Experimenting with Phrase-Based Statistical Translation within the IWSLT 2004 Chinese-to-English Shared Translation Task

Philippe Langlais, Michael Carl, Oliver Streiter

RALI/DIRO
University of Montreal
Canada
felipe@iro.umontreal.ca

IAI
Saarbrücken
Germany
carl@iai.uni-sb.de

NUK
National University of Kaohsiung
Taiwan
ostreiter@nuk.edu.tw

Abstract

This paper describes the system we built for the Chinese to English track of the IWSLT 2004 evaluation campaign. A one month effort was devoted to this exercise, starting from scratch and making use as much as possible of freely available packages. We show that a decent phrase-based translation engine can be built within this short time frame.

1. Introduction

Machine Translation is a very active field nowadays strongly anchored into a paradigm of performance. Evaluation exercises such as those conducted within the TIDES project are pushing system designers to constantly improve their systems. Currently, many of the top-performing systems are phrase-based statistical machine translation (SMT) engines. The fact that SMT systems are among the best ones in those evaluation exercises is not surprizing considering the peculiarities of the translation tasks considered. The popularity of phrase-based models (PBMs) in SMT is neither a surprise, since PBMs allow to a certain extent to cope with local word reordering across languages, as well as to account for local context modelling. [1] also credit PBMs for being somehow tolerant to tokenization errors, an interesting characteristic when dealing with languages such as Chinese, the source language under consideration in this study.

This effervescent activity comes with some bonus. Several freely available valuable packages (*e.g.* `Giza++` [2], `Pharaoh` [3], `SRILM` [4]) make possible the fast development of a phrase-based translation engine. Other packages allow to quickly evaluate a system according to a gold standard (*e.g.* MTEVAL `http://www.nist.gov/speech/tests/mt/mt2001/resource` and GTM `http://nlp.cs.nyu.edu/GTM`). This paper reports on the one month effort we spent building a system for the Chinese-to-English track of the IWSLT workshop, relying intensively on these packages.

2. Phrase-Based Models

Very recently, several authors [5, 6] proposed at the same time an astonishingly simple but powerful model which we designate hereafter as Flat Phrase-Based model (FPBM). A FPBM is simply a collection of pairs of sequences of words with one or several scores (or probabilities) attached to them. The main difference between a FPBM and an alignment template (AT) model [7] being that the former does not attempt to model internal reordering of phrases. Thus, FPBMs as such do not have generalization capabilities. Zens and Ney [8] give an experimental comparison of both models on three different test sets. On the German-English Verbmobil task, the AT engine outperforms the PB engine they tested, while on the other tasks — the Spanish-English Xerox task, and the French-English Hansards task — they observed the opposite. Tomàs et al. [9] recently revisited the AT model and report that combining it with a FPBM brings some improvements.

The recipe given in [5, 6] for acquiring a FPBM is simple: use a word-alignment to identify in an heuristic way an alignment relation at the so-called phrasal level. Both articles propose relative frequency counts to score each pair of phrases. Several authors noted that the relative frequency estimator is particularly inappropriate to the task, since many phrases (and especially long ones) are seen only few times, sometimes only once. [5, 10] proposed to score a pair of phrases according to an IBM-like word-based model, the lexicon probabilities $p(f|n)$ being learned by relative frequency over the word alignment set. This idea has also been tested by Vogel et al. [10]. Also, in [8], the authors propose to score a pair of phrases according to a smoothed probabilistic word bilingual lexicon. And Vogel et al. [1] demonstrated experimentally that rating phrases according to an information-based score yields noticeable improvements.

Other variations to the recipe mentioned above have been extensively investigated by the CMU team and summarized in [10]. They investigated variations on the way the word alignment is produced, considering for instance a bilingual bracketing alignment [11], an alignment technique also tried at the same workshop by [12].

Zhang et al. [13] proposed an alternative way to collect

phrases without requiring word alignment. They rely instead on point-wise mutual information between source and target words to identify phrases in both languages. This has a clear advantage over methods that rely on a monolingual segmentation step, followed by a bilingual mapping one, as for instance the one described in [14].

