A finite-state approach to phrase-based statistical machine translation

Jorge González
Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València
València, Spain
jgonzalez@dsic.upv.es

Abstract
This paper presents a finite-state approach to phrase-based statistical machine translation where a log-linear modelling framework is implemented by means of an on-the-fly composition of weighted finite-state transducers. Moses, a well-known state-of-the-art system, is used as a machine translation reference in order to validate our results by comparison. Experiments on the TED corpus achieve a similar performance to that yielded by Moses.

1 Introduction

Statistical machine translation (SMT) is a pattern recognition approach to machine translation which was defined by Brown et al. (1993) as follows: given a sentence $s$ from a certain source language, a corresponding sentence $t$ in a given target language that maximises the posterior probability $\Pr(t|s)$ is to be found. State-of-the-art SMT systems model the translation distribution $\Pr(t|s)$ via the log-linear approach (Och and Ney, 2002):

$$
\hat{t} = \arg \max_t \Pr(t|s)
$$

$$
\approx \arg \max_t \sum_{m=1}^M \lambda_m h_m(s, t)
$$

where $h_m(s, t)$ is a logarithmic function representing an important feature for the translation of $s$ into $t$, $M$ is the number of features (or models), and $\lambda_m$ is the weight of $h_m$ in the log-linear combination.

This feature set typically includes several translation models so that different relations between a source and a target sentence can be considered. Nowadays, these models are strongly based on phrases, i.e. variable-length $n$-grams, which means that they are built from some other lower-context models that, in this case, are defined at phrase level. Phrase-based (PB) models (Tomas and Casacuberta, 2001; Och and Ney, 2002; Marcu and Wong, 2002; Zens et al., 2002) constitute the core of the current state-of-the-art in SMT. The basic idea of PB-SMT systems is:

1. to segment the source sentence into phrases, then
2. to translate each source phrase into a target phrase, and finally
3. to reorder them in order to compose the final translation in the target language.

In a monotone translation framework however, the third step is omitted as the final translation is just generated by concatenation of the target phrases.

Apart from translation functions, the log-linear approach is also usually composed by means of a target language model and some other additional elements such as word penalties or phrase penalties. The word and phrase penalties allow an SMT system to limit the number of words or target phrases, respectively, that constitute a translation hypothesis.

In this paper, a finite-state approach to a PB-SMT state-of-the-art system, Moses (Koehn et al., 2007), is presented. Experimental results validate our work because they are similar to those yielded by Moses. A related study can be found in Kumar et al. (2006) for the alignment template model (Och et al., 1999).
2 Log-linear features for monotone SMT

As a first approach to Moses using finite-state models, a monotone PB-SMT framework is adopted. Under this constraint, Moses’ log-linear model is usually taking into account the following 7 features:

Translation features
1. Direct PB translation probability
2. Inverse PB translation probability
3. Direct PB lexical weighting
4. Inverse PB lexical weighting

Penalty features
5. PB penalty
6. Word penalty

Language features
7. Target language model

2.1 Translation features

All 4 features related to translation are PB models, that is, their associated feature functions \( h_m(s, t) \), which are in any case defined for full sentences, are modelled from other PB distributions \( \eta_m(\tilde{s}, \tilde{t}) \), which are based on phrases.

Direct PB translation probability

The first feature \( h_1(s, t) = \log P(t|s) \) is based on modelling the posterior probability by using the segmentation between \( s \) and \( t \) as a hidden variable \( \beta_1 \). In this manner, \( P(t|s) = \sum_{\beta_1} P(t|s, \beta_1) \) is approximated by \( P(t|s) \) by using maximization instead of summation: \( P(t|s) = \max_{\beta_1} P(t|s, \beta_1) \).

