Improved Search Strategy for Interactive Predictions in Computer-Assisted Translation

Fatemeh Azadi ft.azadi@aut.ac.ir
Department of Computer Engineering,
Amirkabir University of Technology, Tehran, Iran
Shahram Khadivi[1] skhadivi@ebay.com
eBay Inc, Aachen, Germany

Abstract

The statistical machine translation outputs are not error-free and in a high quality yet. So in the cases that we need high quality translations we definitely need the human intervention. An interactive-predictive machine translation is a framework, which enables the collaboration of the human and the translation system. Here, we address the problem of searching the best suffix to propose to the user in the phrase-based interactive prediction scenario. By adding the jump operation to the common edit distance based search, we try to overcome the lack of some of the reorderings in the search graph which might be desired by the user. The experiments results shows that this method improves the base method by 1.35% in KSMR[2], and if we combine the edit error in the proposed method with the translation scores given by the statistical models to select the offered suffix, we could gain the KSMR improvement of about 1.63% compared to the base search method.

1. Introduction

Although the significant improvements achieved in the field of statistical machine translation, the current models and therefore the systems which can be built from them are still far from perfect. So, in order to achieve good or even acceptable translations, manual post editing is needed. An alternative to this approach is given by the interactive predictive machine translation paradigm (Barrachina et al., 2009). Under this paradigm, translation is considered as an iterative process where the human translator and the computer collaborates to generate the final translation, and in this way both the human and the computer can help each other.

In the interactive predictive MT scenario, the system initially offers a translation for a source sentence. From this translation, the user could mark a prefix as correct and begins to type the rest of the target sentence as it's desirable for him. By typing the single next character (or maybe the full next word, according to the system settings) the interactive MT system would suggest a new suffix that completes the user's confirmed prefix with the new character the user has just typed. This procedure continues iteratively until the user accepts the translation to be complete and correct.

One of the important issues in the development of interactive predictive MT systems is the strategy to search and offer the best suffix that completes the prefix confirmed by the user. One of the most common ways for phrase-based translation systems is to use the search graph that is obtained once during the translation process, and complete each user's confirmed pre-

[1] This work has been done when the author was with Amirkabir University of Technology.
[2] Keystroke and mouse-action ratio

fix by searching the best path which has a prefix with highest matching to the user's prefix, and offering the rest of it to the user. The most important advantage of using the pre-built search graph in each interaction is that it leads to an efficient response time for each interaction.

The problem in searching the best system offer arises when the prefix confirmed by the user is not producible by the statistical models of the translation system. That is it could not be produced by any of the paths in the search graph, so that the interactive predictive MT system could not offer any completion to the user. A common solution to this problem is that instead of searching a path in the graph whose prefix matches exactly with the user's prefix, we search more freely, for a path which has the most similarity with the user's prefix. Usually, the edit distance similarity measure is used for this purpose.

The search graph may not contain all of the possible reorderings for a certain set of words, due to the weakness of translation and reordering models or the pruning done during the generation of the search graph. So, the user's desired translation might have a reordering which doesn't exist in any of the search graph paths and the search based on edit distance could not perform well in these cases, since it aligns the user's prefix and a prefix of each path in the search graph monotonically and with no reordering.

In this paper we try to solve this problem for phrase-based interactive MT systems, by adding the jump operation in computing edit distance, which allows jumping over some parts in each path while aligning with the user's prefix, and then offering those parts as the suffix to the user; i.e. shifting some parts to a position after user's prefix, so that it changes the reordering of the system's translation.

Also, we compare and combine this method with another improvement done to the edit distance based search in (Vakil and Khadivi, 2012), which uses the weighted sum of the edit distance and the translation score for each path for selecting the best suffix.

In section 2 the related works done in the computer assisted translation field are reviewed. In section 3, we briefly explain the statistical framework for the interactive predictive MT. Then in section 4, the searching algorithms for finding the best suffix are discussed and the proposed method in this paper is presented. Finally, the experiments done for the evaluation of our method is presented in section 5.

2. Related Work

Interactivity in CAT[3] has been explored for a long time. Systems have been designed to interact with human translators in order to solve different types of (lexical, syntactic, or semantic) ambiguities (Slocum 1985; Whitelock et al. 1986).