3. Our system

We developed a translation engine around the freely available package `Pharaoh` [3]. This package is provided with a binary file, as well as a carefully written user manual. The core of this decoder is a beam search engine optimizing a noisy channel model, as described in equation 1, where $s_1^I = s_1, \ldots, s_I$ stands for the best sequence of I phrases that fully cover the source sentence s.

$$\begin{aligned} \hat{e} &= \text{argmax}_e \, p(c|e) p_{lm}(e)^{\lambda_{lm}} \omega^{|e| \times \lambda_\omega} \\ &= \text{argmax}_{e,I} \, p_\phi(c_1^I|e_1^I) p_{lm}(e)^{\lambda_{lm}} \omega^{|e| \times \lambda_\omega} \end{aligned} \quad (1)$$

Here, an independence assumption is further assumed between phrases, and the transfer model p_ϕ is formulated as in equation 2, where ϕ is a FPBM, and d is a simple distortion model depending on a_i the starting position of the foreign phrase c_i, and b_{i-1} the ending position of the native phrase e_{i-1} (see [5] for more).

$$p(c_1^I|e_1^I) = \prod_{i=1}^{I} \phi(c_i|e_i)^{\lambda_\phi} d(a_i - b_{i-1})^{\lambda_d} \quad (2)$$

What really matters from the user point of view of this package, is the fact that the decoder takes as input:

- a pair of FPBMs, one for each direction, the direct model (in our case $\phi(e|c)$) being used for nbest-list rescoring, a functionality of `Pharaoh` we did not use in this study.
- a target language model (English), in the format output by the SRILM package [4],
- a set of weights applied in a log-linear fashion to the different models, namely: λ_ϕ, the weight given to the transfer model; λ_{lm}, the weight given to the language model, λ_ω, the word-penalty weight and λ_d, the weight given to the distortion model.

We trained the language models of the target part of the training corpus (20 000 English sentences) with the SRILM package[1]. In order to feed the phrase extractor, we first word-aligned the training bitext making use of the `Giza++` package. Since [5] shown that the degree of the IBM model from which the viterbi alignment is computed was not playing a crucial role, we used the viterbi approximation computed by `Giza++` for the IBM model 3 (training IBM model 4 is more demanding, since we need to train the word classes that are conditioning the distortion probabilities of this model).

The only thing we had really to implement was a prescription to get the FPBMs required by `Pharaoh`. This is described in the following section.

[1] We did not investigate the many smoothing options this package handles, but applied the setting recommended in [15].

4. Phrase extraction

We tried two kinds of strategies to compute the FPBMs. The first one, is directly following the approach described in [5, 6] and is detailed in section 4.1. The second one is a simple string-based approach described in section 4.2.

4.1. Word-alignment-based extractor

A nowadays standard practice among the PBM practitioners consists in aligning the bitext at the word level making use of word-alignment models trained in both directions (here C→E and E→C). This double alignment process makes senses since the underlying alignment model (most often an IBM model [16]) is not symmetrical. Two sets of links between words are then distinguished. We call $\mathcal{P}$ (for Precision) the set of links that are present in both alignment directions, and $\mathcal{R}$ (for Recall) the links that are present in at least one alignment (C→E or E→C). Note that $\mathcal{P} \subseteq \mathcal{R}$. The word alignment retained is constituted of the $\mathcal{P}$-links, as well as some $\mathcal{R}$-links in the neighborhood of $\mathcal{P}$.

We implemented a variant of this approach which is strongly inspired by [5, 6]. Although the principle is very straightforward, we did not find in those articles a precise enough description of the algorithm. So, for the sake of completeness, we report in algorithm 1 the pseudo-code of the variant that we implemented. Our algorithm works in 4 steps. First, the $\mathcal{P}$-links are considered (line 6), then extended by considering $\mathcal{R}$-links (lines 9-21). Third, independent boxes are collected (lines 24-33). An independent box $((x_1, x_2), (y_1, y_2))$ represents a region in the alignment matrix where none of the source words $S_{x_1}^{x_2}$ is aligned to a word not belonging to $T_{y_1}^{y_2}$ and vice-versa:

$$\begin{aligned} &\forall x \in [x_1, x_2], \forall y : \Re(x, y), y \in [y_1, y_2] \\ &\forall y \in [y_1, y_2], \forall x : \Re(x, y), x \in [x_1, x_2] \end{aligned} \quad (3)$$

where $\Re$ is an alignment relation made explicit by step 1 and 2 of the algorithm. The fourth and last step of the algorithm (lines 36-42) consists in electing pairs of phrases, any sequence of adjacent (on the source side) boxes.