Given a monotone segmentation between \( s \) and \( t \), \( P(t|s, \beta_1) \) is generatively computed as the product of the translation probabilities for each segment pair according to some PB probability distributions:

\[
P(t|s, \beta_1) = \prod_{k=1}^{\beta_1} P(\tilde{t}_k | \tilde{s}_k)
\]

where \( \beta_1 \) is the number of phrases that \( s \) and \( t \) are segmented into, i.e. every \( \tilde{s}_k \) and \( \tilde{t}_k \), respectively, whose dependence on \( \beta_1 \) is omitted for the sake of an easier reading.

Feature 1 is finally formulated as follows:

\[
h_1(s, t) = \log \max_{\beta_1} \prod_{k=1}^{\beta_1} P(\tilde{t}_k | \tilde{s}_k) \quad (3)
\]

where \( \eta_1(\tilde{s}, \tilde{t}) = P(\tilde{t}|\tilde{s}) \) is a set of PB probability distributions estimated from bilingual training data, once statistically word-aligned (Brown et al., 1993) by means of GIZA++ (Och and Ney, 2003), which Moses relies on as far as training is concerned. This information is organized as a translation table where a pool of phrase pairs is previously collected.

Inverse PB translation probability

Similar to what happens with Feature 1, Feature 2 is formulated as follows:

\[
h_2(s, t) = \log \max_{\beta_2} \prod_{k=1}^{\beta_2} P(\tilde{s}_k | \tilde{t}_k) \quad (4)
\]

where \( \eta_2(\tilde{s}, \tilde{t}) = P(\tilde{s}|\tilde{t}) \) is another set of PB probability distributions, which are simultaneously trained together with the ones for Feature 1, \( P(\tilde{t}|\tilde{s}) \), over the same pool of phrase pairs already extracted.

Direct PB lexical weighting

Given the word-alignments obtained by GIZA++, it is quite straightforward to estimate a maximum likelihood stochastic dictionary \( P(t_i|s_j) \), which is used to score a weight \( D(\tilde{s}, \tilde{t}) \) to each phrase pair in the pool. Details about the computation of \( D(\tilde{s}, \tilde{t}) \) are given in Koehn et al. (2007). However, as far as this work is concerned, these details are not relevant.

Feature 3 is then similarly formulated as follows:

\[
h_3(s, t) = \log \max_{\beta_3} \prod_{k=1}^{\beta_3} D(\tilde{s}_k, \tilde{t}_k) \quad (5)
\]

where \( \eta_3(\tilde{s}, \tilde{t}) = D(\tilde{s}, \tilde{t}) \) is yet another score to use with the pool of phrase pairs aligned during training.

Inverse PB lexical weighting

Similar to what happens with Feature 3, Feature 4 is formulated as follows:

\[
h_4(s, t) = \log \max_{\beta_4} \prod_{k=1}^{\beta_4} I(\tilde{s}_k, \tilde{t}_k) \quad (6)
\]
where \( \eta_{\theta}(\hat{s}, \hat{t}) = I(\hat{s}, \hat{t}) \) is another weight vector, which is computed by using a dictionary \( P(s_j | t_i) \), with which the translation table is expanded again, thus scoring a new weight per phrase pair in the pool.

### 2.2 Penalty features

The penalties are not modelled in the same way. The PB penalty is similar to a translation feature, i.e. it is based on a monotone sentence segmentation. The word penalty however is formulated as a whole, being taken into account by Moses at decoding time.

#### PB penalty

The PB penalty scores \( e = 2.718 \) per phrase pair, thus modelling somehow the segmentation length. Therefore, Feature 5 is defined as follows:

\[
h_5(s, t) = \max_{\beta_5} \prod_{k=1}^{[\beta_5]} e \tag{7}
\]

where \( \eta_5(s, t) = e \) extends the PB table once again.

#### Word penalty

Word penalties are not modelled as PB penalties. In fact, this feature is not defined from PB scores, but it is formulated at sentence level just as follows:

\[
h_6(s, t) = \log e^{\|t\|} \tag{8}
\]

where the exponent of \( e \) is the number of words in \( t \).

### 2.3 Language features

Language models approach the a priori probability that a given sentence belongs to a certain language. In SMT, they are usually employed to guarantee that translation hypotheses are built according to the peculiarities of the target language.

