but even if they *look similar*, for example if they agree in:

- the source words covered so far;
- the last n target words generated;
- the end of the last source phrase covered.

If there are two paths that lead to two hypotheses that agree in these properties, only the cheaper hypothesis is kept, e.g., the one with the least cost so far. The other hypothesis cannot be part of the path to the best translation, so it can be **safely** discarded.

Hypothesis recombination is very useful for reducing spurious ambiguities in the search, therefore allowing a more efficient search. However, it does not solve the decoding exponential complexity problem. This requires riskier methods for search space reduction, which will be discussed in what follows.

6.3 Stack-based Beam Search Decoding

A common way to organize hypothesis in the search space is by using "hypothesis stacks" based on the number of source words translated. One stack contains all the hypothesis that have translated one given source word, another stack contains all the hypothesis that have translated two source words in their path, and so on. Figure 6 shows the representation of hypothesis stacks.

The initial hypothesis is placed in the stack for hypotheses with 0 source words covered. New hypotheses are then generated by committing to phrasal translations that covered previously untranslated source words. Each derived hypothesis is placed in a stack based on the number of foreign words it covers. We proceed through these hypothesis stacks, going through each hypothesis in the stack, deriving new hypotheses for this hypothesis and placing them into the appropriate stack. A stack has limited space, so after a new hypothesis is placed into a stack, the stack may have to be pruned if it becomes too large. In the end, the best hypothesis from the stack covering all foreign words is taken as the best translation.

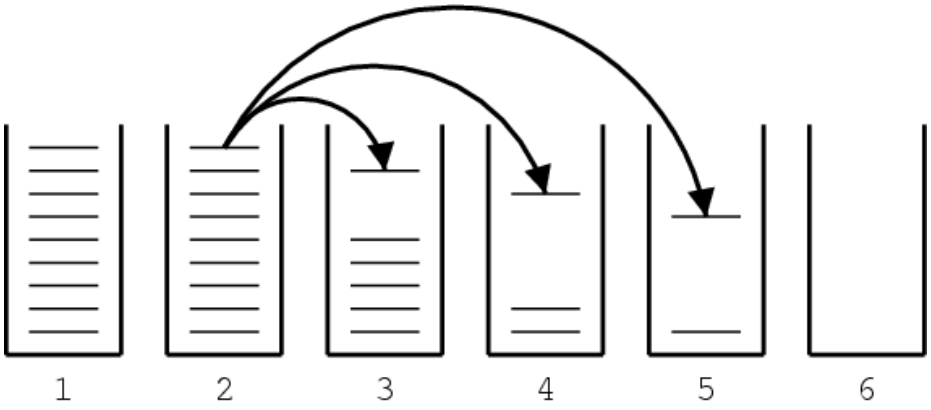

Fig. 6. Hypothesis stacks, as taken from the Moses SMT system description: http://www.statmt.org/moses/?n=Moses.Background.

The idea of using stacks for decoding is therefore to keep only a number of hypothesis that are "promising" by limiting the sizes of the stacks: is the stack gets too large, we prune the worst hypotheses from it. However, there is no guarantee that hypotheses pruned at early stage because they did not look promising are actually bad and would not redeem themselves later and lead to the best overall translation. This is due to the fact that we cannot compute the path in the graph until completion to find out the real cost of each hypothesis, since it is too expensive. Therefore, the decoder tries to make good guesses at which hypotheses are too bad to lead to the best overall translation.

The stack size can be defined by **threshold pruning** or **histogram pruning**. Histogram pruning keeps a certain number n of hypotheses (e.g., $n = 100$). A relative threshold cuts out a hypothesis with a probability less than a factor of the best hypotheses (e.g., $= 0.001$). According to [21, p.166], this threshold can be seen as a "beam of light that shines through the search space. The beam follows the (presumably) best hypothesis path, but with a certain width it also illuminates neighboring hypotheses that differ not too much in score from the best one", and therefore the term **beam search**.

Alternative decoding algorithms include A* search, beam search on coverage stacks, greedy hill-climbing and finite-state transducers (see [21]).

6.4 Future Cost Estimation

As previously described, the general procedure for pruning is to compare the hypotheses that cover the same number of foreign words and prune out the inferior hypotheses. If the judgment of what inferior hypotheses are is based on the cost of each hypothesis so far, it can bias the search to translating the easy part of the sentence first, that is, the search will prefer to start the sentence with the easy part and discount alternatives too early.