A major change in the interactive CAT technology was done within the TransType project in (Foster et al. ,1997; Langlais et al., 2000). A new approach called Target-Text Mediation was proposed, in which the focus of the interaction was shifted from the source text to the production of the target text directly. This project with TransType2 (TT2) (Bender et al., 2005; Tomás and Casacuberta 2006; Barrachina et al., 2009) proposed to embed an MT system in an interactive scenario. This way, the human translator can ensure a high quality output while the MT system ensures the significant gain of productivity.

Particularly the interactive-predictive machine translation paradigm was proposed in (Barrachina et al., 2009). In this paradigm a statistical MT system uses the source sentence and a prefix of it's translation which is confirmed by the user to propose a suitable continuation. And the text would be translated in an iterative process of interactions between

[3] Computer Assisted Translation

the user and the interactive MT system. The results showed that interactive MT can save a significant amount of human effort. Also different translation models where tested in (Barrachina et al., 2009), and the results showed that the phrased based models have a better performance than the other ones.

After TT2 some sporadic works were done on developing CAT systems (Ortiz-Martinez et al., 2009; Huang et al., 2012), but the more famous one was the Caitra system (Koehn 2009; Koehn and Haddow, 2009). It was a web-based system, which could be accessed online by the users. The search method used as the base method in this paper, which will be discussed in section 4.1, is the method proposed and implemented in this system.

Recently two other CAT projects, CASMACAT[4] and MATECAT[5] were done, which were committed to develop an open source workbench targeted both at researchers to investigate novel and enhanced types of assistance and at professional translators for actual use (Federico et al., 2014; Alabau et al., 2014). One of the goals of these projects was to gain insight into the cognitive processes involved in human translation, by using eye tracking and key logging. The cognitive analysis will determine what types of assistance are offered to the translator, what information should be displayed on the screen, and what information should be hidden as it would be distracting. Also they try to add the capability of the automatic adaptation to the translated content to the CAT tools, which is not related to the subject of this paper. Also the interactive-predictive CAT scenario when the user pronounce the correct translation instead of typing it has been studied in (Vakil and Khadivi, 2012b).

The other work done to improve the searching methods for the interactive prediction is (Vakil and Khadivi, 2012a), which tries to consider both the edit distance and the translation score of each path in the search graph instead of just using the edit distance. This approach, which is called the weighted edit distance, is discussed more in section 4.2.

Also in (Ortiz-Martínez, 2011) and (González-Rubio et al., 2013) a search approach is proposed, which is somewhat alike the weighted edit distance approach. But they introduce a unified statistical framework, which integrates all of the statistical models used in the translation machine with the edit distance error, which is called error correction model here. Also the proposed approach, models the edit distance as a Bernoulli process where each character of the candidate string has a probability of being erroneous.

The jump operation that we used here is somehow like the shift operation used in Translation Error Rate (Snover et al., 2006), which is an evaluation metrics for machine translation and measures the number of edits required to change a system output into one of the references.

3. Interactive-Predictive Machine Translation

In the interactive predictive machine translation paradigm, we want to produce a suffix which is the best according to the user's prefix and the source sentence. So in the statistical approach, we should maximize the probability of the suffix given the source sentence and the confirmed prefix, as follows:

$$\hat{t}_s = \arg\max_{t_s} \Pr(t_s \mid s, t_p)$$

Where t_s is the suffix, t_p is the user's confirmed prefix, and s is the source sentence. We can write the above equation as follows:

[4] http://www.casmacat.eu/index.php
[5] http://www.matecat.com/

$$\hat{t}_s = \arg\max_{t_s} \Pr(t_p, t_s \mid s) = \arg\max_{t_s} \Pr(t_p, t_s).\Pr(s \mid t_p, t_s)$$

Noting that $t_p t_s = t$, this equation would be very similar to the following, which is the statistical machine translation's equation.

$$\hat{t} = \arg\max_{t} \Pr(t).\Pr(s \mid t)$$

The main difference between these two is that in the interactive MT the search process is restricted to those target sentences that contains t_p as prefix. This implies that we can use the same MT models if the search procedures are adequately modified (Och et al., 2003).