#### Target language model

An \( n \)-gram is used as target language model \( P(t) \), where a word-based approach is usually considered. Then, \( h_T(s, t) = \log P(t) \) is based on a model where sentences are generatively built word by word under the influence of the last \( n - 1 \) previous words, with the cutoff derived from the start of the sentence:

\[
h_T(s, t) = \log \prod_{i=1}^{\|t\|} P(t_i | t_{i-n+1} \ldots t_{i-1}) \tag{9}
\]

where \( P(t_i | t_{i-n+1} \ldots t_{i-1}) \) are word-based probability distributions learnt from monolingual corpora.

### 3 Data structures

This section shows how the features from Section 2 are actually organized into different data structures in order to be efficiently used by the Moses decoder, which implements the search defined by Equation 2 to find out the most likely translation hypothesis \( t \) for a given source sentence \( s \).

#### 3.1 PB models

The PB distributions associated to Features 1 to 5 are organized in table form as a translation table for the collection of phrase pairs previously extracted. That builds a PB database similar to that in Table 1

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>( \eta_1 )</th>
<th>( \eta_2 )</th>
<th>( \eta_3 )</th>
<th>( \eta_4 )</th>
<th>( \eta_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>barato</td>
<td>low cost</td>
<td>1</td>
<td>0.3</td>
<td>1</td>
<td>0.6</td>
<td>2.718</td>
</tr>
<tr>
<td>me gusta</td>
<td>I like</td>
<td>0.6</td>
<td>1</td>
<td>0.9</td>
<td>1</td>
<td>2.718</td>
</tr>
<tr>
<td>es decir</td>
<td>that is</td>
<td>0.8</td>
<td>0.5</td>
<td>0.7</td>
<td>0.9</td>
<td>2.718</td>
</tr>
<tr>
<td>por favor</td>
<td>please</td>
<td>0.4</td>
<td>0.2</td>
<td>0.1</td>
<td>0.4</td>
<td>2.718</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td>2.718</td>
</tr>
</tbody>
</table>

Table 1: A Spanish-into-English PB translation table. Each source-target phrase pair is scored by all \( \eta \) models.

where each phrase pair is scored by all five models.

#### 3.2 Word-based models

Whereas PB models are an interesting approach to deal with translation relations between languages, language modelling itself is usually based on words. Feature 6 is a length model of the target sentence, and Feature 7 is a target language model.

#### Word penalty

Penalties are not models that need to be trained. However, while PB penalties are provided to Moses to take them into account during the search process (see for example the last column of Table 1, \( \eta_5 \)), word penalties are internally implemented in Moses as part of the log-linear maximization in Equation 2, and are automatically computed on-the-fly at search.

#### Target \( n \)-gram model

Language models, and \( n \)-grams in particular, suffer from a sparseness problem (Rosenfeld, 1996). The \( n \)-gram probability distributions are smoothed to be able to deal with the unseen events out of training data, thus aiming for a larger language coverage.
This smoothing is based on the backoff method, which introduces some penalties for level downgrading within hierarchical language models. For example, let $M$ be a trigram language model, which, as regards smoothing, needs both a bigram and a unigram model trained on the same data. Any trigram probability, $P(c|ab)$, is then computed as follows:

$$
\begin{align*}
\text{if} & \quad abc \in M: & P_M(c|ab) \\
\text{else if} & \quad bc \in M: & BO_M(ab)P_M(c|b) \\
\text{else if} & \quad c \in M: & BO_M(ab)BO_M(b)P_M(unk) \\
\text{else} & & BO_M(ab)BO_M(b)P_M(unk)
\end{align*}
$$

(10)

where $P_M$ is the probability estimated by $M$ for the corresponding $n$-gram, $BO_M$ is the backoff weight to deal with the unseen events out of training data, and finally, $P_M(unk)$ is the probability mass reserved for unknown words.