The pseudo-code of our variant makes use of a data structure $T[x]$ (resp. $T[y]$) which stands for the target (resp. source) positions associated to the source (resp. target) position x (resp. y):

$$\begin{aligned} T[x] &= \{y \, : \, \Re(x, y)\}, \forall x \in [1, |S|] \\ T[y] &= \{x \, : \, \Re(x, y)\}, \forall y \in [1, |T|] \end{aligned} \quad (4)$$

We also need a few functions to simplify the description of the algorithm. The first function maintains the T structure during step 1 and 2 of the projection algorithm.

function $add(x,y)$ $\left\{ \begin{array}{l} T[x] \leftarrow T[x] \cup \{y\} \\ T[y] \leftarrow T[y] \cup \{x\} \end{array} \right.$

The second one, called during the extension stage verifies that (x, y) is a valid link to extend on:

function $neighbor(x,y)$
if $(x,y) \in \mathcal{R}$, $\not\in \mathcal{P}$ **then**
  if $T[x] = \{\}$ **or** $T[y] = \{\}$ **then**
    $a \leftarrow a \cup (x,y)$

The third function collects the pairs of phrases after checking some few length properties and is called during step 4 of the algorithm:

function $add(x_1, x_2, y_1, y_2)$
$x \leftarrow x_2 - x_1 + 1$
$y \leftarrow y_2 - y_1 + 1$
if $x \in [minLength, maxLength]$ **then**
  if $y \in [minLength, maxLength]$ **then**
    if $(max(x,y)/min(x,y)) \leq ratio$ **then**
      $res \leftarrow res \cup (S_{x_1}^{x_2}, T_{y_1}^{y_2})$

4.2. String-based extractor

The second phrase extractor performs simple string operations. It is intended to capture obvious redundancies at the sentence and phrasal level in the training corpus. It is based on the simplifying assumption that if two strings are in relation of translation and if part of them also are, then we can induce a specific translation relation between the other parts. This is the idea formulated in the algorithm 2.

In practice, we factor out the prefix and suffix test carried out in lines 10 and 11 of Algorithm 2 by sorting the training corpus using as sort key: a) the Chinese sentence, b) the English sentence, c) the inverted Chinese sentence and d) the inverted English sentence. Iterating from the top to the bottom of these lists, whenever a line contains it's preceding line, the preceding line is subtracted and the new pair of phrases added to the training corpus. The process was stopped when the productivity of the algorithm decreased, producing about 60 000 new pairs of phrases.

5. Phrase ranking

We examined two ways of scoring the pairs of phrases (s,t). Both are estimates of the conditional probability $p(t|s)$. The first estimator is relative frequency (equation 5) which, as mentioned earlier, largely overestimates the probability of rare phrases. Table 1 reports the frequency distributions of the pair of phrases observed for different settings on the training corpus (20 000 pairs of sentences). Approximatively 90% of the observed pairs appear only once in the training corpus, and around 70% of the parameters are set to unity by the relative frequency estimator.

An alternative is to resort to IBM model 1 [16] to score a pair. This is done by computing equation 6.

Algorithm 1 A Koehn-Tilmann-like variant for phrase extraction

Require: $\mathcal{P}, \mathcal{R}, minLength, maxLength, ratio$
Ensure: res contains all the pairs of phrases

1: *Initialization*
2: $res \leftarrow \{\}$
3: **for all** $x : 1 \rightarrow |S|$ **do** $T[x] \leftarrow \{\}$
4: **for all** $y : 1 \rightarrow |T|$ **do** $T[y] \leftarrow \{\}$
5:
6: *Step1: $\mathcal{P}$-projection*
7: **for all** $(x,y) \in \mathcal{P}$ **do** $add(x,y)$
8:
9: *Step2: Extension*
10: **for** $p : 1 \rightarrow 2$ **do**
11: **repeat**
12: $a \leftarrow \{\}$
13: **for** $s : 1 \rightarrow |S|$ **do**
14: **for all** $t \in T[s]$ **do**
15: **if** $p = 2$ **then**
16: neighbor(x-1,y-1); neighbor(x+1,y-1);
17: neighbor(x-1,y+1); neighbor(x+1,y-1);
18: **else**
19: neighbor(x-1,y); neighbor(x+1,y);
20: neighbor(x,y-1); neighbor(x,y+1);
21: **for all** $(x,y) \in a$ **do** $add(x,y)$
22: **until** $|a| = 0$
23:
24: *Step3: Collect independent boxes*
25: $b \leftarrow \{\}$
26: **for** $x : 1 \rightarrow |S|$ **do**
27: $X \leftarrow \{x\}; Y \leftarrow \{\}$
28: **repeat**
29: $X_m \leftarrow X; Y_m \leftarrow Y$
30: **for all** $x \in X$ **do** $Y \leftarrow Y \cup T[x]$
31: **if** Y != Y_m **then**
32: **for all** $y \in Y$ **do** $X \leftarrow X \cup T[y]$
33: **until** $X = X_m$ and $Y = Y_m$
34: $b \leftarrow b \cup \left\{ \begin{array}{l} (min\{x : x \in X\}, max\{x : x \in X\}), \\ (min\{y : y \in Y\}, max\{y : y \in Y\}) \end{array} \right\}$
35: $x \leftarrow max\{x : x \in X\} + 1$
36:
37: *Step4: Combine boxes*
38: **for** $i : 1 \rightarrow |b|$ **do**
39: **let** $((x_{m_i}, x_{M_i}), (y_{m_i}, y_{M_i})) = b_i$
40: $add(x_{m_i}, x_{M_i}, y_{m_i}, y_{M_i})$
41: **for** $j : i + 1 \rightarrow |b|$ **do**
42: **let** $((x_{m_j}, x_{M_j}), (y_{m_j}, y_{M_j})) = b_j$
43: **if** $x_{M_i} + 1 = x_{m_j}$ **then**
44: $add(x_{m_i}, x_{M_j}, y_{m_i}, y_{M_j})$