Therefore, the measure for pruning out hypotheses during the search has to take also into account an estimate of the **future cost**. The future cost estimation means estimating how hard it is to translate the untranslated part of the source sentence. It should favor hypotheses that already covered difficult parts of the sentence and have only easy parts left, and discount hypotheses that covered the easy parts first. Using the cost so far and the future cost estimation, we can prune out hypotheses that fall outside the beam.

Only some of the feature functions are usually considered for the computations of the future cost, mainly language model and translation model. For the translation model, for a given translation option, once can simply lookup its translation cost (i.e., probability) from the phrase-table. For the language model, since we do not know the preceding target words for a translation option, an approximation may be done by computing the language model score for the generated target words alone: if only one target word is generated, its unigram probability is used; if two words are generated, the unigram probability of the first word and the bigram probability of the second word are used, and so on.

This procedure can be used to calculate the cost for each translation option. The cheapest way to translate the sequence of source words is the one including the cheapest translation options. Therefore, one can approximate the cost for a path with a sequence of translation options by the product of the cost for each option.

6.5 N-Best Lists

For some applications, besides the actual best translation for a given source sentence, it can be helpful to have the second best translation, third best translation, and so on. A list of n "best" translations is called a **n-best list**. Post-edition systems using richer features than the ones common to an SMT system could be applied on n-best lists to re-score the translations, for example.

7 Parameter Estimation in Log-Linear Models

As discussed in Section 5, state-of-the-art SMT systems are based on discriminative phrase-based models, more specifically, log-linear models. Such models consist of a number of feature functions and weights associated with each feature function. In this section we describe the most common strategy to estimate such weights: the *Minimum Error-Rate Training* (MERT) algorithm [31].

MERT assumes that the best model is the one that produces the smallest overall error with respect to a given error function, that is, a function that evaluates the quality of the system translation. Therefore, the idea is to use the same function that will be considered for assessing the final translation evaluation in the training of the internal parameters of the system. As we will discuss in Section 8, most evaluation metrics are based on n-gram comparisons between the

system output (*hypothesis*) and one or more human (and therefore, good quality) translations (*references*). Hence MERT uses a development set containing a number of source sentences and their reference translations to, over several trials or iterations, optimize the feature functions weights such as to make the system translations as close as possible to the reference translations according to a certain evaluation metrics, commonly BLEU [34]. The trials are performed by considering the n-best lists produced by translating each input sentence using the current model.

Formally, given an error function $E(\hat{e}, e)$ defining the amount of error in some hypothesized translation $\hat{e}$ with respect to a known good actual translation e, then the objective function is:

$$\lambda_1^K = \underset{\hat{\lambda}_1^K}{\operatorname{argmin}} \sum_{(\mathbf{e},\mathbf{f}) \in C} E(\underset{\hat{e}}{\operatorname{argmax}} P_{\hat{\lambda}_1^K}(\hat{\mathbf{e}}|\mathbf{f}), e)$$

MERT works by iteratively generating random values for λ_1^K , which it then tries to improve by minimizing each parameter λ_k in turn while holding the others constant. At the end of this optimization step, the optimized λ_1^K yielding the greatest error reduction is used as input to the next iteration. Details about the algorithm can be found in [31] and [29, p.30-31].

8 Evaluation

The ability to automatically assess the quality of MT systems is an important aspect of this feld and has attracted considerable attention in recent years. Most research papers report results in terms of BLEU [34], although it has a number of well-known limitations and there are a number of alternative metrics. In this section we describe BLEU and a few other popular metrics for MT evaluation.

A common element of all automatic metrics is their use of human translations, i.e., the reference translations. The intuition behind such metrics is that automatic translations should resemble human translations. In order to measure the level of resemblance, these metrics consider the partial string matching between the machine and the reference translations. Single words or phrases can be considered as matching units. Most of these metrics can be applied when either a single reference or multiple references are available. Since a source sentence can usually have more than one correct translation, the use of multiple references allows avoiding biases towards one specific correct translation.

A few recent metrics consider also inexact matches, for example, using lemmas instead of word forms and considering paraphrases or entailments at the lexical or sentence level to compute matches.