The $P(t_i|t_{i-n+1} \ldots t_{i-1})$ term from Equation 9 is then computed according to that algorithm above, given the model data organized again in table form as a collection of probabilities and backoff weights for the $n$-grams appearing in the training corpus. This model displays similarly to that in Table 2.

<table>
<thead>
<tr>
<th>$n$-gram</th>
<th>$P$</th>
<th>$BO$</th>
</tr>
</thead>
<tbody>
<tr>
<td>please</td>
<td>0.02</td>
<td>0.2</td>
</tr>
<tr>
<td>low cost</td>
<td>0.05</td>
<td>0.3</td>
</tr>
<tr>
<td>I like</td>
<td>0.1</td>
<td>0.7</td>
</tr>
<tr>
<td>that is</td>
<td>0.08</td>
<td>0.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 2: An English word-based backoff $n$-gram model. The likelihood and the backoff model score for each $n$-gram.

4 Weighted finite-state transducers

Weighted finite-state transducers (Mohri et al., 2002) (WFSTs) are defined by means of a tuple $(\Sigma, \Delta, Q, q_0, f, P)$, where $\Sigma$ is the alphabet of input symbols, $\Delta$ is the alphabet of output symbols, $Q$ is a finite set of states, $q_0 \in Q$ is the initial state, $f : Q \rightarrow \mathbb{R}$ is a state-based weight distribution to quantify that states may be final states, and finally, the partial function $P : Q \times \Sigma^* \times \Delta^* \times Q \rightarrow \mathbb{R}$ defines a set of edges between pairs of states in such a way that every edge is labelled with an input string in $\Sigma^*$, with an output string in $\Delta^*$, and is assigned a transition weight.

When weights are probabilities, i.e. the range of functions $f$ and $P$ is constrained between 0 and 1, and under certain conditions, a weighted finite-state transducer may define probability distributions. Then, it is called a stochastic finite-state transducer.

4.1 WFSTs for SMT models

Here, we show how the SMT models described in Section 3 (that is, the five $\eta$ scores in the PB translation table, the word penalty, and the $n$-gram language model) are represented by means of WFSTs.

First of all, the word penalty feature in Equation 8 is equivalently reformulated as another PB score, as in Equations 3 to 7:

$$
\eta_6(s, t) = \log e_{|t|} = \log \max_{\beta_6} \prod_{k=1}^{|\beta_6|} e_{|t_k|}
$$

(11)

where the length of $t$ is split up by summation using the length of each phrase in a segmentation $\beta_6$. Actually, this feature is independent of $\beta_6$, that is, any segmentation produces the expected value $e_{|t|}$, and therefore the maximization by $\beta_6$ is not needed.

However, the main goal is to introduce this feature as another PB score similar to those in Features 1 to 5, and so it is redefined following the same framework. The PB table can be now extended by means of $\eta_6(s, t) = e_{|t|}$, just as Table 3 shows.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>$\eta_1$</th>
<th>$\eta_2$</th>
<th>$\eta_3$</th>
<th>$\eta_4$</th>
<th>$\eta_5$</th>
<th>$\eta_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>barato</td>
<td>low cost</td>
<td>...</td>
<td>$e$</td>
<td>$e^2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>me gusta</td>
<td>I like</td>
<td>...</td>
<td>$e$</td>
<td>$e^2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>es decir</td>
<td>that is</td>
<td>...</td>
<td>$e$</td>
<td>$e^2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>por favor</td>
<td>please</td>
<td>...</td>
<td>$e$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: A word-penalty-extended PB translation table. The exponent of $e$ in $\eta_6$ is the number of words in Target.

Now, the translation table including 6 PB scores and the target-language backoff $n$-gram model can be expressed by means of (some stochastic) WFSTs.
Translation table

Each PB model included in the translation table, i.e., any PB distribution in \( \{ \eta_1(\tilde{s}, \tilde{t}), \ldots, \eta_6(\tilde{s}, \tilde{t}) \} \), can be represented as a particular case of a WFST. Figure 1 shows a PB score encoded as a WFST, using a different looping transition per table row within a WFST of only one state.