Algorithm 2 A String-based phrase extractor

Require: $\mathcal{T} = \{(E_i, C_i), i \in [1, |\mathcal{T}|]\}$, a training corpus
Ensure: res contains the pair of phrases

```
Initialization
res ← {}
for i : 1 → |T| do
    res ← res ∪ (E_i, C_i)

Applying compositionality
repeat
    if (E_1, C_1) ∈ res then
        if (E_2, C_2) ∈ res then
            if C_2 = C_1 α or C_1 = C_2 α then
                if E_2 = E_1 β or E_1 = E_2 β then
                    res ← res ∪ (β, α)
until convergence of res
```

$$p_{rel}(t|s) = \frac{|(t,s)|}{|t|} \quad (5)$$

$$p_{ibm}(t|s) = (|S|+1)^{-|T|} \prod_{i=1}^{|T|} \sum_{j \in [0,|S|]} p(t_i|s_j) \quad (6)$$

6. Corpora

During this exercise, we only used the corpora made available by the organizers, the characteristic of which are reported in Table 2. No pre-processing was done to try to reinforce the parallelism between the two languages. Neither did we try to account for class of tokens such as numbers or dates. We did not change either the tokenization provided, but did convert the English into lowercase. Punctuation marks were left as is in the corpora, but removed after translation, as required by the organizers. The TRAIN corpus was split into TRAIN-Q and TRAIN-A corpus, gathering interrogative and affirmative sentences respectively. See section 7.5 for the motivations behind this split.

The CSTAR corpus contains 506 Chinese sentences with 15 to 16 English reference translations[2]. It was available four weeks before the official test and was used to gain some expertise on the phrase-based models.

Table 1: Frequency distribution of pairs of phrases observed in the training corpus for different values of $minLength$ and $maxLength$. A ratio of 2.0 was applied. %f1, %f2 and %f3+ stand for the percentage of parameters (pairs of phrases) seen 1, 2 or at least 3 times in the TRAIN corpus. $\%p = 1$ stands for the percentage of parameters that have a relative frequency of 1.

min	max	$\|model\|$	%f1	%f2	%f3+	$\%p = 1$
1	8	166 481	90.6	4.9	4.5	74.6
2	8	153 512	92.7	4.3	3.0	78.5
2	4	73 369	87.0	7.1	5.9	68.7

Table 2: Main characteristics of the corpora used in this study.

		Chinese		English	
corpus	|pair|	tokens	words	tokens	words
TRAIN	20 000	182 904	7 643	188 935	7 181
TRAIN-A	11 884	112 000	6 456	116 343	6 008
TRAIN-Q	8 116	70 904	4 024	72 592	3 900
CSTAR	506	3 515	870	–	–
TEST	500	3 794	893	–	–

7. Experiments

In this section, we summarize the experiments we did with the above described phrase-based acquisition methods. We ran the decoder on the CSTAR corpus. The best parameter setting would be the one we would use for translating the official test set. As discussed in few moments, many things have been tried, some useful, some not, and much script code has been churned out, with some inevitable bugs (recall that we devoted one month for the full exercise).

Repeating the experiments with a less stringent schedule (that is, after the official test), we detected and corrected several bugs. The results that are reported here are mainly those we measured after the competition (after correcting the few bugs we found), but we also report in section 7.6 the results of the translations we officially submitted.

