Text Prediction with Fuzzy Alignments

George Foster, Philippe Langlais, and Guy Lapalme

RALI, Université de Montréal
www-rali.iro.umontreal.ca

Abstract. Text prediction is a form of interactive machine translation that is well suited to skilled translators. In recent work it has been shown that simple statistical translation models can be applied within a user-modeling framework to improve translator productivity by over 10% in simulated results. For the sake of efficiency in making real-time predictions, these models ignore the alignment relation between source and target texts. In this paper we introduce a new model that captures fuzzy alignments in a very simple way, and show that it gives modest improvements in predictive performance without significantly increasing the time required to generate predictions.

1 Introduction

The idea of using text prediction as a tool for translators was first introduced by Church and Hovy as one of many possible applications for "crummy" machine translation technology [1]. Text prediction can be seen as a form of interactive MT that is well suited to skilled translators. Compared to the traditional form of IMT based on Kay's original work [2]—in which the user's role is to help disambiguate the source text—prediction is less obtrusive and more natural, allowing the translator to focus on and directly control the contents of the target text. Predictions can benefit a translator in several ways: by accelerating typing, by suggesting translations, and by serving as an implicit check against errors.

The first implementation of a predictive tool for translators was described in [3], in the form of a simple word-completion system based on statistical models. Various enhancements to this were carried out as part of the TransType project [4], including the addition of a realistic user interface, better models, and the capability of predicting multi-word lexical units. In the final TransType prototype for English to French translation, the translator is presented with a short pop-up menu of predictions after each character typed. These may be incorporated into the text with a special command or rejected by continuing to type normally.

Although TransType is capable of correctly anticipating over 70% of the characters in a freely-typed translation (within the domain of its training corpus), this does not mean that users can translate in 70% less time when using the tool. In fact, in a trial with skilled translators, the users' rate of text production *declined* by an average of 17% as a result of using TransType [5]. There are two main reasons for this. First, it takes time to read the system's proposals, so that in cases where they are wrong or too short, the net effect will be to slow

the translator down. Second, translators do not always act "rationally" when confronted with a proposal; that is, they do not always accept correct proposals and they occasionally accept incorrect ones.

In previous work [6], we described a new approach to text prediction intended to address these problems. The main idea is to make proposals that maximize the expected benefit to the user in each context, rather than systematically predicting a fixed amount of text after each character typed. The expected benefit is estimated from two components: a statistical translation model that gives the probability that a candidate prediction will be correct or incorrect, and a user model that determines the benefit to the translator in either case. Simulated results indicate that this approach has the potential to increase translator productivity by over 10%, a considerable improvement over the -17% observed in the TransType trials.

For the sake of efficiency in making real-time predictions, the statistical translation model used in [6] ignores the alignment relation between source and target texts. Although this has negligible effect on very short predictions (for instance completions for the current word), it is noticable in longer predictions, which occasionally repeat previous segments of target text or contain translations for words that have already been translated. In this paper, we introduce and evaluate a new translation model that adds a notion of fuzzy alignments to the maximum-entropy model of [6].

The structure of the paper is as follows. Section 2 outlines the basic approach to text prediction. The subsequent three sections describe the main elements of this approach: translation model, user model, and search procedure (the latter two are condensed versions of the descriptions given in [6]). The last two sections give results and conclude.

2 The Text Prediction Task

In the basic prediction task, the input to the predictor is a source sentence $\mathbf{s}$ and a prefix $\mathbf{h}$ of its translation (ie, the target text before the current cursor position); the output is a proposed extension $\mathbf{x}$ to $\mathbf{h}$. Figure 1 gives an example. Unlike the TransType prototype, which proposes a set of alternate single-word suggestions, each prediction here consists of only a single proposal, but one that may span an arbitrary number of words.