It is straightforward to see that the application of the Viterbi method (Viterbi, 1967) on these WFSTs provides the corresponding feature value \( h_m(s, t) \) for all Features 1 to 6 as defined in Equations 3 to 8.

Language model

It is well known that \( n \)-gram models are a subclass of stochastic finite-state automata where backoff can also be adequately incorporated (Llorens, 2000).

Then, they can be equivalently turned into transducers by means of the concept of identity, that is, transducers which map every input label to itself. Figure 2 shows a WFST for a backoff bigram model.

It is also quite straightforward to see that \( h_7(s, t) \) (as defined in Equation 9 for a target \( n \)-gram model where backoff is adopted according to Equation 10) is also computed by means of a parsing algorithm, which is actually a process that is simple to carry out given that these backoff \( n \)-gram WFSTs are deterministic.

To sum up, our log-linear combination scenario considers 7 (some stochastic) WFSTs, 1 per feature: 6 of them are PB models related to a translation table while the 7th one is a target-language \( n \)-gram model.

Next in Section 4.2, we show how these WFSTs are used in conjunction in a homogeneous framework.

4.2 Search

Equation 2 is a general framework for log-linear approaches to SMT. This framework is adopted here in order to combine several features based on WFSTs, which are modelled as their respective Viterbi score.

As already mentioned, the computation of \( h_m(s, t) \) for each PB-WFST, let us say \( T_m \) (with \( 1 \leq m \leq 6 \)), provides the most likely segmentation \( \beta_m \) for \( s \) and \( t \) according to \( T_m \). However, a constraint is used here so that all \( T_m \) models define the same segmentation \( \beta \):

\[
|\beta| > 0
\]

\[
s = \tilde{s}_1 \ldots \tilde{s}_{|\beta|}
\]

\[
t = \tilde{t}_1 \ldots \tilde{t}_{|\beta|}
\]

where the PB scores corresponding to Features 1 to 6 are directly applied on that particular segmentation for each phrase pair \((\tilde{s}_k, \tilde{t}_k)\) monotonically aligned. Equations 3 to 7 and 11 can be simplified as follows:

\[
\forall m = 1, \ldots, 6
\]

\[
h_m(s, t) = \log \max_{\beta} \prod_{k=1}^{\beta} \eta_m(\tilde{s}_k, \tilde{t}_k) \quad (12)
\]
Once their scores are transferred into them, the most conclusive models can also be merged by means of union, where they are reranked, to finally choose the best hypothesis as soon as they are partially produced.

Equation 13 represents a Viterbi-based composition framework where all the (weighted) models contribute to the overall score to be maximized, provided that the set of \( \lambda_m \)-weights is instantiated. Using a development corpus, the set of \( \lambda_m \)-weights can be empirically determined by means of running several iterations of this framework, where different values for the \( \lambda_m \)-weights are tried in each iteration.

\[
\hat{t} = \arg \max_t \sum_{m=1}^{7} \lambda_m h_m(s, t) \tag{13}
\]

5 Experiments

Experiments were carried out on the TED corpus, which is described in depth throughout Section 5.1. Automatic evaluation for SMT is often considered and we use the measures enumerated in Section 5.2. Results are shown and also discussed, in Section 5.3.

5.1 Corpora data

The TED corpus is composed of a collection of English-French sentences from audiovisual content whose main statistics are displayed in Table 4.

<table>
<thead>
<tr>
<th>Subset</th>
<th>English</th>
<th>French</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>Sentences</td>
<td>747.2K</td>
</tr>
<tr>
<td></td>
<td>Running words</td>
<td>24.6K</td>
</tr>
<tr>
<td></td>
<td>Vocabulary</td>
<td>9.2K</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.9K</td>
</tr>
<tr>
<td>Develop</td>
<td>Sentences</td>
<td>571</td>
</tr>
<tr>
<td></td>
<td>Running words</td>
<td>12.6K</td>
</tr>
<tr>
<td></td>
<td>Vocabulary</td>
<td>2.4K</td>
</tr>
<tr>
<td>Test</td>
<td>Sentences</td>
<td>641</td>
</tr>
<tr>
<td></td>
<td>Running words</td>
<td>12.6K</td>
</tr>
<tr>
<td></td>
<td>Vocabulary</td>
<td>2.4K</td>
</tr>
</tbody>
</table>

Table 4: Main statistics from the TED corpus and its split.