One of the goal of the IWSLT exercise was to evaluate the salience of different evaluation metrics. For our own purpose, we computed a subset of those metrics: the BLEU[3] and NIST scores using the *mteval* script. We also computed MWER and MSER measures with an in-house tool as follow:

$$\text{MWER}(T_1^S, R_1^N) = \frac{100}{S} \sum_{i=1}^{S} \min_{r \in [1,N]} ed(T_i, R_i^r) \quad (7)$$

$$\text{MSER}(T_1^S, R_1^N) = \frac{100}{S} \sum_{i=1}^{S} \delta\left(\min_{r \in [1,N]} ed(T_i, R_i^r) \right) \quad (8)$$

with

$$\delta(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{otherwise} \end{cases} \quad (9)$$

where T_1^S is the set of S candidate translations to be evaluated, T_i being the candidate translation of the ith source

[2] For some reasons, a few sentences had only 15 translations. Therefore, the reference we consider in this study is constituted only of the first 15 translations provided for each source sentence.

[3] For readability reasons, we report BLEU% scores, that is, the BLEU score multiplied by 100.

sentence, R_1^N stands for the set of N reference translations to the S source sentence; R_i^r being the rth reference translation of the ith source sentence.

$ed(a, b)$ is the classic *edit distance* between a and b (counting 1 for insertion, substitution and deletion) normalized by the total number of operations involved to map a into b (counting as well the identity operation).

7.1. From word-based models to phrase-based models

The first experiment we conducted was to compare the performance of a word-based translation engine to the performance of `Pharaoh` seeded with a FPBM acquired by the approach described in section 4.1 ($ratio = 2$, $maxLength = 8$, and $minLength = 1$). Each parameter of this model was estimated by relative frequency.

The word-based SMT engine is an extension to a trigram language model of the inverted dynamic programming approach described in [17]. This decoder which is designed for an IBM model 2 had been implemented before the IWSLT exercise. The performance of both the word-based and the phrase-based engines are reported in Table 3.

As can be observed, the performances of both decoders do elicit differences, notably on the NIST score, but not as much as we would have expected at first, especially if we consider the simplicity of the word-based model (WBM) embedded into our word-based engine.

As a raw check that our word-based engine was not too buggy, we ran the `Pharaoh` decoder with the transfer parameters of the IBM model 2 converted into the appropriate format. The results are reported in line 3 of Table 3.

Table 3: Performances measured on the CSTAR corpus of the word-based engine (line 1), and the phrase-based engine (line 2). Line 3 shows the performance of the `Pharaoh` decoder fed with an IBM model 2 transfer model

engine	NIST	BLEU%	mWER	mSER
ibm2+3g	5.0726	26.57	60.56	94.47
`Pharaoh`	5.5646	26.16	59.70	94.27
wbm by `Pharaoh`	4.8417	15.54	64.95	97.63

7.2. Tuning the decoder parameters

One important thing we learned is that significant improvements — as measured by the automatic metrics we computed — can be obtained by tuning the engine adequately. What we call tuning here is the choice of the decoder parameters (or meta parameters) we can control via the built-in options of `Pharaoh`. This is done without modifying the model themselves (translation or language models), but finding the appropriate value of: λ_{lm}, the weight given to the language model (see equation 1); λ_ω the word penalty (see equation 1); λ_ϕ, the weight given to the translation model (see equation 2), and λ_d, the weight given to the distortion model: (see equation 2).

Since we had only a few parameters to tune, we applied a poor man's strategy: a) sample uniformly the range of each parameter, b) generate all the combinations of parameter values, and c) translate the full test corpus with all these configurations generated. Clearly, there are more clever ways to tune the decoder, but we had at our disposal around 30 processors, and thanks to the decoder speed, it was manageable to tune the decoder for a set of models within a few hours of computation[4]. We made the arbitrary choice of optimizing the performance as measured by the NIST metric.

We report in Table 4 the influence of tuning for a translation model obtained by the FPBMs used in the previous experiment (relative frequency estimator, $ratio = 2, maxLength = 8, minLength = 1$).

The first line of this table shows the performance we obtained with the default configuration of `Pharaoh`. The third line shows the configuration of the decoder which yields the largest NIST score. Clearly, tuning is very important, since we obtained a relative gain over the default configuration (line 1) of 23%. If we had to tune only one parameter, then the word penalty would be the one to tune, since it brings alone a relative improvement of 14%, which represents 61% of the higher gain observed. The performance of the decoder, tuned for the word penalty only is reported in the second line of Table 4.

From now on, the performance reported for a given set of models are those obtained after tuning the NIST metric.

Table 4: Performances measured on the CSTAR corpus without tuning (line 1), after tuning the word penalty weight (line 2), and after the tuning of all the parameters (line 3).