8.1 Edit Rate Metrics

Edit or error rate measures estimate the amount of changes that must be applied to the automatic translation in order to transform it into a reference translation.

WER
Word Error Rate (WER) [41] and [30] is a measure based on the Levenshtein or edit distance [26]. It computes the minimum number of substitutions, deletions and insertions that have to be performed to convert the automatic translation into the human reference. This metric does not recognize word reorderings: a word that is translated correctly but in the wrong position will be penalized as a deletion in the machine translation and an insertion in the reference translation.

PER
Position-independent Word Error Rate (PER) [41] addresses a shortcoming of the WER: it does not penalize reorderings. PER compares the words in the machine and reference translations without taking into account the word order, that is, the machine and reference translations are considered as unordered sets.

TER and HTER
Translation Edit Rate (TER) [37] also measures the edit distance between machine and reference translations. The idea is to approximate the amount of post-editing that would be needed by a human to correct the translations. Besides the standard edit operations (insertions, deletions, and substitutions of single words), TER considers "shifts" of word sequences. All edits, including shifts of sequences of any length, have equal cost:

$$TER = \frac{\#edits}{\#reference\_words}$$

For multiple references, the number of edits is computed for each reference individually, and the one with the fewest number of edits is chosen. In the search process for the minimum number of edits, shifts are prioritized over other edits: a greedy search process tries to find the shifts that most reduce the number of insertions, deletions and substitutions.

Human-targeted Translation Edit Rate (HTER) is a semi-automatic variation of TER in which the references are build as human-corrected versions of the machine translations in order to guarantee that the edit rate is measured as the minimum of edits necessary to make the system output a fluent and adequate translation. The goal is to find the closest possible reference to the system output from the space of all possible fluent and adequate references. HTER was found to correlate better with human scores than TER [37].

TERp
Translation Edit Rate Plus (TERP) [38] is an extension of TER that (1) aligns words in the system output and reference translation not only if they are exact matches, but also when the words have the same stem or are (potentially) synonyms; (2) uses paraphrase probabilities to align phrasal substitutions in the system output and reference translation; and (3) optimize the costs of different types of edits to maximize the correlations with human judgments. Besides TER's standard edit operations (matches, insertions, deletions, substitutions and shifts), TERp adds three new operations: stem matches, synonym matches and

phrase substitutions. Two words match at the stem level if they share the same stem, and at the synonym level if they belong to the same synset in WordNet [14]. Two phrases are considered paraphrases if they belong to a pre-computed probabilistic database of paraphrases extracted using bilingual parallel corpora as described in [4].

The optimization process is a hill climbing search for costs that maximize the correlation of the global TERp score with human scores. This results in the same cost for each edit operation regardless of the actual words involved in the operation, except for phrase substitutions. The cost for phrase substitutions is given by:

$$cost(p_1, p_2) = w_1 + w_1 + edit(p_1, p_2) * (w_2 log Pr(p_1, p_2)) + w_3 Pr(p_1, p_2) + w_4)$$

where w_1, w_2, w_3, w_4 are the four free parameters of the edit cost, $edit(p_1, p_2)$ is the TERp edit cost to align p_1 to p_2 (excluding phrase substitutions) and $Pr(p_1, p_2)$ is the probability of paraphrasing p_1 as p_2.

Other differences with respect to TER are the fact that TERp is, by default, case insensitive, uses a list of stop words to constrain the shift operation (stop words and punctuations are only shifted if together with a non-stop word) and is capped at 1.0 (a system output cannot be more than 100% wrong).

8.2 Precision, Recall and F-measure Metrics

Precision-oriented metrics focus on lexical precision, i.e., the proportion of lexical units (typically n-grams of varying size) in the automatic translation which are covered by reference translations. Recall-oriented metrics focus on lexical recall, i.e., the proportion of lexical units in the reference translations which are covered by the machine translation. F-measure metrics combine lexical precision and recall.

BLEU and some of its variations

Bilingual Evaluation Understudy (BLEU) [34] is the most widely used metric for MT evaluation in general and computes lexical matching among n-grams, that is, matches at the n-gram level (from 1 up to some maximum n, usually 4) between the system and the reference translation. BLEU rewards translations whose word choice and word order is close similar to the reference. The number of n-gram matches normalized by the number of total n-grams in the system translation, and therefore it is a precision-oriented metric.