One of the most common ways, that we used here, for searching the best suffix in the interactive MT is to use the search graph that is obtained once during the translation process, and complete each user's confirmed prefix by searching the best path which has a prefix which matches to the user's prefix, and offering the rest of it to the user. The most important advantage of using the pre-built search graph in each interaction is that it leads to an efficient response time for each interaction. However, there's not always a path in the search graph whose prefix matches exactly with the user's prefix, so some mechanisms should be chosen the search for a path which has the most similarity with the user's prefix. A common approach is to use the edit distance measure which will be discussed in the next section.

4. Searching Algorithms for the Interactive Prediction

In this section, we first introduce the searching strategy based on the edit distance, which is presented in (Koehn, 2009), and is used as our base search algorithm. Then we try to improve this algorithm in two ways; first by adding the edit distance error as a weighted model to the translation scores computed for each path by the decoder, and using this score for selecting the best path, as in (Vakil and Khadivi 2012). And in the second method we add a jump operation to the computation of the edit distance, to consider the reorderings which are not in the search graph but might match with the user's prefix.

4.1. Searching Based on Edit Distance

According to (Koehn, 2009), for giving a completed translation offer to the user, we should find a hypothesis (or node) in the graph whose generated partial translation has the minimum edit distance with the confirmed prefix; This approach is called edit distance-based search. The edit distance between two strings is defined as the minimum number of edits needed to transform one string into the other, while the allowable edit operations are insertion, deletion, or substitution (Levenshtein, 1966). Although these operations could be done at character or word level, the word level distance makes more sense for our purpose, as performing one or more character level operations on a word might change the whole meaning of that word, and the number of operations done is not matter.

In this method the prediction is the optimal completion path that matches the user's prefix with minimal edit distance, and if there were more than one such paths, the one with highest translation score will be chosen. This approach is based on the assumption that a hypothesis with minimum edit distance with the confirmed prefix has the continuation which is more likely to be consistent with the remained part of the user's desired translation.

This search method has shortcomings, as the search graph may not contain all of the reordering possibilities of words due to the pruning that is applied during the generation of the search graph. Since the edit distance only includes the insertion, deletion and substitution op-

erations, it could not be able to consider the difference in reorderings between the user prefix and any of the translation hypotheses in the search graph. Thus, it's only able to find those hypotheses which have a similar reordering of words to the user's prefix. In order to solve this problem we discuss two approaches in the next sections.

4.2. Searching Based on Weighted Edit Distance

This approach which is presented in (Vakil and Khadivi, 2012), is based on the fact that the paths in the search graph that have higher translation scores according to the translation and language models, are more probable to be better translations in terms of both semantics and syntax. However, these better translation hypotheses might not sometimes have lowest edit distance with the user's prefix due to the different reordering of words they have. Thus, using the edit distance alone for selecting the best path, regardless of its translation score, would select another path in the graph that might be an unfavourable translation.

To clarify this, consider the example shown in Table 1. We have the user's desired translation and two translation hypotheses samples, extracted from the search graph with their translation scores, for a specific sentence. Assume that the prefix confirmed up to now by the user is «Newton is one of the greatest scientists w», and the system should now offer a continuation for it.

Table 1. Example 1; the user's desired translation and system's translation hypotheses

User's desired translation :	Newton is one of the greatest scientists who discovered gravity	
Translation hypothesis 1 :	one of the greatest scientists is Newton who discovered gravity	Score: -5.25
Translation hypothesis 2 :	Newton gravity one of the greatest discovered which is the greatest	Score:-10.23

For offering the rest of the translation by the method based on the edit distance, the translation hypothesis whose prefix has less edit distance with the confirmed prefix, should be chosen.

The least edit distance for each of the hypotheses is shown in Table 2. The first hypothesis needs 2 insert operations and 2 delete operations, and the second hypothesis needs just 2 substitutions to match the user's prefix. So the second hypothesis which is not a good translation at all would be chosen, and the completion offered to the user would be «hich is the greatest», which is obviously not desired. Instead, if we consider the translation scores along with the edit distances the first hypothesis could be chosen that has a much better offer for the continuation.

