Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Philadelphia, July 2002, pp. 148-155. Association for Computational Linguistics.

User-Friendly Text Prediction for Translators

George Foster and **Philippe Langlais** and **Guy Lapalme**
RALI, Université de Montréal
{foster,felipe,lapalme}@iro.umontreal.ca

Abstract

Text prediction is a form of interactive machine translation that is well suited to skilled translators. In principle it can assist in the production of a target text with minimal disruption to a translator's normal routine. However, recent evaluations of a prototype prediction system showed that it significantly *decreased* the productivity of most translators who used it. In this paper, we analyze the reasons for this and propose a solution which consists in seeking predictions that maximize the expected benefit to the translator, rather than just trying to anticipate some amount of upcoming text. Using a model of a "typical translator" constructed from data collected in the evaluations of the prediction prototype, we show that this approach has the potential to turn text prediction into a help rather than a hindrance to a translator.

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 (Church and Hovy, 1993). 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 (Kay, 1973)—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 (Foster et al., 1997), 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 (Langlais et al., 2000), 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 popup 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 (Langlais et al., 2002). 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. Many of the former cases correspond to translators simply ignoring proposals altogether, which is understandable behaviour given the first point.

This paper describes a new approach to text prediction intended to address these problems. The main idea is to make predictions that maximize the expected benefit to the user in each context, rather than systematically proposing 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. The user model takes into account the cost of reading a proposal, as well as the random nature of the decision to accept it or not. This approach can be characterized as making fewer but better predictions: in general, predictions will be longer in contexts where the translation model is confident, shorter where it is less so, and absent in contexts where it is very uncertain.

Other novel aspects of the work we describe here are the use of a more accurate statistical translation model than has previously been employed for text prediction, and the use of a decoder to generate predictions of arbitrary length, rather than just single words or lexicalized units as in the TransType prototype. The translation model is based on the maximum entropy principle and is designed specifically for this application.

To evaluate our approach to prediction, we simulated the actions of a translator over a large corpus of previously-translated text. The result is an increase of over 10% in translator productivity when using the predictive tool. This is a considerable improvement over the -17% observed in the TransType trials.

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 single-word (or single-unit) suggestions, we assume that each prediction 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 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 $\hat{\mathbf{x}}$. The following three sections describe our solutions to these.

3 Translation Model

The correct-prefix probabilities $p(k|\mathbf{x}, \mathbf{h}, \mathbf{s})$ are derived from a word-based statistical translation

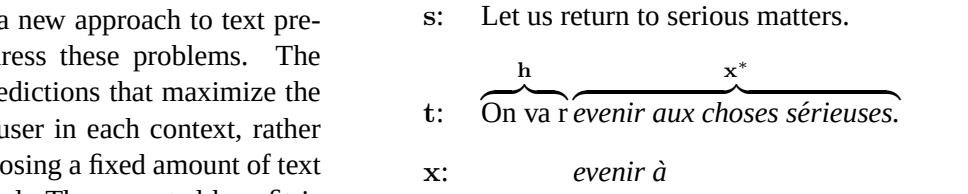

Figure 1: Example of a prediction for English to French translation. $\mathbf{s}$ is the source sentence, $\mathbf{h}$ is the part of its translation that has already been typed, $\mathbf{x}^*$ is what the translator wants to type, and $\mathbf{x}$ is the prediction.

model. The first step in the derivation is to convert these into a form that deals explicitly with character strings. This is accomplished by noting that $p(k|\mathbf{x}, \mathbf{h}, \mathbf{s})$ is the probability that the first k characters of $\mathbf{x}$ are correct *and* that the $k+1$th character (if there is one) is incorrect. For $k < l$:

$$p(k|\mathbf{x}, \mathbf{h}, \mathbf{s}) = p(x_1^k|\mathbf{h}, \mathbf{s}) - p(x_1^{k+1}|\mathbf{h}, \mathbf{s})$$

where $x_1^k = x_1 \ldots x_k$. If $k = l$, $p(k|\mathbf{x}, \mathbf{h}, \mathbf{s}) = p(\mathbf{x}|\mathbf{h}, \mathbf{s})$. Also, $p(x_1^0) \equiv 1$.