As described above, the goal of the predictor is to find the prediction $\hat{\mathbf{x}}$ that maximizes the expected benefit to the user:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmax}}\, B(\mathbf{x}, \mathbf{h}, \mathbf{s}), \tag{1}$$

where $B(\mathbf{x}, \mathbf{h}, \mathbf{s})$ measures typing time saved. This obviously depends on how much of $\mathbf{x}$ is correct, and how long it would take to edit it into the desired text. A major simplifying assumption we make is that the user edits only by erasing wrong characters from the end of a proposal. Given a TransType-style interface where acceptance places the cursor at the end of a proposal, this is

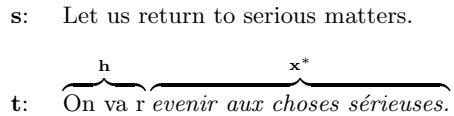

Fig. 1. Example of a prediction for English to French translation. $\mathbf{s}$ is the source sentence, $\mathbf{h}$ is the part of its translation that has already been typed, $\mathbf{x}^*$ is what the translator wants to type, and $\mathbf{x}$ is the prediction.

the most common editing method, and it gives a conservative estimate of the cost attainable by other methods. With this assumption, the key determinant of edit cost is the length of the correct prefix of $\mathbf{x}$, so the expected benefit can be written as:

$$B(\mathbf{x}, \mathbf{h}, \mathbf{s}) = \sum_{k=0}^{l} p(k|\mathbf{x}, \mathbf{h}, \mathbf{s})\; B(\mathbf{x}, \mathbf{h}, \mathbf{s}, k), \tag{2}$$

where $p(k|\mathbf{x}, \mathbf{h}, \mathbf{s})$ is the probability that exactly k characters from the beginning of $\mathbf{x}$ will be correct, l is the length of $\mathbf{x}$, and $B(\mathbf{x}, \mathbf{h}, \mathbf{s}, k)$ is the benefit to the user given that the first k characters of $\mathbf{x}$ are correct.

Equations (1) and (2) define three main problems: estimating the prefix probabilities $p(k|\mathbf{x}, \mathbf{h}, \mathbf{s})$, estimating the user benefit function $B(\mathbf{x}, \mathbf{h}, \mathbf{s}, k)$, and searching for $\hat{\mathbf{x}}$.

3 Translation Models

The correct-prefix probabilities $p(k|\mathbf{x}, \mathbf{h}, \mathbf{s})$ are derived from a statistical translation model that gives the probability $p(w|\mathbf{h}, \mathbf{s})$ that a word w will follow a previous sequence of words $\mathbf{h}$ in the translation of $\mathbf{s}$. (Note that this model does not distinguish between words in $\mathbf{h}$ that have been sanctioned by the translator and those that are hypothesized as part of the search procedure to find the best prediction.) As described in [6], the derivation involves first converting prefix probabilities to explicit character string probabilities of the form $p(x_1^k|\mathbf{h}, \mathbf{s})$, then calculating these by summing over all compatible token sequences.

3.1 MEMD2B

The model for $p(w|\mathbf{h}, \mathbf{s})$ used in [6] is a maximum entropy/minimum divergence (MEMD) model of the form:

$$p(w|\mathbf{h}, \mathbf{s}) = \frac{q(w|\mathbf{h}) \exp\left(\sum_{s \in \mathbf{s}} \alpha_{sw} + \alpha_{D(s,w,i,j)}\right)}{Z(\mathbf{h}, \mathbf{s})},$$

where $q(w|\mathbf{h})$ is a trigram language model; α_{sw} is a weight that captures the strength of the association between the current target word w and a word s in the

source sentence $\mathbf{s}$; $\alpha_{D(s,w,i,\hat{j})}$ is a weight that depends on the distance between the position i of w, and the position $\hat{j}$ of the closest occurrence of s in $\mathbf{s}$; and $Z(\mathbf{h},\mathbf{s})$ is a normalizing factor (the sum over all w of the numerator). The parameters of the model are the families of weights α_{sw} (one for each selected bilingual word pair), and $\alpha_{D(s,w,i,\hat{j})}$ (one for each position/word-pair class), which are set so as to maximize the likelihood of a training corpus. More details are given in [7], where this model is labelled MEMD2B.