Table 2. The least edit distance between each hypothesis and user's prefix in example 1

		# of edits
1	**Newton is** one of the greatest scientists ~~**is Newton**~~ w|*ho discovered gravity*	4
2	Newton ~~**gravity**~~$^{\textbf{is}}$ one of the greatest ~~**discovered**~~$^{\textbf{scientists}}$ w|*hich is the greatest*	2

As the previous example shows, using only the edit distance to find the best path may lead to a hypothesis which has low quality according to the translation and language models. To overcome this problem, in this approach the total score for each path that a prefix of it matches with the user's prefix, would be a weighted summation of the edit distance with the

translation scores obtained by the statistical models, and in the search process we look for a hypothesis with best total score.

4.3. Edit Distance with Jump Operation

The weighted edit distance approach somehow overcomes the problem of lack of some reorderings in the search graph indirectly in some cases. In order to targeting this problem and resolving it directly, in this paper we propose another method that gives more freedom to the edit distance based searching by adding another operation, called jump operation.

To better explain the aim of adding jump operation and the effect that it could have on the system's offers, consider the example shown in Table 3. In this example, alike the previous one we have the user's desired translation and two system's translation hypotheses, except that here both hypotheses are good enough translations.

Table 3. Example 2; the user's desired translation and system's translation hypotheses

desired translation :	Newton who discovered gravity is one of the greatest scientists	
Translation hypothesis 1:	one of the greatest scientists is Newton who discovered gravity	Score:-5.25
Translation hypothesis 2:	Newton is one of the greatest scientists who discovered gravity	Score:-5.62

Assume that the user's confirmed prefix is «Newton who discovered », and the system should now give an offer. For this purpose the minimum edit distance between the confirmed prefix and any prefix of each of the translation hypotheses should be computed, which is shown in Table 4.

Table 4. The least edit distance between each hypothesis and user's prefix in example 2

		edits
1	~~one~~$^{\text{Newton}}$ ~~of~~$^{\text{who}}$ ~~the~~$^{\text{discovered}}$ |*greatest scientists is Newton who discovered gravity*	3
2	Newton ~~is~~$^{\text{who}}$ ~~one~~$^{\text{discovered}}$ |*of the greatest scientists who discovered gravity*	2

According to Table 4, if we use only the edit distance, the second hypothesis will be chosen and the system's offer would be "*of the greatest scientists who discovered gravity*". Also if we use the weighted edit distance method, the first hypothesis might be selected due to its better translation score, and the system might offer "*greatest scientists is Newton who discovered gravity*". But obviously none of these offers are desirable by the user.

This problem caused by the fact that we consider the alignment between the user's prefix and the prefix of the hypothesis is monotone and with no reordering, while due to the weakness of translation and reordering models and also the pruning done during the generation of the search graph, many of the reordering possibilities might not be produced or might get low scores and be pruned. So in some cases, the part that the user translates at the beginning is at the end or at the middle of the system's translations. In these cases, if we could shift the parts of the hypothesis that match with the user's prefix to the beginning of the sentence and arranged the remained parts in a good order (as more than one segments might be remained, that needs concatenating to make a single segment), we could propose a better offer. For instance, in the hypothesis 2 of the above example, if we could shift "*who discovered*

gravity" after the word "*Newton*" at the beginning of it, we could get the user's desired translation.

For this purpose, we define jump operation in the edit distance. With this operation we can jump from some parts of each translation hypothesis and after matching the user's prefix completely, we collect the parts that we jumped over with the last remained part and arrange them in a good order to generate the system's offer.

As using the jump operation without any limitations enlarges the search space exponentially, we just allow one jump with variable length over multiple edges in each path in the search graph. This is equivalent to allowing just one jump over some contiguous phrases in each system's translations. Also the jump operation could add an error to the edit distance value. Different amounts of errors considered for the jump operation, which will be discussed in the experiments, but the best was adding one unit of error for each jump with any length.

In the above example we can match "*Newton*" in user's prefix and hypothesis 2, then by jumping from "*is one of the greatest scientists*", we match the part "*who discovered*" with the rest of the user's prefix, and the part "*gravity*" will remain at the end of the hypothesis. So, with just one jump operation we could match the user's prefix and the hypothesis.