The next step is to convert string probabilities into word probabilities. To do this, we assume that strings map one-to-one into token sequences, so that:

$$p(x_1^k|\mathbf{h}, \mathbf{s}) \approx p(v_1, w_2, \ldots, w_{m-1}, u_m|\mathbf{h}, \mathbf{s}),$$

where v_1 is a possibly-empty word suffix, each w_i is a complete word, and u_m is a possibly empty word prefix. For example, if $\mathbf{x}$ in figure 1 were *evenir aux choses*, then x_1^{14} would map to $v_1 =$ *evenir*, $w_2 =$ *aux*, and $u_3 =$ *cho*. The one-to-one assumption is reasonable given that entries in our lexicon contain neither whitespace nor internal punctuation.

To model word-sequence probabilities, we apply the chain rule:

$$\begin{aligned} p(v_1, w_2, \ldots, w_{m-1}, u_m|\mathbf{h}, \mathbf{s}) = \\ p(v_1|\mathbf{h}, \mathbf{s}) \prod_{i=2}^{m-1} p(w_i|\mathbf{h}, v_1, w_2^{i-1}, \mathbf{s}) \times \\ p(u_m|\mathbf{h}, v_1, w_2^{m-1}, \mathbf{s}). \end{aligned} \quad (3)$$

The probabilities of v_1 and u_m can be expressed in terms of word probabilities as follows. Letting u_1 be the prefix of the word that ends in v_1 (eg, *r* in figure 1), $w_1 = u_1 v_1$, and $\mathbf{h} = \mathbf{h}' u_1$:

$$p(v_1|\mathbf{h}, \mathbf{s}) = p(w_1|\mathbf{h}', \mathbf{s}) / \sum_{w: w = u_1 v} p(w|\mathbf{h}', \mathbf{s}),$$

where the sum is over all words that start with u_1. Similarly:

$$p(u_m|\mathbf{h}', w_1^{m-1}, \mathbf{s}) = \sum_{w: w = u_m v} p(w|\mathbf{h}', w_1^{m-1}, \mathbf{s}). \quad (4)$$

Thus all factors in (3) can be calculated from probabilities of the form $p(w|\mathbf{h}, \mathbf{s})$ which give the likelihood that a word w will follow a previous sequence of words $\mathbf{h}$ in the translation of $\mathbf{s}$.[1] This is the family of distributions we have concentrated on modeling.

Our model for $p(w|\mathbf{h}, \mathbf{s})$ is a log-linear combination of a trigram language model for $p(w|\mathbf{h})$ and a maximum-entropy translation model for $p(w|\mathbf{s})$, described in (Foster, 2000a; Foster, 2000b). The translation component is an analog of the IBM model 2 (Brown et al., 1993), with parameters that are optimized for use with the trigram. The combined model is shown in (Foster, 2000a) to have significantly lower test corpus perplexity than the linear combination of a trigram and IBM 2 used in the TransType experiments (Langlais et al., 2002). Both models support $O(mJV^3)$ Viterbi-style searches for the most likely sequence of m words that follows $\mathbf{h}$, where J is the number of tokens in $\mathbf{s}$ and V is the size of the target-language vocabulary.

Compared to an equivalent noisy-channel combination of the form $p(\mathbf{t})p(\mathbf{s}|\mathbf{t})$, where $\mathbf{t}$ is the target sentence, our model is faster but less accurate. It is faster because the search problem for noisy-channel models is NP-complete (Knight, 1999), and even the fastest dynamic-programming heuristics used in statistical MT (Niessen et al., 1998; Tillmann and Ney, 2000), are polynomial in J—for instance $O(mJ^4V^3)$ in (Tillmann and Ney, 2000). It is less accurate because it ignores the alignment relation between $\mathbf{s}$ and $\mathbf{h}$, which is captured by even the simplest noisy-channel models. Our model is therefore suitable for making predictions in real time, but not for establishing complete translations unassisted by a human.

3.1 Implementation

The most expensive part of the calculation in equation (3) is the sum in (4) over all words in the vocabulary, which according to (2) must be carried out for every character position k in a given prediction $\mathbf{x}$. We reduce the cost of this by performing sums only at the end of each sequence of complete tokens in $\mathbf{x}$ (eg, after *revenir* and *revenir aux* in the above example). At these points, probabilities for all possible prefixes of the next word are calculated in a

[1] Here we ignore the distinction between previous words that have been sanctioned by the translator and those that are hypothesized as part of the current prediction.

single recursive pass over the vocabulary and stored in a trie for later access.