MEMD2B is analogous to a linear combination of a trigram and the IBM model 2 [8], as used in the TransType prototype:

$$p(w|\mathbf{h},\mathbf{s}) = \lambda q(w|\mathbf{h}) + (1-\lambda)\sum_{j=0}^{J} p(w|s_j)p(j|i,J)$$

where $\lambda \in [0,1]$ is a combining weight, J is the number of tokens in $\mathbf{s}$, $p(w|s_j)$ is a translation probability that plays a role similar to the MEMD word-pair weight α_{sw}, and $p(j|i,J)$ is a position probability that plays a role similar to the MEMD position weight $\alpha_{D(s,w,i,\hat{j})}$. The most significant differences between the MEMD and linear models are that MEMD model combines the contributions from its language and translation components by multiplying instead of adding them; and that the MEMD translation parameters are learned in the presence of the trigram language model. These characteristics make MEMD2B significantly more powerful than its linear counterpart, even when it is defined using an order of magnitude fewer paramers. It yields about 50% lower test-corpus perplexity [7], and about 69% higher keystroke savings on the text prediction task (see table 2 in [6]—this is using the "best" estimates for both models, when predictions are limited to 5 words or fewer).

Both the MEMD and linear models have an obvious weakness, in that their translation components depend only on the length of $\mathbf{h}$ (ie, the position of w), and not on the actual words it contains. From the standpoint of predictive efficiency, this is a good thing, since it means that the models support $O(mJV^3)$ Viterbi-style searches for the most likely sequence of m words that follows $\mathbf{h}$, where V is the size of the target-language vocabulary. However, this speed comes at the cost of accuracy, because only the trigram and the relatively weak contribution from the position parameters prevent the models from assigning high probabilities to words that are repeat or alternate translations of source words that already have translations in $\mathbf{h}$. To ensure that this does not happen, a model must capture the alignment relation between $\mathbf{h}$ and $\mathbf{s}$ in some way.

3.2 Noisy Channel

In statistical MT, the standard way to capture the alignment relation is a noisy-channel approach, where the natural distribution $p(\mathbf{t}|\mathbf{s})$ for a target text $\mathbf{t}$ given a source text $\mathbf{s}$ is modeled via Bayes law as proportional to $p(\mathbf{s}|\mathbf{t})p(\mathbf{t})$ for a fixed source text. This combines language and translation components in an optimum way, and allows even the simplest IBM translation models 1 and 2 to capture the

alignment relation. The drawback is that the decoding problem is NP-complete for noisy-channel models [9], and even the fastest dynamic-programming heuristics used in statistical MT [10, 11], are polynomial in J—for instance $O(mJ^4V^3)$ in [11].

For the prediction application, a noisy channel decomposition of the distribution $p(w|\mathbf{h}, \mathbf{s})$ is:

$$p(w|\mathbf{h}, \mathbf{s}) = p(w|\mathbf{h})p(\mathbf{s}|\mathbf{h}, w)/p(\mathbf{s}|\mathbf{h}), \tag{3}$$

where, unlike in SMT, the denominator $p(\mathbf{s}|\mathbf{h})$ must be retained in order to give true probabilities for the user benefit calculation. To ensure that the resulting distribution is normalized over all w, $p(\mathbf{s}|\mathbf{h})$ can be calculated by summing the numerator over all words in the vocabulary:

$$p(\mathbf{s}|\mathbf{h}) = \sum_w p(w|\mathbf{h})p(\mathbf{s}|\mathbf{h}, w).$$

To see how the noisy-channel approach enforces alignment constraints, consider an IBM1 model for $p(\mathbf{s}|\mathbf{h}, w)$:

$$p(\mathbf{s}|\mathbf{h}, w) = \prod_{j=1}^{J} \left[\sum_{i=0}^{I} p(s_j|h_i) + p(s_j|w) \right] /(I+2)$$

where I is the length of $\mathbf{h}$, and h_i is the ith word in $\mathbf{h}$. Dividing by the constant factor $\prod_{j=1}^{J} \sum_{i=0}^{I} p(s_j|h_i)/(I+2)$ gives:

$$p(\mathbf{s}|\mathbf{h}, w) \propto \prod_{j=1}^{J} \left[1 + \frac{p(s_j|w)}{\sum_{i=0}^{I} p(s_j|h_i)} \right].$$

So the score assigned to w is a product of source word scores, each of which can only be significantly greater than 1 if the probability that the corresponding source word s translates to w is significantly larger than the sum of the probabilities of all previous translations for s in $\mathbf{h}$. As a consequence, the probability assigned to the first good translation of any source word will be much higher than that assigned to subsequent translations of the same word.