Let $count(ngram)$ be the count of $ngram$ in a given system output sentence and $count_{clip}(ngram)$ be the minimum number of times that $ngram$ appears (matches) in its reference translation. BLEU sums such clipped n-gram matches for all sentences in the test corpus, normalizing them by the number of candidate n-grams in the test corpus. For a given n, this results in the modified precision score, p_n, for the entire corpus:

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{ngram \in C} count_{clip}(ngram)}{\sum_{C' \in \{Candidates\}} \sum_{ngram' \in C'} count(ngram')}$$

Given multiple p_ns for different n-gram precisions, up to some maximum n, BLEU then averages them. The score for a given test corpus is the geometric mean of the modified n-gram precisions, p_n, using n-grams up to a length N (usually 4) and positive weights $w_n = N^{-1}$ summing to one:

$$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

Since the denominator contains the total number of hypothesis n-grams, to avoid giving preference to short translations, a brevity penalty is used:

$$BP = \begin{cases} 1, & \text{if } c > r \\ e^{(1-r/c)}, & \text{if } c \leq r \end{cases}$$

The brevity penalty penalizes system output sentences that are much shorter than the reference translation. It contrasts the total number of words c of all system outputs with reference length r. If multiple references are used, r is defined by summing the lengths of the closest reference to each system output.

A large number of improvements have been proposed to BLEU. [3], for example, weights the n-grams in BLEU according to frequency information computed based on a corpus of reference translations. The two weighting schemes are used, : *tf.idf* (Term Frequency * Inverse Document Frequency) and S-Score.

Using tf.idf, the n-grams are weighted in the following way:

$$tf.idf(i,j) = (1 + log(tf_{ij}))log(N/df_i)$$

where:

tf_{ij} is the number of occurrences of the word w_i in the document d_j;

df_i is the number of documents in the corpus where the word w_i occurs;

N is the number os documents in the corpus.

Using S-score, the n-grams are weighted in the following way:

$$S(i,j) = log\frac{(P_{doc(i,j)} - P_{corp-doc(i)}) * (N - df_i)/N}{P_{corp(i)}}$$

where:

$P_{doc(i,j)}$ is the relative frequency of the word w_i in the document d_j;

$P_{corp-doc(i)}$ is the relative frequency of the word w_i in the other documents of the corpus;

$(N - df_i)/N$ is the proportion of texts in the corpus where w_i does not occur;

$P_{corp(i)}$ is the relative frequency of the word w_i in the whole corpus.

Weighting n-grams has showed to improve correlation of BLEU scores with human judgements for adequacy

NIST

NIST is a recall-based version of BLEU by the National Institute of Standards and Technology [13]. It differs from BLEU in the way n-gram scores are averaged, the weights given to n-grams and the way the brevity penalty is computed. While BLEU relies on the geometric mean, NIST computes arithmetic mean. Moreover, while BLEU uses uniform weights for all n-grams, NIST weights more heavily n-grams which occur less frequently, as an indicator of their higher informativeness. For example, very frequent bi-grams in English like "of the" should be weighted low, since they are very likely to happen in many sentences and the match might therefore be merely by change. Finally, the modified brevity penalty minimizes the impact of small variations in the length of the system output on the final NIST score.

By computing the geometric mean, as opposed to NIST's arithmetic mean, BLEU is equally sensitive to proportional differences in matches for all values of N. Low number of matches for large Ns can therefore highly influence the final score.

The weights of n-grams are computed according to n-gram counts over the set of reference translations:

$$info(w_1...w_n) = \log_2 \left(\frac{count(w_1, ..., w_{n-1})}{count(w_1, ..., w_n)} \right)$$

Based on such weights, NIST's is computed as:

$$NIST = \sum_{n=1}^{N} \left\{ \frac{\sum_{allw_1...w_n thatmatch} info(w_1...w_n)}{\sum_{allw_1...w_n insysoutput} l} \right\} \cdot \exp \left\{ \beta \log^2 \left[\min \left(\frac{L_{sys}}{\overline{L}_{ref}}, l \right) \right] \right\}$$

where:

β is chosen to make the brevity penalty factor = 0.5 when the number of words in the system output is 2/3 of the average number of words reference translation;

$N = 5$;

L_{sys} is the number of words in system output translation;

$\overline{L}_{ref}$ is the average number of words in a reference translation, averaged over all reference translations.