In addition to the exact calculation, we also experimented with establishing exact probabilities via $p(w|\mathbf{h}, \mathbf{s})$ only at the end of each token in $\mathbf{x}$, and assuming that the probabilities of the intervening characters vary linearly between these points. As a result of this assumption, $p(k|\mathbf{x}, \mathbf{h}, \mathbf{s}) = p(x_1^k|\mathbf{h}, \mathbf{s}) - p(x_1^{k+1}|\mathbf{h}, \mathbf{s})$ is constant for all k between the end of one word and the next, and therefore can be factored out of the sum in equation (2) between these points.

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 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 (Langlais et al., 2002), 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

Ideally, a model for $p(a|\mathbf{x}, \mathbf{h}, \mathbf{s}, k)$ would take into account whether the user actually reads the proposal before accepting or rejecting it, eg:

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

where r is a boolean "read" variable. However, this information is hard to extract reliably from the available data; and even if were obtainable, many of the factors which influence whether a user is likely to read a proposal—such as a record of how many previous predictions have been accepted—are not available to the predictor in our formulation. We thus model $p(a|\mathbf{x}, \mathbf{h}, \mathbf{s}, k)$ directly.

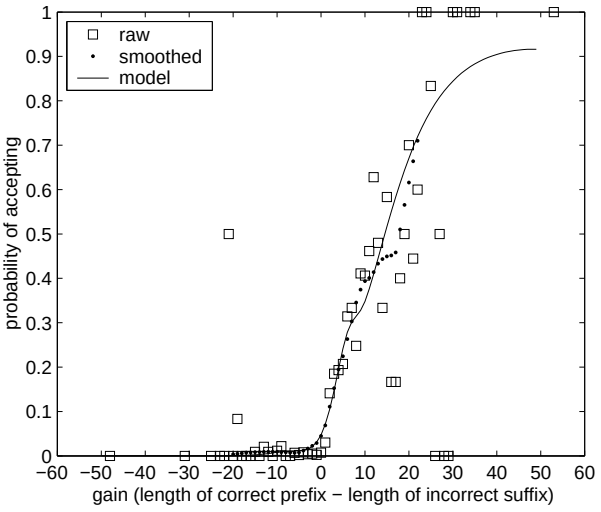

Figure 2: Probability that a prediction will be accepted versus its gain.

Our model is based on the assumption that the probability of accepting $\mathbf{x}$ depends only on 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 $2 \times 7 - 8 = 6$. The strongest part of this assumption is dropping the dependence on $\mathbf{h}$, because there is some evidence from the data that users are more likely to accept at the beginnings of words. However, this does not appear to have a severe effect on the quality of the model.

Figure 2 shows empirical estimates of $p(a = 1|2k - l)$ from the TransType data. There is a certain amount of noise intrinsic to the estimation procedure, since it is difficult to determine $\mathbf{x}^*$, and therefore k, reliably from the data in some cases (when the user is editing the text heavily). Nonetheless, it is apparent from the plot that gain is a useful abstrac-