The subject which is left is how to arrange the jumped over and remained parts to generate the final offer. As we could have at most one jumped over part and one ending part, we consider the two permutations of attaching these two parts; i.e. first the jumped over part comes and then the end part, or first the end part comes and then the jumped over part. Then for selecting between these two states we use language model, and the one whose language model score according to the user's prefix is more will be chosen.

For the above example, the two sentences "***Newton who discovered*** *is one of the greatest scientists gravity*" and "***Newton who discovered*** *gravity is one of the greatest scientists*" will be scored with the language model, and as the second one is more well-formed the final offer to the user would be "*gravity is one of the greatest scientists*".

The pseudocode of the search algorithm based on weighted edit distance with jump operation is shown in Figure 1. The base of this implementation is like the edit distance approach presented in (Koehn, 2009). This algorithm associates backpointers with each hypothesis (or node) in the graph that point back to the lowest error path with which it can be reached. Each backpointer saves the number of edit distance errors, the translation score and the position in the user's prefix that the edit distance is computed up to there till the backpointer's associated hypothesis. As each hypothesis may match the confirmed prefix at different positions, there are multiple backpointers for each hypothesis.

For adding the jump operation a variable named jumpState added to backpointers, which shows the state of the jump operation till that backpointer. If no jump had been occurred until this point of the path, this variable is zero, if in the previous step of the backpointer's path a jump happened, this variable should be 1 (means the jump could continue in the next step or terminate here), and if a jump had been done and finished before this backpointer in its path, the jumpState value would be 2.

The algorithm starts from the first node (empty hypothesis) and iterates over the nodes in the search graph in the topological order. When examining each node's backpointers, all forward transitions (edges of the search graph) are examined using a string edit distance between the remaining prefix and the words in the transition target phrase (line 10). This may consume the remaining prefix, and possibly lead to a new best path whose total score is better than the best total score found so far (lines 11-18). Otherwise, new backpointers for the forward states are created (lines 19-21).

Input: user prefix (u), search graph (g), weight of edit distance error (α)
Output: best completion offer

1. $bestTotalScore = -\infty$, $bestPath = \{\}$
2. Add backpointer (*Score=0.0, Error=0, prefixMatched=0, jumpState=0*) to start state
3. **for all** state $s \in G$ in topologically increasing order **do**
4. **for all** backpointer *b* of state *s* **do**
5. **if** $bestTotalScore \leq b.Score + s.forwardScore + \alpha * b.Error$ **then**
6. **for all** transition *t* from state *s* **do**
7. **if** $b.jumpState == 0 \,||\, b.jumpState == 1$ **then**
8. ProcessJump(*S, b, t*)
9. **end if**
10. Compute edit distance matrix for end part of *u* after *b.prefixMatched* and *t.phrase*
11. **for all** matches *m* in matrix that consumed all of *u* **do**
12. new score $c = b.Score + t.Score + t.toState.forwardScore$
13. new error $e = b.Error + m.Error$
14. **if** $c + \alpha * e > bestTotalScore$ **then**
15. set this as *bestPath*
16. $bestTotalScore = c + \alpha * e$
17. **end if**
18. **end for**
19. **for all** matches *m* in matrix that consumed all of *t.phrase* **do**
20. ProcessMatch(S, b, t, m)
21. **end for**
22. **end for**
23. **end if**
24. **end for**
25. **end for**
26. **return** the best ordering for jumped part and the end part in *bestPath* using LM

Figure 1. The Pseudocode for searching based on weighted edit distance with jump operation

Also, for each forward transition of a specific backpointer, the jumpState of that backpointer will be checked and if it is 0 or 1 it means that a jump could be done over that forward transition. Therefore, this operation performs and the new backpointer for this case is also created and added to the forward state's backpointers (lines 7-9).

The ProcessJump function, shown in Figure 2, will make the new backpointer for performing the jump operation at each state. The error which is considered for this case is one if it is the beginning of a jump, and zero otherwise; i.e. for each jump with any length, 1 unit of error is considered (line 5).

Also, the ProcessMatch function will make the new backpointers for each case of matching a part of user's prefix with the translation phrase produced at the processing transition, at each stage. The jumpState for the new backpointer will remain zero if the previous backpointer's jumpState was zero (means that the jump operation could be performed later in the path), and it would be 2 otherwise (which means the jump operation was performed before, its finished and could no longer be used in the rest of this path) (line 6).