Meteor

METEOR (Metric for Evaluation of Translation with Explicit Ordering) [25] is a metric that balances recall and precision of unigram matches. It includes a fragmentation score which accounts for word ordering, enhances token matching considering stemming and synonymy lookup and allows tuning to weight scoring components in order to optimize correlation with human judgments at document or sentence level.

The basic components of METEOR are: an F_{mean} and a discount factor Pen.

$$METEOR = (1 - Pen) \cdot F_{mean}$$

A matching algorithm performs word-to-word alignment between the system output and reference translations. The matches are computed against each reference separately and the best match is then selected. Different parameters of METEOR allow the unigram matches to be exact word matches, or generalized to stems, and synonyms. Resources are offered for stem – Porter stemmer [35] – and synonymy – WordNet [14] – for English only. Based on those matchings, precision and recall are calculated, resulting in the following F_{mean} metric:

$$F_{mean} = \frac{P \cdot R}{\alpha \cdot P + (1 - \alpha) \cdot R}$$

where: P = unigram precision, that is, fraction of words in the system output that match words in the reference, and R = unigram recall, that is, fraction of the words in the reference translation that match words in the system output.

The matching algorithm returns the fragmentation fraction, which is used to compute the discount factor Pen as follows. The sequence of matched unigrams between system output and reference translation is split into the fewest (and longest) possible chunks, where the matched words is each chunk are adjacent and in identical order in both strings. The number o chunks ch and the total number of matching words in all chunks m are then used to calculate the fragmentation count $frag = ch/m$. The discount factor Pen is then computed as:

$$Pen = \gamma \cdot frag^{\beta}$$

The parameters of METEOR determine the relative weight of precision and recall (α), the maximum penalty weight (γ), and the functional relation between the fragmentation and the penalty (β). These are originally set as $\alpha = 0.9$, $\beta = 3.0$ and $\gamma = 0.5$, based on previous experiments performed by METEOR's authors. However, such parameters can be optimized for better correlation with human judgments on a particular quality aspect (fluency, adequacy, etc.), data collection, language pair or evaluation unit (system, document or sentence level, e.g.). Experiments performed using simple hill climbing search was used to maximize correlation with human judgments for fluency and adequacy for different language pairs are described in [25], while experiments aiming to maximize human judgments on ranking translations are presented in [1].

8.3 Some recent developments in MT evaluation

Metrics proposed more recently go beyond lexical or n-gram matching by using linguistic knowledge, for example, by comparing the system output and reference translation in terms of their syntactic dependencies [16], or by checking whether one entails the other [33].

Other metrics are based on machine learning, where a number of features are combined in a model based on some training data, which shows to be particularly useful for sentence-level MT evaluation [2].

Finally, some metrics avoid the use of reference translations aiming at providing a sentence-level quality score for MT systems in use, that is, a score for translations produced given any new input sentence [39].

Other metrics, as well as results from competitions comparing several SMT systems and MT systems developed according to other paradigms can be found in the proceedings of the WMT workshops[1]: [7–9].

9 Advanced Topics

SMT is a very popular research topic at the moment and hence there are a number of new directions and advanced topics that could be mentioned. Perhaps the most prominent direction which has been studied for a number of years is that of introducing some structure into the flat phrase-based SMT models. Existing attempts include models that allow gaps in their phrases [36] and hierarchical models [12], in which Synchronous Grammars are used to structure the sentences or phrases by replacing words with their part-of-speech tags, for example. While syntactic information can be used to produce such structures [44, 45], the hierarchy does not need to be defined in terms of linguistic constituents. Hierarchical models are implemented in the Joshua SMT system [27].

Using syntax seems like a promising direction and it has been exploited not only directly in the translation model, but also in the language model [11], and for pre-processing (for example, making the syntax of the source language "look like" the syntax of the target language) or post-processing (for example, by re-ranking the n-best list according to the probabilities produced for each candidate by a statistical parser). The general motivations for the use of syntactic knowledge involving addressing several of the limitations of current phrase-based models, allowing:

- producing more grammatical output, particularly in the case of long-distance dependencies;
- having more accurate control over re-ordering;
- having more accurate control over use of function words;
- having better syntactic and semantic disambiguation;
- etc.

Adding other types of linguistic knowledge in the models is also an interesting topic. For example, Moses allows using linguistic knowledge at the word level, like part-of-speech tags or lemmas of the words, as factors within its **factored model** implementation [22].