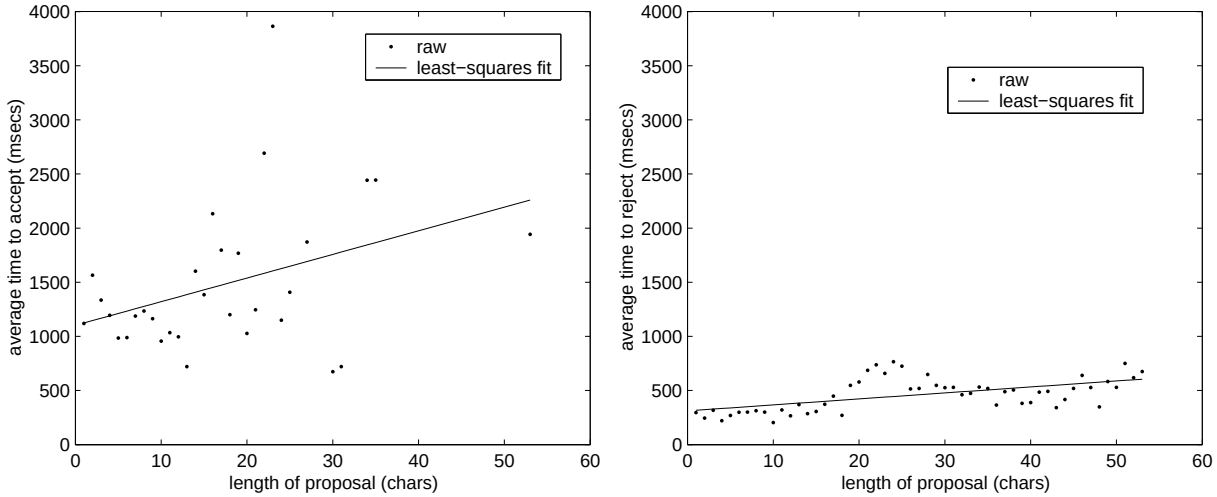

Figure 3: Time to read and accept or reject proposals versus their length

tion, because the empirical probability of acceptance is very low when it is less than zero and rises rapidly as it increases. This relatively clean separation supports the basic assumption in section 2 that benefit depends on k.

The points labelled *smoothed* in figure 2 were obtained using a sliding-average smoother, and the *model* curve was obtained using two-component Gaussian mixtures to fit the smoothed empirical likelihoods $p(\text{gain}|a=0)$ and $p(\text{gain}|a=1)$. The model probabilities are taken from the curve at integral values. As an example, the probability of accepting the prediction in figure 1 is about .25.

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 determine 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.

A natural unit for $B(\mathbf{x}, k, a)$ is the number of keystrokes saved, so all elements of the above equation are converted to this measure. This is straightforward in the case of $T(\mathbf{x}, k)$ and $E(\mathbf{x}, k)$, which are estimated as k and $l-k+1$ respectively—for $E(\mathbf{x}, k)$, this corresponds to one keystroke for the command to accept a prediction, and one to erase each wrong character. This is likely to slightly underestimate the true benefit, because it is usually harder to type n characters than to erase them.

As in the previous section, read costs are interpreted as expected values with respect to the probability that the user actually does read $\mathbf{x}$, eg, assuming 0 cost for not reading, $R_0(\mathbf{x}) = p(r=1|\mathbf{x})R'_0(\mathbf{x})$, where $R'_0(\mathbf{x})$ is the unknown true cost of reading and rejecting $\mathbf{x}$. To determine $R_a(\mathbf{x})$, we measured the average elapsed time in the TransType data from the point at which a proposal was displayed to the point at which the next user action occurred—either an acceptance or some other command signalling a rejection. Times greater than 5 seconds were treated as indicating that the translator was distracted and were filtered out. As shown in figure 3, read times are much higher for predictions that get accepted, reflecting both a more careful perusal by the translator and the fact the rejected predictions are often simply ignored.[2] In both cases there is a weak linear rela-

[2] Here the number of characters read was assumed to include the whole contents of the TransType menu in the case of rejections, and only the proposal that was ultimately accepted in the case of acceptances.

tionship between the number of characters read and the time taken to read them, so we used the least-squares lines shown as our models. Both plots are noisy and would benefit from a more sophisticated psycholinguistic analysis, but they are plausible and empirically-grounded first approximations.

To convert reading times to keystrokes for the benefit function we calculated an average time per keystroke (304 milliseconds) based on sections of the trial where translators were rapidly typing and when predictions were *not* displayed. This gives an upper bound for the per-keystroke cost of reading—compare to, for instance, simply dividing the total time required to produce a text by the number of characters in it—and therefore results in a conservative estimate of benefit.

To illustrate the complete user model, in the figure 1 example the benefit of accepting would be $7-2-4.2 = .8$ keystrokes and the benefit of rejecting would be $-.2$ keystrokes. Combining these with the acceptance probability of .25 gives an overall expected benefit $B(\mathbf{x}, \mathbf{h}, \mathbf{s}, k = 7)$ for this proposal of 0.05 keystrokes.

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:

$$\hat{\mathbf{w}}_m = \underset{w_1:(w_1=u_1 v),\, w_2^m}{\operatorname{argmax}} p(w_1^m | \mathbf{h}', \mathbf{s}),$$

   where u_1 and $\mathbf{h}'$ are as defined in section 3.

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.

M	average time	maximum time
1	0.0012	0.01
2	0.0038	0.23
3	0.0097	0.51
4	0.0184	0.55
5	0.0285	0.57

Table 1: Approximate times in seconds to generate predictions of maximum word sequence length M, on a 1.2GHz processor, for the MEMD model.