Unfortunately, apart from the expensive search properties noted above, the noisy channel model described here does not seem optimal for word prediction with a user model. The reason is that the distribution over w it gives is very flat, yielding probabilities for the best next words that are lower than their true probabilities (which are crucial for this application). This is reflected in the test corpus perplexity of the model, which is only slightly lower than that of the trigram language model on its own (even when IBM2 is used as the translation component). Tuning the model with various parameters such as an exponential weight on the translation component, or weights on the initial source-word scores $p(s_j|h_0)$, does not substantially improve this picture.

3.3 Fuzzy Alignments

Another approach to enforcing alignment constraints is to build them into the existing MEMD2B model. One way to do this would be to add features to capture translation relations between **h** and **s**. We have explored a simpler alternate approach based on modulating the weights of existing features that are active on the words in **h**—essentially "ticking-off" source words that appear to have valid translations in **h**. Our starting point is the observation that the word pairs captured by MEMD2B can be divided into two categories according to the magnitude of their weights, as shown in table 1: pairs with large weights tend to be true translations, while those with small weights tend to be looser semantic and grammatical associations.[1]

We experimented with a number of simple ways of exploiting this distinction, and found that the algorithm shown in figure 2 worked best. This uses a threshold f_1 to classify word-pairs as either translations or associations. Associations that occur between word pairs in **h** and **s** are modulated by a parameter f_2, while translation weights are set to the parameter f_3. For each pair deemed a translation, all weights involving the source word are modulated by f_4. Values for all four f_i parameters were optimized on a cross-validation corpus, using perplexity as a criterion.

This algorithm is applied sequentially to the target words in **h**, and its effects are incremental. Although the results will in general depend on the order in which target words are processed, no attempt is made to find the optimal order—processing is always left-to-right. We expect this dependence to be weak in any case, as the intention is to capture alignments in a fuzzy way, accounting for all possibilities at once and avoiding an expensive search for the optimal alignment.

this	ne	0.000164881
home	circonscription	0.000180333
very	le	0.000299113
,	aussi	0.000300543
at	dans	0.000360243
c-283	c-283	11.9133
mid-november	mi-novembre	11.9148
732	732	11.9304
darryl	darryl	11.9383
c-304	c-304	11.9559

Table 1. Five smallest positive (top box) and five largest (bottom box) word-pair weights for the MEMD2B model.

[1] Many of these appear to be spurious, but they capture statistically valid relationships within the domain—if they are eliminated from the model, its performance on new text within the domain drops.

```
for each source word s ∈ s:
  if α_sw < f_1:
    set α_sw ← α_sw f_2
  else:
    set α_sw ← f_3
    for all target words w' ≠ w
      set α_sw' ← α_sw' f_4
```

Fig. 2. Algorithm for modulating MEMB2B word-pair weights to account for the presence of some target word w in $\mathbf{h}$.

4 User Model

The purpose of the user model is to determine the expected benefit $B(\mathbf{x}, \mathbf{h}, \mathbf{s}, k)$ to the translator of a prediction $\mathbf{x}$ whose first k characters match the text that the translator wishes to type. This will depend heavily on whether the translator decides to accept or reject the prediction, so the first step in our model is the following expansion:

$$B(\mathbf{x}, \mathbf{h}, \mathbf{s}, k) = \sum_{a \in \{0,1\}} p(a|\mathbf{x}, \mathbf{h}, \mathbf{s}, k)\ B(\mathbf{x}, \mathbf{h}, \mathbf{s}, k, a),$$

where $p(a|\mathbf{x}, \mathbf{h}, \mathbf{s}, k)$ is the probability that the translator accepts or rejects $\mathbf{x}$, $B(\mathbf{x}, \mathbf{h}, \mathbf{s}, k, a)$ is the benefit they derive from doing so, and a is a random variable that takes on the values 1 for acceptance and 0 for rejection. The first two quantities are the main elements in the user model, and are described in following sections. The parameters of both were estimated from data collected during the TransType trial described in [5], which involved nine accomplished translators using a prototype prediction tool for approximately half an hour each. In all cases, estimates were made by pooling the data for all nine translators.

4.1 Acceptance Probability

The model for $p(a|\mathbf{x}, \mathbf{h}, \mathbf{s}, k)$ is based on the assumption that the probability of accepting $\mathbf{x}$ depends on roughly what the user stands to gain from it, defined according to the editing scenario given in section 2 as the amount by which the length of the correct prefix of $\mathbf{x}$ exceeds the length of the incorrect suffix:

$$p(a|\mathbf{x}, \mathbf{h}, \mathbf{s}, k) \approx p(a|2k - l),$$

where $k - (l - k) = 2k - l$ is called the *gain*. For instance, the gain for the prediction in figure 1 would be $2 \times 7 - 8 = 6$. It is straightforward to make empirical estimates of acceptance probabilities for each gain value; the model is simply a smoothed curve fit to these points.