Other topic of crescent interest is that of fully-discriminative models. While log-linear models are discriminative, some of the current feature functions are

[1] http://www.statmt.org/wmt10/

generative (e.g., $P(\mathbf{f}|\mathbf{e})$. Fully discriminative models like the ones proposed by [42, 28] are a promising topic, although they still have issues with scalability.

These and many other recent developments in the field of Statistical Machine Translation will be discussed during the tutorial.

References

1. Agarwal, A., Lavie, A.: Meteor, m-bleu and m-ter: evaluation metrics for high-correlation with human rankings of machine translation output. In: StatMT '08: Proceedings of the Third Workshop on Statistical Machine Translation. pp. 115–118 (2008)
2. Albrecht, J., Hwa, R.: A Re-examination of Machine Learning Approaches for Sentence-Level MT Evaluation. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL). pp. 880–887 (2007)
3. Babych, B., Hartley, A.: Extending the BLEU MT Evaluation Method with Frequency Weightings. In: 42nd Annual Meeting on Association for Computational Linguistics. p. 621 (2004)
4. Bannard, C., Callison-Burch, C.: Paraphrasing with bilingual parallel corpora. In: 43rd Annual Meeting on Association for Computational Linguistics. pp. 597–604 (2005)
5. Brown, P.F., Cocke, J., Pietra, S.A.D., Pietra, V.J.D., Jelinek, F., Lafferty, J.D., Mercer, R.L., Roossin, P.S.: A statistical approach to machine translation. Computational Linguistics 16(2), 79–85 (1990)
6. Brown, P.F., deSouza, P.V., Mercer, R.L., Pietra, V.J.D., Lai, J.C.: Class-based n-gram models of natural language. Computational Linguistics 18(4), 467–479 (1992)
7. Callison-Burch, C., Fordyce, C., Koehn, P., Monz, C., Schroeder, J.: (Meta-) Evaluation of Machine Translation. In: Proceedings of the ACL Workshop on Statistical Machine Translation. pp. 136–158 (2007)
8. Callison-Burch, C., Fordyce, C., Koehn, P., Monz, C., Schroeder, J.: Further meta-evaluation of machine translation. In: Proceedings of the Third Workshop on Statistical Machine Translation. pp. 70–106 (2008)
9. Callison-Burch, C., Koehn, P., Monz, C., Schroeder, J.: Findings of the 2009 Workshop on Statistical Machine Translation. In: Proceedings of the Fourth Workshop on Statistical Machine Translation. pp. 1–28 (2009)
10. Charniak, E., Knight, K., Yamada, K.: Syntax-based Language Models for Statistical Machine Translation. In: MT Summit IX (2003)
11. Charniak, E., Knight, K., Yamada, K.: Syntax-based language models for statistical machine translation. In: MT Summit IX (2003)
12. Chiang, D., Lopez, A., Madnani, N., Monz, C., Resnik, P., Subotin, M.: The hiero machine translation system: extensions, evaluation, and analysis. In: Human Language Technology and Empirical Methods in Natural Language Processing. pp. 779–786 (2005)
13. Doddington, G.: Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In: 2nd Human Language Technology Research. pp. 138–145. San Diego (2002)
14. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database (Language, Speech, and Communication). The MIT Press (1998)
15. Gale, W.A., Church, K.W.: A program for aligning sentences in bilingual corpora. In: 29th Annual Meeting on Association for Computational Linguistics. pp. 177–184 (1991)