λ_d	λ_ϕ	λ_w	λ_{lm}	NIST	BLEU%	mWER	mSER
1	1	0	1	5.5646	26.16	59.70	94.27
1	1	-1.5	1	6.3470	25.63	58.93	94.27
.2	.9	-1.5	.8	6.8401	28.44	56.25	94.07

7.3. Merging different FPBMs

We observed that merging a FPBM with a word-based model enlarges its coverage. Merging two FPBMs (a word-based model can be seen as a special case of a FPBM) $p_1(t|s)$ and $p_2(t|s)$ was done by copying the parameters $p_i(t|s)$ when s was not in the other model. In cases both models had s in common, the union of the target phrases associated to s by both models was considered. In cases where both models had the same pair of phrases (s, t), its score was averaged over the two models. The parameters were finally normalized so

[4] Actually, an important part of the tuning process is devoted to computing the NIST scores with the MTEVAL script, as well as loading the parameters into `Pharaoh`

that $\sum_t p(t|s) = 1, \forall s$.

Table 5 shows (line 2) that merging the FPBM described above ($ratio = 2, minLength = 1, maxLength = 8$) with a word-based model resulting from IBM training yields a relative improvement over the phrase-based model alone of 3.7%. Extending the resulting model with the pairs of phrases obtained by the methodology described in section 4.2 only slightly improves the performance (line 3). Actually this gain is probably due to the fact that the TRAIN and TEST corpora share some source sentences, and that the second phrase acquisition method includes the pairs of sentences of the training corpus as parameters.

The improvement observed by merging the WBM with the FPBM is somehow surprising considering the very harsh way we did the merging. We tried a cleaner linear combination of both models without better improvements.

Table 5: Performances of the merged model measured on the CSTAR corpus. SBPM stands for the string-based phrase model extracted by the approach described in section 4.2.

config	$\|p\|$	NIST	BLEU%	mWER	mSER
FPBM		6.8401	28.44	56.25	94.07
+ WBM		7.0766	31.38	54.88	93.28
+ SPBM		7.0926	31.78	54.56	92.69

7.4. Scoring phrases with IBM model 1

We report in this section the influence of the way a pair of phrases is scored within the translation model. The baseline model we consider here is the merged FPBM of the last section (line 3 of Table 5), a model of 306 585 parameters trained by relative frequency. Rating these parameters by equation 6 yields a relative improvement in the NIST score of 3% (line 2 of Table 6).

For a given set of phrase parameters, `Pharaoh` allows to provide several scores, in which case, a specific weight must be given to each model. We tuned a model with two scores, one computed by relative frequency, the other one computed by equation 6. Thus, the tuning of the decoder was involving 5 parameters. The result of this experiment can be seen in line 3 of Table 6. A slight increase of the NIST metric as well as an improvement in BLEU% score can be observed.

Inspecting the last model, we observed that around half the parameters (150 127) where set to 1 by each score. This is due to the fact that up to now, we systematically normalized the parameters so that the stochastic constraints hold. We tried a last model where the IBM model 1 score was not normalized in the cases where only one target phrase was associated to a given source sentence (we also tried with less success a model where no normalization was carried out at all for the IBM score). The result of this experiment is shown in line 4 of Table 6: an improvement is observed on the NIST score, but at the detriment of the word error rate.

Table 6: Influence of the function used to score a parameter. `relfreq` stands for the relative frequency estimator, `ibm` for the IBM model 1 scoring (equation 6), and `pn-ibm` for the partially normalized IBM score.

model	NIST	BLEU%	mWER	mSER
`relfreq`	7.0926	31.78	54.56	92.69
`ibm`	7.3067	32.98	53.86	92.49
`relfreq&ibm`	7.3118	34.48	52.73	91.90
`relfreq&pn-ibm`	7.4219	34.6	53.02	91.70

7.5. Specific models

Based on the observation that around 40% of the training sentences were interrogatives, we investigated whether splitting the training corpus into two parts (interrogative sentences versus affirmative ones) and training separately on these two corpora could lead to some improvements. Splitting the corpus was done by explicitly looking for the presence (or absence) of the question mark word at the end of the Chinese sentences.

At translation time, the sentences ending with the question mark were translated first with the specific *question* configuration. The other sentences were translated with the *affirmative* configuration. The two translation sets were then merged appropriately to get a final translation of the source test corpus.

We first tried to train two different translation models. None of the trials we made resulted in an improvement. The fact that `Pharaoh` does not lend itself to combining different translation models that do not have the exact same set of parameters might be a reason for our lack of success in this experiment. We found however that combining a specific language model with the one trained on the full corpus leads to a slight improvement.