4.2 Benefit

The benefit $B(\mathbf{x}, \mathbf{h}, \mathbf{s}, k, a)$ is defined as the typing time the translator saves by accepting or rejecting a prediction $\mathbf{x}$ whose first k characters are correct. To estimate this, we assume that the translator first reads $\mathbf{x}$, then, if he or she decides to accept, uses a special command to place the cursor at the end of $\mathbf{x}$ and erases its last $l - k$ characters. Assuming independence from $\mathbf{h}, \mathbf{s}$ as before, our model is:

$$B(\mathbf{x}, k, a) = \begin{cases} -R_1(\mathbf{x}) + T(\mathbf{x}, k) - E(\mathbf{x}, k), & a = 1 \\ -R_0(\mathbf{x}), & a = 0 \end{cases}$$

where $R_a(\mathbf{x})$ is the cost of reading $\mathbf{x}$ when it ultimately gets accepted ($a = 1$) or rejected ($a = 0$), $T(\mathbf{x}, k)$ is the cost of manually typing x_1^k, and $E(\mathbf{x}, k)$ is the edit cost of accepting $\mathbf{x}$ and erasing to the end of its first k characters. All of these elements are converted to units of keystrokes saved: $T(\mathbf{x}, k)$ and $E(\mathbf{x}, k)$ are estimated as k and $l - k + 1$ respectively; and read costs are converted from average elapsed times from proposal display to the next user action.

5 Search

Searching directly through all character strings $\mathbf{x}$ in order to find $\hat{\mathbf{x}}$ according to equation (1) would be very expensive. The fact that $B(\mathbf{x}, \mathbf{h}, \mathbf{s})$ is non-monotonic in the length of $\mathbf{x}$ makes it difficult to organize efficient dynamic-programming search techniques or use heuristics to prune partial hypotheses. Because of this, we adopted a fairly radical search strategy that involves first finding the most likely sequence of words of each length, then calculating the benefit of each of these sequences to determine the best proposal. The algorithm is:

1. For each length $m = 1 \ldots M$, find the best word sequence:
$$\hat{\mathbf{w}}_m = \underset{w_1^m}{\operatorname{argmax}}\, p(w_1^m | \mathbf{h}, \mathbf{s}),$$
2. Convert each $\hat{\mathbf{w}}_m$ to a corresponding character string $\hat{\mathbf{x}}_m$.
3. Output $\hat{\mathbf{x}} = \operatorname{argmax}_m B(\hat{\mathbf{x}}_m, \mathbf{h}, \mathbf{s})$, or the empty string if all $B(\hat{\mathbf{x}}_m, \mathbf{h}, \mathbf{s})$ are non-positive.

Step 1 is carried out using a Viterbi beam search with the translation model $p(w|\mathbf{h}, \mathbf{s})$. To speed this up, the search is limited to an *active vocabulary* of target words likely to appear in translations of $\mathbf{s}$, defined as the set of all words connected by some word-pair feature in our translation model to some word in $\mathbf{s}$. Step 2 is a trivial deterministic procedure that mainly involves deciding whether or not to introduce blanks between adjacent words (eg yes in the case of *la + vie*, no in the case of *l' + an*). Step 3 involves a straightforward evaluation of m strings according to equation (2).

Table 2 shows empirical search timings for various values of M, for both the baseline MEMD2B model and the alignment version. Although the average times for the alignment model are higher, they are still well below values that would cause delays perceptible to a user.

M	MEMD2B		MEMD2B-align	
	average time	maximum time	average time	maximum time
1	0.0012	0.01	0.0014	0.02
2	0.0038	0.23	0.0043	0.25
3	0.0097	0.51	0.0109	0.65
4	0.0184	0.55	0.0209	0.72
5	0.0285	0.57	0.0323	0.73

Table 2. Approximate times in seconds to generate predictions of maximum word sequence length M, on a 1.2GHz processor.