As shown in Table 4, develop and test partitions are statistically comparable. The former is used to train the \( \lambda_m \)-weights in the log-linear approach, in the hope that they can also work well for the latter.

5.2 Evaluation measures

Since its appearance as a translation quality measure, the BLEU metric (Papineni et al., 2002), which stands for bilingual evaluation understudy, has become consolidated in the area of automatic evaluation as the most widely used SMT measure. Nevertheless, it was later found that its correlation factor

\[ T \circ L \]

using the \( n \)-gram scores in \( L \) on the target hypotheses from \( T \) as soon as they are partially produced.
with subjective evaluations (the original reason for its success) is actually not so high as first thought (Callison-Burch et al., 2006). Anyway, it is still the most popular SMT measure in the literature.

However, the word error rate (WER) is a very common measure in the area of speech recognition which is also quite usually applied in SMT (Och et al., 1999). Although it is not so widely employed as BLEU, there exists some work that shows a better correlation of WER with human assessments (Paul et al., 2007). Of course, the WER measure has some bad reviews as well (Chen and Goodman, 1996; Wang et al., 2003) and one of the main criticisms that it receives in SMT areas is about the fact that there is only one translation reference to compare with. The MWER measure (Nießen et al., 2000) is an attempt to relax this dependence by means of an average error rate with respect to a set of multiple references of equivalent meaning, provided that they are available.

Another measure also based on the edit distance concept has recently arisen as an evolution of WER towards SMT. It is the translation edit rate (TER), and it has become popular because it takes into account the basic post-process operations that professional translators usually do during their daily work. Statistically, it is considered as a measure highly correlated with the result of one or more subjective evaluations (Snover et al., 2006).

The definition of these evaluation measures is as follows:

**BLEU**: It computes the precision of the unigrams, bigrams, trigrams, and fourgrams that appear in the hypotheses with respect to the $n$-grams of the same order that occur in the translation reference, with a penalty for too short sentences. Unlike the WER measure, BLEU is not an error rate but an accuracy measure.

**WER**: This measure computes the minimum number of editions (replacements, insertions or deletions) that are needed to turn the system hypothesis into the corresponding reference.

**TER**: It is computed similarly to WER, using an additional edit operation. TER allows the movement of phrases, besides replacements, insertions, and deletions.

### 5.3 Results

The goal of this section is to assess experimentally the finite-state approach to PB-SMT presented here. First, an English-to-French translation is considered, then a French-to-English direction is later evaluated.

On the one hand, our log-linear framework is tuned on the basis of BLEU as the only evaluation measure in order to select the best set of $\lambda_m$-weights. That is accomplished by means of development data, however, once the $\lambda_m$-weights are estimated, they are extrapolated to test data for the final evaluation.

Table 5 shows: a) the BLEU translation results for the development data; and b) the BLEU, WER and TER results for the test data. In both a) and b), the $\lambda_m$-weights are trained on the development partition. These results are according to different feature combinations in our log-linear approach to PB-SMT.

As shown in Table 5, the first experimental scenario is not a log-linear framework since only one feature, (a direct PB translation probability model) is considered. The corresponding results are poor and, judging by the remaining results in Table 5, they reflect the need for a log-linear approach.