ProcessJump(state S, backpointer b, transition t)

1. reached state $s' = t.toState$
2. create new backpointer b'
3. $b'.BackState = S$
4. $b'.Score = b.Score + t.Score$ (the translation score till here)
5. $b'.Error = b.Error + (b.\ jumpState == 0\ ?1:0)$ (number of errors till state s')
6. $b'.\ prefixMatched = b.\ prefixMatched$
   (the index in the user's prefix that the edit distance error is computed till there)
7. $b'.\ jumpState = 1$
8. $b'' =$ *backpointer of state* s' *with prefixMatched equals* $b'.\ prefixMatched$ *and jumpState equal to* 1
9. **if** b'' not defined **or** $b''.Score + \alpha * b''.Error < b'.Score + \alpha * b'.Error)$
10. *set* b' *new backpointer for state* s' *instead of* b''
11. **end if**

Figure 2. The Pseudocode for adding a jump operation

ProcessMatch(state S, backpointer b, transition t, match m)

1. reached state $s' = t.toState$
2. create new backpointer b'
3. $b'.BackState = S$
4. $b'.Score = b.Score + t.Score$, $b'.Error = b.Error + m.Error$
5. $b'.\ prefixMatched = m.\ prefixMatched$
6. $b'.\ jumpState = (b.\ jumpState == 0\ ?0:2)$
7. $b'' =$ *backpointer for state* s' *with prefixMatched equals* $b'.\ prefixMatched$ *and jumpState equal to* $b'.\ jumpState$
8. **if** b'' not defined $b''.Score + \alpha * b''.Error < b'.Score + \alpha * b'.Error$
9. *set* b' *new backpointer for state* s' *instead of* b''
10. **end if**

Figure 3. The Pseudocode for adding a match case

5. Experiments

In this section, we discuss the experiments done to evaluate the searching algorithms presented in the previous section for interactive prediction.

5.1. Evaluation Metrics

The common way for evaluating the performance of the interactive predictive machine translation systems automatically is to estimate the effort of a human translator to produce correct translations using that system. For this purpose, the translating process of the user is simulated using a set of reference translations as the desired translations in the user's mind.

In this simulation, for each given source sentence in the test set, the translation is produced by the interactive MT system at the first step. Then it's compared with the reference

given for that source sentence, and the character-level longest common prefix of them will be computed. Afterwards, the first non-matching character is replaced by the corresponding reference character and then the system offers a new completion for the given prefix. This process is iterated until a full match with the reference is obtained. Each time a part of the new system's offer is accepted by the user and it expands the prefix, a mouse action is done, and each time the new character is added to the prefix from the reference a keystroke is considered to be done by the user.

The measure used here for evaluation is KSMR[6], which is the overall number of interactions, i.e. keystrokes and mouse-actions, done to translate the whole test set divided by the number of running characters in the reference translations.

5.2. Translation System Description

In this work, we use a Farsi to English phrase-based translation system to evaluate our proposed method. We use Giza++ (Och and Ney, 2003), to train the translation models and Moses Toolkit[7] (Koehn et al., 2007) as the MT decoder, as well as the SRILM[8] (Stolcke, 2002) tools to build the language model. The interactive system is implemented as a server using Moses decoder, which gets a new source sentence or the user's confirmed prefix each time and gives the continuation offer to the client, while the client simulates the user's interactions.

The statistics of the corpora used for training the translation system (Jabbari et al., 2012) and the test set is shown in Table 5.

Table 5. The statistics of the training and testing corpora

Corpora		# of Sentences	# of Running Words	# of Lexicons
Train	Farsi	2164408	37356049	394697
	English		36770830	314372
Test	Farsi	200	4524	
	English		4431	

5.3. Results

In our experiments, at first we evaluate each of the search methods discussed in section 4, to compare their performance, and to study the effect of each improvements done to the baseline. For the weighted edit distance method, the weight of the edit distance error is tuned on a development set and is finally set to -0.5. Also for the jump operation error, three different cases are considered; number of words we jumped over, number of phrases we jumped over and 1. The results of the KSMR and the average response time for each interaction are given in Table 6 for each of the methods.

