Refinements to Interactive Translation Prediction Based on Search Graphs

Philipp Koehn[◇⋆], **Chara Tsoukala**[⋆] **and Herve Saint-Amand**[⋆]
[◇]Center for Speech and Language Processing, The Johns Hopkins University
[⋆]School of Informatics, University of Edinburgh
phi@jhu.edu, ctsoukal@inf.ed.ac.uk, hsamand@inf.ed.ac.uk

Abstract

We propose a number of refinements to the canonical approach to interactive translation prediction. By more permissive matching criteria, placing emphasis on matching the last word of the user prefix, and dealing with predictions to partially typed words, we observe gains in both word prediction accuracy (+5.4%) and letter prediction accuracy (+9.3%).

1 Introduction

As machine translation enters the workflow of professional translators, the exact nature of this human-computer interaction is currently an open challenge. Instead of tasking translators to post-edit the output of machine translation systems, a more interactive approach may be more fruitful.

One such idea is interactive translation prediction (Langlais et al., 2000b): While the user writes the translation for a sentence, the system makes suggestions for sequent words. If the user diverges from the suggestions, the system recalculates its prediction, and offers new suggestions. This input modality is familiar to anybody who has used auto-complete functions in text editors, cell phones, or web applications.

The technical challenge is to come up with a method that predicts words that the user will accept. The standard approach to this problem uses the search graph of the machine translation system. Such search graphs may be recomputed in a constraint decoding process restricted to the partial user input (called the *prefix*), but this is often too slow with big models and limited computing resources, so we use static word graphs.

The user prefix is matched against the search graph. If the user prefix cannot be found in the search graph, approximate string matching is used by finding a path with minimal string edit distance, i.e., a path in the graph with the minimal number of insertions, deletions and substitutions to match the user prefix.

This paper presents a number of refinements to extend this approach, by allowing more permissive matching criterion, placing emphasis on matching the last word of the user prefix, and dealing with predictions to partially typed words. We show improvements in word prediction accuracy from 56.1% to 60.5% and letter prediction accuracy from 75.2% to 84.5% on a publicly available benchmark (English-Spanish news translation).

2 Related Work

The interactive machine translation paradigm was first explored in the TransType and TransType2 projects (Langlais et al., 2000a; Foster et al., 2002; Bender et al., 2005; Barrachina et al., 2009). Given the computational cost and need for quick response time, most current word operates on search graphs (Och et al., 2003). Such search graphs can be efficiently represented and processed with finite state tools (Civera et al., 2004). More recently, the approach has been extended to SCFG-based translation models (González-Rubio et al., 2013).

There are several ways the sentence completion predictions can be presented to the user: showing the complete sentence prediction, only a few words, or multiple choices. User actions may be also extended to mouse actions to pinpoint the divergence from an acceptable translation (Sanchis-Trilles et al., 2008), or hand-writing (Alabau et al., 2011) and speech modalities (Cubel et al., 2009).

3 Properties of Core Algorithm

Our implementation of the core algorithm follows closely Koehn (2009). It is a dynamic programming solution that computes the minimal cost to reach each node in the search graph by matching parts of the user prefix. Cost is measured primarily in terms of string edit distance (number of deletions, insertions and substitutions), and secondary in terms of translation model score for the matched path in the graph. Search is done iteratively, with an increasing number of allowable edits.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 574–578,
Baltimore, Maryland, USA, June 23-25 2014.

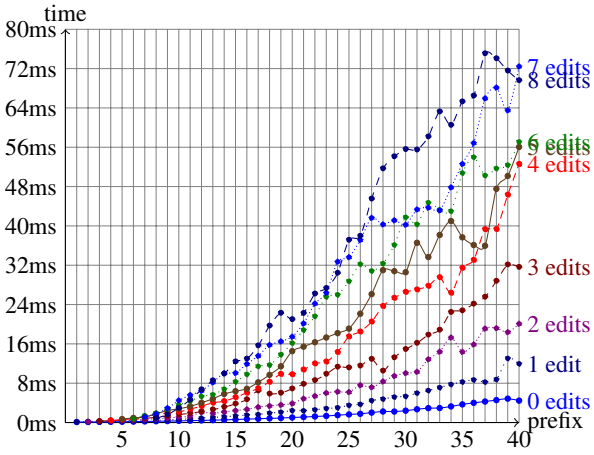

Figure 1: Average response time of baseline method based on length of the prefix and number of edits: The main bottleneck is the string edit distance between prefix and path.

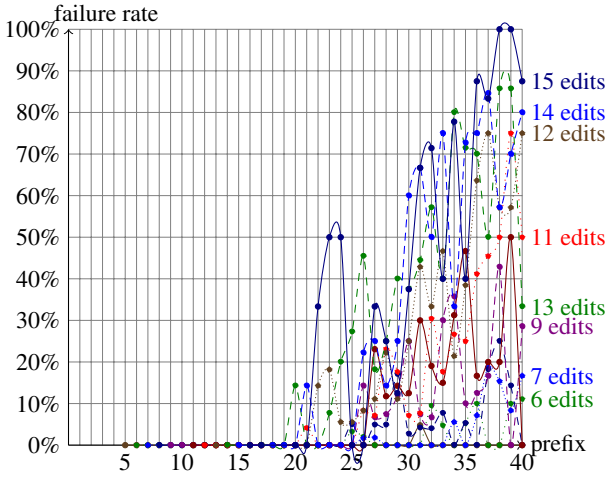

Figure 2: Ratio of prefix matching processes abandoned due to exceeding the 100ms time limit (showing only curves with a minimum of 5 edits).