	MEMD2B					MEMD2B-align				
	M					M				
config	1	2	3	4	5	1	2	3	4	5
fixed	-8.5	-0.4	-3.6	-11.6	-20.8					
standard	5.8	10.7	12.0	12.5	12.6	5.8	10.9	12.7	13.2	13.4
best	7.9	17.9	24.5	27.7	29.2					

Table 3. Prediction results. Numbers give estimated percent reductions in keystrokes. Columns give the maximum permitted number of words M in predictions. Rows correspond to different predictor configurations: *fixed* ignores the user model and systematically makes M-word predictions; *standard* optimizes according to the user model, with model probabilities modified by the length-specific correction factors described in [6] (tuned separately for each model); and *best* gives an upper bound obtained by choosing m in step 3 of the search algorithm so as to maximize $B(\hat{\mathbf{x}}_m, \mathbf{h}, \mathbf{s}) = B(\hat{\mathbf{x}}_m, \mathbf{h}, \mathbf{s}, k_m)$, where k_m is the true value of k for $\hat{\mathbf{x}}_m$, from the test corpus.

6 Evaluation

To test the effect of adding alignment parameters to MEMD2B, we evaluated the English to French prediction performance of both the baseline model and the alignment version using the simulation technique described in [6]. The test corpus consisted of 5,020 Hansard sentence pairs and approximately 100k words in each language; details of the training corpus are given in [12]. The results are shown in table 3. The difference between the two models is negligible for predictions of less than three words, but increasingly significant for longer predictions, reaching a maximum relative improvement for the alignment model of about 6% with a prediction length limit of 5. This is in line with our intuition that the effect of including alignments should be more pronounced for longer predictions.

7 Conclusion

We have described a new maximum-entropy translation model for text prediction that improves on a previous model by incorporating a fuzzy notion of the alignment relation between the source text **s** and some initial part **h** of its translation. The improved model works at essentially the same speed as the previous one, and gives an increase of about 6% in estimated translator effort saved when predictions are limited to at most five words. This is a modest improvement, but

on the other hand it is achieved by adding only four parameters to the baseline maximum-entropy model. We feel that it demonstrates the potential of fuzzy alignments for this application, and we plan to investigate more sophisticated approaches in the future, possibly involving the addition of dedicated alignment features to the model.

References

1. Church, K.W., Hovy, E.H.: Good applications for crummy machine translation. Machine Translation **8** (1993) 239–258
2. Kay, M.: The MIND system. In Rustin, R., ed.: Natural Language Processing. Algorithmics Press, New York (1973) 155–188
3. Foster, G., Isabelle, P., Plamondon, P.: Target-text Mediated Interactive Machine Translation. Machine Translation **12** (1997) 175–194
4. Langlais, P., Foster, G., Lapalme, G.: Unit completion for a computer-aided translation typing system. Machine Translation **15** (2000) 267–294
5. Langlais, P., Lapalme, G., Loranger, M.: TransType: From an idea to a system. Machine Translation (2002) To Appear.
6. Foster, G., Langlais, P., Lapalme, G.: User-friendly text prediction for translators. In: Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP), Philadelphia, PA (2002)
7. Foster, G.: Incorporating position information into a Maximum Entropy / Minimum Divergence translation model. In: Proceedings of the 4th Computational Natural Language Learning Workshop (CoNLL), Lisbon, Portugal, ACL SigNLL (2000)
8. Brown, P.F., Pietra, S.A.D., Pietra, V.D.J., Mercer, R.L.: The mathematics of Machine Translation: Parameter estimation. Computational Linguistics **19** (1993) 263–312
9. Knight, K.: Decoding complexity in word-replacement translation models. Computational Linguistics, Squibs and Discussion **25** (1999)
10. Niessen, S., Vogel, S., Ney, H., Tillmann, C.: A DP based search algorithm for statistical machine translation. In: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (ACL) and 17th International Conference on Computational Linguistics (COLING) 1998, Montréal, Canada (1998) 960–967
11. Tillmann, C., Ney, H.: Word re-ordering and DP-based search in statistical machine translation. In: Proceedings of the International Conference on Computational Linguistics (COLING) 2000, Saarbrucken, Luxembourg, Nancy (2000)
12. Foster, G.: A Maximum Entropy / Minimum Divergence translation model. In: Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL), Hong Kong (2000)