The following experiments in Table 5 represent a log-linear framework for Features 1 to 6, i.e. the PB translation table encoded as a WFST $\mathcal{T}$, where different PB models are the focus of attention. Only the log-linear combination of Features 1 and 2

<table>
<thead>
<tr>
<th>Log-linear features</th>
<th>Develop BLEU</th>
<th>Test BLEU</th>
<th>Test WER</th>
<th>Test TER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (baseline)</td>
<td>8.5</td>
<td>7.1</td>
<td>102.9</td>
<td>101.5</td>
</tr>
<tr>
<td>1+2</td>
<td>4.0</td>
<td>3.0</td>
<td>116.6</td>
<td>115.6</td>
</tr>
<tr>
<td>1+2+3</td>
<td>22.7</td>
<td>18.4</td>
<td>66.6</td>
<td>64.4</td>
</tr>
<tr>
<td>1+2+3+4</td>
<td>22.8</td>
<td>18.5</td>
<td>66.3</td>
<td>64.2</td>
</tr>
<tr>
<td>1+2+3+4+5</td>
<td>22.7</td>
<td>18.8</td>
<td>65.2</td>
<td>63.2</td>
</tr>
<tr>
<td>1+2+3+4+5+6</td>
<td>23.1</td>
<td>19.1</td>
<td>65.9</td>
<td>63.8</td>
</tr>
<tr>
<td>1+7</td>
<td>24.6</td>
<td>20.5</td>
<td>65.1</td>
<td>62.9</td>
</tr>
<tr>
<td>1+2+7</td>
<td>25.5</td>
<td>21.3</td>
<td>63.7</td>
<td>61.6</td>
</tr>
<tr>
<td>1+2+3+7</td>
<td>25.9</td>
<td>22.2</td>
<td>62.5</td>
<td>60.4</td>
</tr>
<tr>
<td>1+2+3+4+7</td>
<td>26.3</td>
<td>22.0</td>
<td>63.4</td>
<td>61.3</td>
</tr>
<tr>
<td>1+2+3+4+5+7</td>
<td>26.4</td>
<td>22.1</td>
<td>63.1</td>
<td>61.0</td>
</tr>
<tr>
<td>1+2+3+4+5+6+7</td>
<td>27.0</td>
<td>21.8</td>
<td>64.4</td>
<td>62.2</td>
</tr>
<tr>
<td>Moses (1+...+7)</td>
<td>27.1</td>
<td>22.0</td>
<td>64.0</td>
<td>61.8</td>
</tr>
</tbody>
</table>

Table 5: English-to-French results for development and test data according to different log-linear scenarios. The set of $\lambda_m$-weights is learnt from development data for every feature combination log-linear scenario defined.
is worse than the baseline, which feeds us back on the fact that the $\lambda_m$-weights can be better trained, that is, the log-linear model for Features 1 and 2 can be upgraded until baseline’s results with $\lambda_2 = 0$.

This battery of experiments on Features 1 to 6 allows us to see the benefits of a log-linear approach. The baseline results are clearly outperformed now, and we can say that the more features are included, the better are the results.

The next block of experiments in Table 5 always include Feature 7, i.e. the target language model $L$. Features 1 to 6 are progressively introduced into $T$. These results confirm that the target language model is still an important feature to take into account, even though PB models are already providing a surrounding context for their translation hypotheses because translation itself is modelled at phrase level. These results are significantly better than the ones where the target language model is not considered. Again, the more translation features are included, the better are the results on the development data. However, an overtraining is presumably occurring with regard to the optimization of the $\lambda_m$-weights, as results on the test partition do not reach their top the same way the ones for the development data do, i.e. when using all 7 features, but when combining Features 1, 2, 3, and 7, instead. These differences are not statistically significant though.

Finally, our finite-state approach to PB-SMT is validated by comparison, as it allows us to achieve similar results to those yielded by Moses itself.

On the other hand, a translation direction where French is translated into English gets now the focus. Their corresponding results are presented in Table 6. A similar behaviour can be observed in Table 6 for the series of French-to-English empirical results.

### 6 Conclusions and future work

In this paper, a finite-state approach to Moses, which is a PB-SMT state-of-the-art system, is presented. A monotone framework is adopted, where 7 models in log-linear combination are considered: a direct and an inverse PB translation probability model, a direct and an inverse PB lexical weighting model, PB and word penalties, and a target language model.