This time, we conducted separately two tunings — one for affirmative sentences, one for questions — over the 6 parameters now controlling the decoder: two parameters for the language models (one for the specific model, one for the general one), two parameters for the translation model (one for the relative frequency score, one for the IBM score), one parameter for the distortion model, and one for the word penalty. The best improvement on the NIST score is reported in line 4 of Table 7. We observe that it is not correlated with improvements in the other metrics.

7.6. The translations we submitted before the deadline

According to the experiences we conducted on the CSTAR corpus, we identified several variants that we wanted to submit. They are enumerated in increasing order of their expected merit as estimated by the NIST metric[5].

[5] At the time of the exercise, the performance of the QA-model we had was significantly higher than that of other variants we considered.

Table 7: Performances of the merged model measured on the CSTAR corpus. A and Q stand for the performance measured on the subset of respectively affirmative and interrogative sentences of the test corpus.

config	NIST	BLEU%	mWER	mSER
relfreq&ibm	**7.3118**	**34.48**	**52.73**	**91.90**
A	7.1862	34.21	53.12	91.18
Q	6.4995	34.92	52.12	93.00
specific-lm	**7.4702**	**33.64**	**53.27**	**91.90**
A	7.3229	33.66	53.08	90.85
Q	6.7010	33.58	53.55	93.50

ibm2+3g our word-based translation engine,

straight the model obtained by the extraction method described in section 4.1,

merge the best model obtained by merging word associations and phrase associations acquired by the two methods we described,

QA the engine combining a general language model with one specifically trained on the interrogative (resp. affirmative) sentences of the TRAIN corpus,

The variant we submitted for manual evaluation was the *QA* one, the last one in this list. We also submitted a translation involving manual intervention in order to measure the usefulness of the automatic translations for human post-editing. One way of measuring the usefulness of a MT system is to see whether a post-editor can enhance the amount of correct translations without seeing the source text being translated. Therefore, for each source sentence, we presented a subject[6] with translations (produced by the above variants) from which he produced a final translation. He could do that by selecting one translation among the generated translations and enhancing its quality though slight modifications .

Out of the 500 target sentences that were produced in this way, 423 (84.6%) were just selections of one of the automatic translations. Out of these 423 translations, 85 (20%) were produced by the word-based engine (*ibm2+3g*).

In many cases it was impossible to guess a meaning from the translations. Particularly and in most cases for longer sentences, it was hopeless to amend the translation (*e.g. very sorry to have in case two people eight with two suitcases sit in the difference from here*). These translations were just copied. In other cases, almost any choice of the produced translations was a priori equally good, as in:

> how many minutes on foot ?
> how many minutes ?
> how many minutes does it take to get to the station by taxi ?

[6]One of the authors of this paper, not familiar with Chinese.

In total, 77 sentences were manually modified. Some obvious errors, such as repetitions of words, missing or additional articles and incomplete phrases, were corrected and the word position was adjusted. A one-sentence session is illustrated in Table 8; the full session can be seen at `www.iro.umontreal.ca/~felipe/iwslt/manual`.

Table 8: Illustration of the *manual* experiment. The user was presented here with 7 different translations, and produced his own one out of them.

trans1	take a bath for a twin room .
trans2	please take a bath for a double .
trans3	take a bath of double .
trans4	take one twin room with bath .
trans5	have a bath for double .
trans6	have a twin room with bath , please .
trans7	have a double room with bath , please .
manual	**please, a twin room with bath .**

This submission is called *manual* in Table 9 which shows the scores that were returned to us by the organizers.

As can be observed in Table 9, the order of merit of the variants we tried as measured on the CSTAR corpus is close to the one we observe on the TEST corpus. The exception is for the *QA* variant which performed worst on the latter corpus than the *merge* variant. We also observed a gain in performance for the *manual* version.

Table 9: Quality of the translations submitted before the deadline for the TEST corpus. The *QA* variant is the one we submitted for manual evaluation. The figures reported are rounded versions of the ones reported by the organizers.

config	BLEU%	NIST	GTM	WER	PER
ibm2+3g	27.27	6.55	62.49	58.12	48.82
straight	30.92	7.52	66.93	56.05	47.90
merge	35.32	8.00	68.60	51.74	43.86
QA	33.89	7.85	68.55	53.24	45.14
manual	36.93	8.13	68.42	49.62	42.53

8. Conclusions and future work

We took the opportunity of the IWSLT 2004 shared task to experiment with phrase-based models. Although FPBMs are conceptually very simple, we found that many factors must be considered to get the best out of them, and that a great amount of time must be spent to monitor the improvements.