In all experiments reported below, M was set to a maximum of 5 to allow for convenient testing. Step 1 is carried out using a Viterbi beam search. 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*). This also removes the prefix u_1 from the proposal. Step 3 involves a straightforward evaluation of m strings according to equation (2).

Table 1 shows empirical search timings for various values of M, for the MEMD model described in the next section. Times for the linear model are similar. Although the maximum times shown would cause perceptible delays for $M > 1$, these occur very rarely, and in practice typing is usually not noticeably impeded when using the TransType interface, even at $M = 5$.

6 Evaluation

We evaluated the predictor for English to French translation on a section of the Canadian Hansard corpus, after training the model on a chronologically earlier section. The test corpus consisted of 5,020 sentence pairs and approximately 100k words in each language; details of the training corpus are given in (Foster, 2000b).

To simulate a translator's responses to predictions, we relied on the user model, accepting probabilistically according to $p(a|\mathbf{x}, \mathbf{h}, \mathbf{s}, k)$, determining the associated benefit using $B(\mathbf{x}, \mathbf{h}, \mathbf{s}, k, a)$, and advancing the cursor k characters in the case of an

config	M				
	1	2	3	4	5
fixed	-8.5	-0.4	-3.60	-11.6	-20.8
linear	6.1	9.40	8.8	8.1	7.8
exact	5.3	10.10	**10.7**	10.0	9.7
corr	5.8	10.7	12.0	12.5	**12.6**
best	7.9	17.90	24.5	27.7	29.2
fixed	**-11.5**	-9.3	-15.1	-22.0	-28.2
exact	3.0	4.3	5.0	5.2	5.2
best	6.2	12.1	15.4	16.7	17.3

Table 2: Results for different predictor configurations. Numbers give % reductions in keystrokes.

user	M				
	1	2	3	4	5
superman	48.6	53.5	51.8	51.1	50.9
rational	11.7	17.8	17.2	16.4	16.1
real	5.3	10.10	10.7	10.0	9.7

Table 3: Results for different user simulations. Numbers give % reductions in keystrokes.

acceptance, 1 otherwise. Here k was obtained by comparing $\mathbf{x}$ to the known $\mathbf{x}^*$ from the test corpus. It may seem artificial to measure performance according to the objective function for the predictor, but this is biased only to the extent that it misrepresents an actual user's characteristics. There are two cases: either the user is a better candidate—types more slowly, reacts more quickly and rationally—than assumed by the model, or a worse one. The predictor will not be optimized in either case, but the simulation will only overestimate the benefit in the second case. By being conservative in estimating the parameters of the user model, we feel we have minimized the number of translators who would fall into this category, and thus can hope to obtain realistic lower bounds for the average benefit across all translators.

Table 2 contains results for two different translation models. The top portion corresponds to the MEMD2B maximum entropy model described in (Foster, 2000a); the bottom portion corresponds to the linear combination of a trigram and IBM 2 used in the TransType experiments (Langlais et al., 2002). Columns give the maximum permitted number of words in predictions. Rows show different predictor configurations: *fixed* ignores the user model and makes fixed M-word predictions; *linear* uses the linear character-probability estimates described in section 3.1; *exact* uses the exact character-probability calculation; *corr* is described below; and *best* gives an upper bound on performance by choosing m in step 3 of the search algorithm so as to maximize $B(\mathbf{x}, \mathbf{h}, \mathbf{s}, k)$ using the true value of k.

Table 3 illustrates the effects of different components of the user model by showing results for simulated users who read infinitely fast and accept only predictions having positive benefit (*superman*); who read normally but accept like superman (*rational*); and who match the standard user model (*real*). For each simulation, the predictor optimized benefits for the corresponding user model.

Several conclusions can be drawn from these results. First, it is clear that estimating expected benefit is a much better strategy than making fixed-word-length proposals, since the latter causes an increase in time for all values of M. In general, making "exact" estimates of string prefix probabilities works better than a linear approximation, but the difference is fairly small.

Second, the MEMD2B model significantly outperforms the trigram+IBM2 combination, producing better results for every predictor configuration tested. The figure of -11.5% in bold corresponds to the TransType configuration, and corroborates the validity of the simulation.[3]

Third, there are large drops in benefit due to reading times and probabilistic acceptance. The biggest cost is due to reading, which lowers the best possible keystroke reduction by almost 50% for $M = 5$. Probabilistic acceptance causes a further drop of about 15% for $M = 5$.