Five out of these models are based on PB scores which are organized under a PB translation table. The word penalty can also be equivalently redefined as another PB model, similar to the five others, which allows us to constitute a translation model $T$ composed of six parallel WFSTs that are constrained to share the same monotonic bilingual segmentation.

A backoff $n$-gram model for the target language $L$ can be represented as an identity WFST where $P(t)$ is modelled on the basis of the Viterbi algorithm. The whole log-linear approach to Moses is attained by means of the on-the-fly WFST composition $T \circ L$.

Our finite-state log-linear approach to PB-SMT is validated by comparison, as it has allowed us to achieve similar results to those yielded by Moses. Monotonicity is an evident limitation of this work, as Moses can also feature some limited reordering. However, future work on that line is straightforward since the framework described in this paper can be easily extended to include a PB reordering model $R$, by means of the on-the-fly composition $T \circ R \circ L$.

### Acknowledgments

The research leading to these results has received funding from the European Union 7th Framework Programme (FP7/2007-2013) under grant agreement no. 287576. Work also supported by the EC (FEDER, FSE), the Spanish government (MICINN, MITYC, “Plan E”, grants MIPRCV “Consolider Ingenio 2010” and iTrans2 TIN2009-14511), and the Generalitat Valenciana (grant Prometeo/2009/014).

### Table 6: French-to-English results for development and test data according to different log-linear scenarios.

<table>
<thead>
<tr>
<th>Log-linear features</th>
<th>Develop BLEU</th>
<th>Test BLEU</th>
<th>WER</th>
<th>TER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (baseline)</td>
<td>7.1</td>
<td>7.4</td>
<td>101.6</td>
<td>100.0</td>
</tr>
<tr>
<td>1+2</td>
<td>4.1</td>
<td>3.5</td>
<td>117.5</td>
<td>116.0</td>
</tr>
<tr>
<td>1+2+3</td>
<td>24.2</td>
<td>21.1</td>
<td>58.9</td>
<td>56.5</td>
</tr>
<tr>
<td>1+2+3+4</td>
<td>24.4</td>
<td>20.8</td>
<td>58.0</td>
<td>55.7</td>
</tr>
<tr>
<td>1+2+3+4+5</td>
<td>24.9</td>
<td>21.2</td>
<td>56.9</td>
<td>54.8</td>
</tr>
<tr>
<td>1+2+3+4+5+6</td>
<td>25.2</td>
<td>21.2</td>
<td>57.1</td>
<td>55.0</td>
</tr>
<tr>
<td>1+7</td>
<td>24.7</td>
<td>22.5</td>
<td>60.0</td>
<td>57.7</td>
</tr>
<tr>
<td>1+2+7</td>
<td>26.0</td>
<td>23.2</td>
<td>58.8</td>
<td>56.5</td>
</tr>
<tr>
<td>1+2+3+7</td>
<td>28.5</td>
<td>23.0</td>
<td>56.1</td>
<td>54.0</td>
</tr>
<tr>
<td>1+2+3+4+7</td>
<td>28.4</td>
<td>23.1</td>
<td>56.0</td>
<td>53.8</td>
</tr>
<tr>
<td>1+2+3+4+5+7</td>
<td>28.8</td>
<td>23.4</td>
<td>56.0</td>
<td>53.9</td>
</tr>
<tr>
<td>1+2+3+4+5+6+7</td>
<td>28.7</td>
<td>23.8</td>
<td><strong>55.8</strong></td>
<td><strong>53.7</strong></td>
</tr>
<tr>
<td>Moses (1+...7)</td>
<td>28.9</td>
<td>23.5</td>
<td>55.8</td>
<td>53.6</td>
</tr>
</tbody>
</table>

These models can also be implemented by means of WFSTs on the basis of the Viterbi algorithm. The whole log-linear approach to Moses is attained by means of the on-the-fly WFST composition $T \circ L$. The whole log-linear approach to Moses is attained by means of the on-the-fly WFST composition $T \circ L$.
References