Due to lack of time, we studied in this exercise only few of the factors that can affect the performance. We did not for example study the impact of word alignment techniques on our phrase acquisition method. Neither did we test carefully the different variants of the phrase extractors we imple-

mented. Finally, we did not find time to analyse why a given variant was working better than another very close one.

However, we experienced the importance of adequately tuning the meta-parameters of the decoder. We also observed that improvements could be obtained by merging the parameters of phrase-based and word-based models.

This work was manageable in a short period of time, thanks to the availability of the `Pharaoh` decoder. A by-product that we found useful is that this decoder offers a reference performance against which we can compare another decoder. In particular, we verified that our word-based engine had reasonable performances compared to `Pharaoh` seeded with the same transfer model.

The greatest frustration we had after accomplishing this work was to contemplate the numerous experiments we could have done but did not. One bottleneck into a systematic exploration of phrase-based variants is the tuning required after each change in any step of the acquisition of a FPBM. We plan to consider a better way of tuning the parameters toward a given metric or set of metrics.

9. References

[1] Zhao B., Vogel S. and Waibel A., "Phrase Pair Rescoring with Term Weightings for Statistical Machine Translation", In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Barcelona, Spain, July 2004

[2] Och F.J. and Ney H., "Improved Statistical Alignment Models", in Proceedings of the Conference of the Association for Computational Linguistic (ACL), Hongkong, China, pp. 440–447, 2000

[3] Koehn P., "Pharaoh: a Beam Search Decoder for Phrase-Based SMT", To appear in Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA), 2004

[4] Stolcke A., "SRILM - An Extensible Language Modeling Toolkit", In Proceedings of the International Conference for Speech and Language Processing (ICSLP), Denver, Colorado, September 2002

[5] Koehn P., Och F.J. and Marcu D., "Statistical Phrase-Based Translation", In Proceedings of the Human Language Technology Conference (HLT), pp. 127–133, 2003

[6] Tillmann C., "A Projection Extension Algorithm for Statistical Machine Translation", In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2003

[7] Och. F.J., Tillmann C. and Ney H., "Improved Alignment Models for Statistical Machine Translation", in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 20–28, College Park, Maryland, USA, 1999

[8] Zens R. and Ney H., "Improvements in Phrase-Based Statistical Machine Translation", In Proceedings of the Human Language Technology Conference (HLT-NAACL), Boston, MA, pp. 257–264, May 2004

[9] J. Tomàs and F. Casacuberta., "Combining phrase-based and template-based aligned models in statistical translation", In Proceedings of the First Iberian Conference on Pattern Recognition and Image Analysis, pp. 1020–1031, Mallorca, Spain, June 2003

[10] Vogel S., Zhang Y., Huang F., Tribble A., Venugopal A., Zhao B., Waibel A., "The CMU Statistical Machine Translation System", in Proceedings of MT Summit, pp.110–117, New Orleans, Louisiana, September, 2003

[11] Zhao B. and Vogel S., "Word Alignment Based on Bilingual Bracketing", In Proceedings of HLT-NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond, pp. 15–18, Edmonton, Alberta, Canada, May, 2003

[12] Simard M. and Langlais P., "Statistical Translation alignment with Compositionnality Constraints", HLT-NAACL Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond, Edmonton, Canada, May 31, pp.19–22, 2003

[13] Zhang Y., Vogel S. and Waibel A., "Integrated Phrase Segmentation and alignment Algorithm for Statistical Machine Translation", In Proceedings of the International Conference on Natural Language Processing and Knowledge Engineering (NLP-KE), Beijing, China, 2003

[14] Langlais P., Foster G. and Lapalme G., "Unit Completion for a Computer-aided Translation Typing System", In Proceedings of Applied Natural Language Processing (ANLP), Seattle, Washington, pp. 135–141, May, 2000

[15] Koehn P., "A Beam Search Decoder for Phrase-Based Statistical Machine Translation Models", Technical Manual of the Pharaoh decoder, 2003.

[16] Brown P.F, Della Pietra S.A, Della Pietra V.J and Mercer R.L., "The Mathematics of Statistical Machine Translation: Parameter Estimation", in Computational Linguistics, vol. 19 (2), pp. 263–311, 1993

[17] Niessen S., Vogel S. Ney H. and Tillmann C., "A DP based Search Algorithm for Statistical Machine Translation", in Proceedings of the International Conference On Computational Linguistics (COLING), pp. 960–966, 1998