The main disappointment in these results is that performance peaks at $M = 3$ rather than continuing to improve as the predictor is allowed to consider longer word sequences. Since the predictor knows $B(\mathbf{x}, \mathbf{h}, \mathbf{s}, k)$, the most likely cause for this is that the estimates for $p(\hat{\mathbf{w}}_m|\mathbf{h}, \mathbf{s})$ become worse with increasing m. Significantly, performance lev-

[3] Although the drop observed with real users was greater at about 20% (= 17% reduction in speed), there are many differences between experimental setups that could account for the discrepancy. For instance, part of the corpus used for the TransType trials was drawn from a different domain, which would adversely affect predictor performance.

els off at three words, just as the search loses direct contact with $\mathbf{h}$ through the trigram. To correct for this, we used modified probabilities of the form $\lambda_m \; p(\hat{\mathbf{w}}_m|\mathbf{h},\mathbf{s})$, where λ_m is a length-specific correction factor, tuned so as to optimize benefit on a cross-validation corpus. The results are shown in the *corr* row of table 2, for exact character-probability estimates. In this case, performance improves with M, reaching a maximum keystroke reduction of 12.6% at $M = 5$.

7 Conclusion and Future Work

We have described an approach to text prediction for translators that is based on maximizing the benefit to the translator according to an explicit user model whose parameters were set from data collected in user evaluations of an existing text prediction prototype. Using this approach, we demonstrate in simulated results that our current predictor can reduce the time required for an average user to type a text in the domain of our training corpus by over 10%. We look forward to corroborating this result in tests with real translators.

There are many ways to build on the work described here. The statistical models which are the backbone of the predictor could be improved by making them adaptive—taking advantage of the user's input—and by adding features to capture the alignment relation between $\mathbf{h}$ and $\mathbf{s}$ in such a way as to preserve the efficient search properties. The user model could also be made adaptive, and it could be enriched in many other ways, for instance so as to capture the propensity of translators to accept at the beginnings of words.

We feel that the idea of creating explicit user models to guide the behaviour of interactive systems is likely to have applications in areas of NLP apart from translators' tools. For one thing, most of the approach described here carries over more or less directly to monolingual text prediction, which is an important tool for the handicapped (Carlberger et al., 1997). Other possibilities include virtually any application where a human and a machine communicate through a language-rich interface.

References

Peter F. Brown, Stephen A. Della Pietra, Vincent Della J. Pietra, and Robert L. Mercer. 1993. The mathematics of Machine Translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312, June.

Alice Carlberger, Johan Carlberger, Tina Magnuson, Sira E. Palazuelos-Cagigas, M. Sharon Hunnicutt, and Santiago Aguilera Navarro. 1997. Profet, a new generation of word prediction: an evaluation study. In *Proceedings of the 2nd Workshop on NLP for Communication Aids*, Madrid, Spain, July.

Kenneth W. Church and Eduard H. Hovy. 1993. Good applications for crummy machine translation. *Machine Translation*, 8:239–258.

George Foster, Pierre Isabelle, and Pierre Plamondon. 1997. Target-text Mediated Interactive Machine Translation. *Machine Translation*, 12:175–194.

George Foster. 2000a. Incorporating position information into a Maximum Entropy / Minimum Divergence translation model. In *Proceedings of the 4th Computational Natural Language Learning Workshop (CoNLL)*, Lisbon, Portugal, September. ACL SigNLL.

George Foster. 2000b. A Maximum Entropy / Minimum Divergence translation model. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL)*, Hong Kong, October.

Martin Kay. 1973. The MIND system. In R. Rustin, editor, *Natural Language Processing*, pages 155–188. Algorithmics Press, New York.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics, Squibs and Discussion*, 25(4).

Philippe Langlais, George Foster, and Guy Lapalme. 2000. Unit completion for a computer-aided translation typing system. *Machine Translation*, 15(4):267–294, December.

Philippe Langlais, Guy Lapalme, and Marie Loranger. 2002. TransType: From an idea to a system. *Machine Translation*. To Appear.

S. Niessen, S. Vogel, H. Ney, and C. Tillmann. 1998. A DP based search algorithm for statistical machine translation. In *Proceedings of the 36th Annual Meeting of the ACL and 17th COLING 1998*, pages 960–967, Montréal, Canada, August.

C. Tillmann and H. Ney. 2000. 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, August.