16. Gimenez, J., Marquez, L.: A smorgasbord of features for automatic MT evaluation. In: 3rd Workshop on Statistical Machine Translation. pp. 195–198. Columbus, Ohio (2008)
17. Hutchins, J.: Milestones in machine translation. Part 1: How it all began in 1947 and 1948. Language Today (3), 22–23 (1997)
18. Jelinek, F.: Statistical Methods for Speech Recognition. MIT Press, Cambridge, MA (1998)
19. Knight, K.: A Statistical MT Tutorial Workbook (1999), http://www.isi.edu/natural-language/mt/wkbk.pdf
20. Knight, K.: Decoding complexity in word-replacement translation models. Computational Linguistics 25(4), 607–615 (1999)
21. Koehn, P.: Statistical Machine Translation. Cambridge University Press (2010)
22. Koehn, P., Hoang, H.: Factored translation models. In: Empirical Methods in Natural Language. Prague (2007)
23. Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E.: Moses: Open Source Toolkit for Statistical Machine Translation. In: 45th Annual Meeting of the Association for Computer Linguistics (2007)
24. Koehn, P., Och, F.J., Marcu, D.: Statistical phrase-based translation. In: Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology. pp. 48–54 (2003)
25. Lavie, A., Agarwal, A.: METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In: 2nd Workshop on Statistical Machine Translation. pp. 228–231. Prague, Czech Republic (2007)
26. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Tech. Rep. 8 (1966)
27. Li, Z., Callison-Burch, C., Dyer, C., Ganitkevitch, J., Khudanpur, S., Schwartz, L., Thornton, W.N.G., Weese, J., Zaidan, O.F.: Demonstration of joshua: an open source toolkit for parsing-based machine translation. In: ACL-IJCNLP 2009 Software Demonstrations. pp. 25–28 (2009)
28. Liang, P., Bouchard-Côté, A., Klein, D., Taskar, B.: An End-to-end Discriminative Approach to Machine Translation. In: 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics. pp. 761–768 (2006)
29. Lopez, A.: Statistical Machine Translation. ACM Computing Surveys 40(3), 1–49 (2008)
30. Nießen, S., Och, F.J., Leusch, G., Ney, H.: An evaluation tool for machine translation: Fast evaluation for mt research. In: 2nd International Conference on Language Resources and Evaluation (LREC). Athens, Greece (2000)
31. Och, F.J.: Minimum error rate training in statistical machine translation (2003)
32. Och, F.J., Ney, H.: A systematic comparison of various statistical alignment models. Computational Linguistics 29(1), 19–51 (2003)
33. Padó, S., Galley, M., Jurafsky, D., Manning, C.D.: Robust machine translation evaluation with entailment features. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP. pp. 297–305 (2009)
34. Papineni, K., Roukos, S., Ward, T., Zhu, W.: Bleu: a method for automatic evaluation of machine translation. In: 40th Meeting of the Association for Computational Linguistics. pp. 311–318. Morristown (2002)
35. Porter, M.F.: An algorithm for suffix stripping pp. 313–316 (1997)

36. Simard, M., Cancedda, N., Cavestro, B., Dymetman, M., Gaussier, E., Goutte, C., Yamada, K.: Translating with non-contiguous phrases. In: Empirical Methods in Natural Language. pp. 755–762. Vancouver (2005)
37. Snover, M., Dorr, B., Schwartz, R., Micciulla, L., Makhoul, J.: A study of translation edit rate with targeted human annotation. In: 7th Association for Machine Translation in the America. pp. 223–231. Cambridge, MA (2006)
38. Snover, M., Madnani, N., Dorr, B.J., Schwartz, R.: Fluency, Adequacy, or HTER?: Exploring Different Human Judgments with a Tunable MT Metric. In: 4th Workshop on Statistical Machine Translation. pp. 259–268 (2009)
39. Specia, L., Turchi, M., Cancedda, N., Dymetman, M., Cristianini, N.: Estimating the sentence-level quality of machine translation systems. In: 13th Meeting of the European Association for Machine Translation. Barcelona (2009)
40. Stolcke, A.: SRILM - An Extensible Language Modeling Toolkit. In: International Conference on Spoken Language Processing. pp. 901–904 (2002)
41. Tillmann, C., Vogel, S., Ney, H., Zubiaga, A., Sawaf, H.: Accelerated dp based search for statistical translation. In: European Conference on Speech Communication and Technology. pp. 2667–2670 (1997)
42. Tillmann, C., Zhang, T.: A Discriminative Global Training Algorithm for Statistical MT. In: 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics. pp. 721–728 (2006)
43. Vogel, S., Ney, H., Tillmann, C.: Hmm-based word alignment in statistical translation. In: 16th conference on Computational linguistics. pp. 836–841 (1996)
44. Yamada, K., Knight, K.: A syntax-based statistical translation model. In: 39th Annual Meeting on Association for Computational Linguistics. pp. 523–530 (2001)
45. Yamada, K., Knight, K.: A decoder for syntax-based statistical mt. In: 40th Annual Meeting on Association for Computational Linguistics. pp. 303–310 (2002)