[6] Keystroke and mouse-action ratio
[7] https://github.com/moses-smt/mosesdecoder
[8] http://www.speech.sri.com/projects/srilm/

Table 6. Evaluation results of different search methods

Searching Method		KSMR (%)	Average Response Time (s)
Edit Distance Method		31.78	0.21
Adding Jump to Edit Distance Method	Jump Error = # of Words	31.62	0.93
	Jump Error = # of Phrases	31.23	1.09
	Jump Error = 1	30.43	1.67
Weighted Edit Distance Method		31.36	0.27
Adding Jump to Weighted Edit Distance Method	Jump Error = # of Words	30.31	1.1
	Jump Error = # of Phrases	30.2	1.42
	Jump Error = 1	30.15	1.65

As the results shows, using the weighted sum of edit distance error and translation scores makes 0.42% improvements in KSMR and no significant growth in response time, compared to the base edit distance method.

Also, adding the jump operation to the edit distance method shows that it could improves its KSMR up to 1.35%, when the jump error is set to one regardless of the jump length. The results of the other settings for jump error are not as good as considering it to be one, which shows assuming it to be the number of words or phrases that we jumped over is a high error, which prevents some long jumps that might be useful.

Furthermore, when we combine weighted edit distance method with jump operation, we get 1.21% improvements in KSMR, in comparison to the weighted method itself, and 1.63% improvements compared to the base edit distance method. This shows that these two improving methods do not have much overlap in the cases that they improve the baseline system.

Despite the improvements in KSMR, the response time while we add the jump operation increases significantly. This increase can be expected as the number of backpointers created in the search process grows with adding jump operation, and the search space becomes much greater. For solving this issue we limit the jump operation more, by putting a threshold for the maximum amount of phrases (edges in the graph) that we could jump over. In the next experiment, the effect of this threshold is studied, and the results are shown in Table 7.

As the results in Table 7 shows, if we assume that the response time below 1 second is acceptable, then by selecting the threshold equal to 5, we can gain about 0.82% improvement in KSMR from the weighted edit distance method without jump and 1.24% improvement compared to the base edit distance method.

Table 7. Results of the effect of limiting the jump length

Searching Method		KSMR (%)	Average Response Time (s)
	Maximum Jump Length		
Adding Jump to Weighted Edit Distance Method	1	31.16	0.79
	2	30.92	0.8
	3	30.71	0.9
	4	30.64	0.93
	5	30.54	0.96
	6	30.41	1.04
	8	30.27	1.13
	10	30.23	1.23
	∞	30.15	1.65

6. Conclusions and Future Work

In this paper we aimed to improve a common search method used for the interactive prediction in the computer assisted translation systems, which searches for a path in the search graph, whose prefix has least edit distance with the user's confirmed prefix.

As the reordering of words which is desired by the user might not be in the search graph, the simple edit distance method has the problem of not considering the differences in reorderings between the system's translation and the user's prefix. In order to solve this problem, we propose adding a jump operation to the edit distance, which allows jumping over a contiguous sequence of phrases in a system's translation while matching the user's prefix with that translation, and then add the part that it jumped over to the continuation offer of the system.

The results show that adding jump to the edit distance improved the KSMR measure by 1.35%. Also, we show that this method does not have much overlap with another method which has been proposed for this purpose and used the weighted sum of edit distance and translation scores to select the best offer. Combining these two methods improved the base edit distance method by 1.63% in KSMR.

However, adding jump operation increases the average response time of the system in each interaction. This is because of the significant increase in the number of backpointers generated in the search process. We limit the length of the jumps to reduce the response time. Instead, the mechanisms could be presented to prune the inappropriate backpointers, and prevent them from further processing, so that we could increase the response speed with less reduction in KSMR

References

Alabau, V., Buck, C., Carl, M., Casacuberta, F., Garcıa-Martınez, M., Germann, U., et. al. (2014). Casmacat: A computer-assisted translation workbench. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. 25-28.

Barrachina, S., Bender, O., Casacuberta, F., Civera, J., Cubel, E., Khadivi, S., Lagarda, A., Ney, H., Tomás, J., Vidal, E. and Vilar, J. M. (2009). Statistical approaches to computer-assisted translation. *Computational Linguistics* 35, no. 1: 3-28.

Bender, O., Hasan, S., Vilar, D., Zens, R., and Ney, H. (2005). Comparison of generation strategies for interactive machine translation. *Proceedings of the 10th Annual Conference of the European Association for Machine Translation (EAMT 05).* 33-40.

Federico, M., Bertoldi, N., Cettolo, M., Negri, M., Turchi, M., Trombetti, M., et. al. (2014). The MateCat tool. In *Proceedings of COLING.* 129-132.

Foster, G., Isabelle, P., and Plamondon, P. (1997). Target-Text Mediated Interactive Machine Translation. *Machine Translation* 12, no. 1-2: 175-194.

González-Rubio, J., Ortiz-Martínez, D., Benedí, J., and Casacuberta. F. (2013) Interactive Machine Translation using Hierarchical Translation Models. *EMNLP*. 244-254.

Huang, C., Yang, P., Chen, K., and Chang, J.S. (2012). TransAhead: a computer-assisted translation and writing tool. *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Association for Computational Linguistics, 352-356.

Jabbari, F., Bakhshaei, S., Mohammadzadeh Ziabary, S. M., and Khadivi, S. (2012). Developing an Opendomain English-Farsi Translation System Using AFEC: Amirkabir Bilingual Farsi-English Corpus. *Association for MachineTranslation in the Americas (AMTA).*

Koehn, P. (2009). A process study of computer-aided translation. *Machine Translation* 23, no. 4: 241-263.

Koehn, P., and Haddow, B. (2009). Interactive assistance to human translators using statistical machine translation methods. *MT Summit XII.*

Koehn, P., et al. (2007) Moses: Open source toolkit for statistical machine translation. *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions.* Association for Computational Linguistics. 177-180.

Langlais, P., Foster, G., and Lapalme, G. (2000). TransType: a computer-aided translation typing system. *Proceedings of the 2000 NAACL-ANLP Workshop on Embedded machine translation systems-Volume 5.* Association for Computational Linguistic. 46-51.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady* 10, no. 8: 707-710.

Och, F.J., and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational linguistics* 29, no. 1: 19-51.

Och, F.J., Zens, R., and Ney, H. (2003). Efficient search for interactive statistical machine translation. *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1.* Association for Computational Linguistics. 387-393.

Ortiz-Martínez, D. (2011). *Advances in Fully-Automatic and Interactive Phrase-Based Statistical Machine Translation.* PhD thesis, Universitat Politècnica de València.

Ortiz-Martınez, D., Garcıa-Varea, I., and Casacuberta, F. (2009). Interactive machine translation based on partial statistical phrase-based alignments. *RECENT ADVANCES IN NATURAL LANGUAGE PROCESSING*. 330-336.

Slocum, J. (1985). A survey of machine translation: its history, current status, and future prospects. *Computational linguistics* 11, no. 1 (1985): 1-17.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A Study of Translation Edit Rate with Targeted Human Annotation. *Proceedings of Association for Machine Translation in the Americas*.

Stolcke, A. (2002). SRILM-an extensible language modeling toolkit. *Proc. Intl. Conf. on Spoken Language Processing*, vol. 2, pp. 901-904, Denver.

Tomás, J., and Casacuberta, F. (2006). Statistical phrase-based models for interactive computer-assisted translation. *Proceedings of the COLING/ACL on Main conference poster sessions.* Association for Computational Linguistics. 835-841.

Vakil, Z., and Khadivi, S. (2012a). A New Search Approach for Interactive-Predictive Computer-Assisted Translation. *COLING (Posters)*. 1261-1270.

Vakil, Z., and Khadivi, S. (2012b). Interactive-predictive speech-enabled computer-assisted translation. *International Workshop on Spoken Language Translation (IWSLT)*. 237-243.

Whitelock, P. J., Wood M. M., Chandler, B. J., Holden, N., and Horsfall, H. J. (1986) Strategies for interactive machine translation: the experience and implications of the UMIST Japanese project. *Proceedings of the 11th coference on Computational linguistics.* Association for Computational Linguistics. 329-334.