3.1 Experimental Setup

Given the large number of proposed variations of the algorithm, we do not carry out user studies, but rather use a simulated setting. We predict translations that were crafted by manual post-editing of machine translation output. We also use the search graphs of the system that produced the original machine translation output.

Such data has been made available by the CASMACAT project[1]. In the project's first field trial[2], professional translators corrected machine translations of news stories from a competitive English–Spanish machine translation system (Koehn and Haddow, 2012). This test set consists of 24,444 word predictions and 141,662 letter predictions.

3.2 Prediction Speed

Since the interactive translation prediction process is used in an interactive setting where each key stroke of the user may trigger a new request, very fast response time is needed. According to standards in usability engineering

> *0.1 second is about the limit for having the user feel that the system is reacting instantaneously* (Nielsen, 1993).

So, this is the time limit we have to set ourselves to predict the next words of a translator.

What are the main factors that influence processing time in our core algorithm? See Figure 1 for an illustration. We plot processing time against the length of the user prefix and the string edit distance between the user prefix and the search graph. The graph clearly shows that the main slowdown in processing time occurs when the edit rate increases.

To guarantee a response in 100ms, the algorithms aborts when this time is exceeded and relies on a prediction based on string edit distance against the best path in the graph. The larger the number of edits, the more often this occurs, as Figure 2 shows.

3.3 Accuracy

We are mainly interested in the accuracy of the method: How often does it predict a word that the user accepts? There is a trade-off between speed and accuracy.

One way we can balance this trade-off is by removing nodes from the search graph. By threshold pruning (Sanchis-Trilles and Ortiz-Martínez, 2014), we remove nodes from the search graph that are only part of paths that are worse than the best path by a specified score difference.

See Table 1 how the choice of the score difference threshold impacts failure rate and accuracy. A wider threshold has the potential to achieve better results (if we allows for up to 1 second of processing time), but with the constraint of 100ms response time, the optimum is with a threshold of 0.4. Wider thresholds lead to a higher failure rate, causing overall lower accuracy.

[1] http://www.casmacat.eu/

[2] http://www.casmacat.eu/uploads/Deliverables/d6.1.pdf

Threshold	100ms Max		1000ms Max	
	Acc.	Fail	Acc.	Fail
0.3	55.8%	4.5%	56.9%	0.0%
0.4	**56.1%**	**6.5%**	**58.0%**	**0.0%**
0.5	55.9%	9.0%	58.8%	0.0%
0.6	55.5%	11.6%	59.4%	0.0%
0.8	54.4%	17.1%	59.4%	0.0%
1.0	52.7%	21.7%	58.6%	6.5%

Table 1: Impact of threshold pruning on search accuracy and failure rate (i.e., failure to complete search in given time and resorting to matching against best translation).

4 Refinements

We now introduce a number of refinements over the core method. Given the constraints established in the previous section (maximum response time of 100ms, pruning threshold 0.4), we set out to improve accuracy.

4.1 Matching Last Word

The first idea is that it is more important to match the last word of the user prefix than having mismatches in earlier words. We attempt to find the last word in the predicted path either before or after the optimal matching position according to string edit distance.

We combine the matched path in the prefix with the optimal suffix, and search for the last user prefix word within a window. This means that we either move words from the suffix to the prefix or the other way around, without changing the overall string along the path.

Table 2 shows the impact on accuracy for different window sizes. While we expected some gains by checking for the word somewhere around the optimal position in the predicted path, we do see significant gains by not placing any restrictions to where the word can be found, except for a bias to less distant positions. For instance, examining a window of up to 3 words gives us a word prediction accuracy of 57.2% versus the 56.1% baseline. Finding the last word anywhere boosts performance to 59.1%.

The table also reports accuracy numbers when we allow the process to run up to 1 second — which is basically an exhaustive search but not practically useful. These numbers shed some light on why an unlimited window size in matching the last word helps: the gains come partially from the cases where the initial search fails. Finding the last user word anywhere in the machine translation output is a better fallback than computing optimal string edit distance. Analysis of the data suggests that gains mainly come from large length mismatches between user translation and machine translation, even in the case of first pass searches.

Window	100ms Max	1000ms Max
baseline	56.1%	58.0%
1 word	56.6%	58.4%
2 words	56.9%	58.6%
3 words	57.2%	58.9%
5 words	57.8%	59.3%
anywhere	59.1%	59.5%

Table 2: Search for the last prefix word in a window around the predicted position in the matched path.

Word Matching	100ms Max	1000ms Max
baseline	59.1%	59.5%
case-insensitive	58.7%	59.4%

Table 3: Search with case-insensitive word matching (say, *University* and *university*).

4.2 Case-Insensitive Matching

Some mismatches between words matter less than others. For instance, if the user prefix differs only in casing from the machine translation (say, *University* instead of *university*), then we may still want to treat that as a word match in our algorithm. However, as Table 3 shows, allowing case-insensitive matching leads to lower accuracy (58.7% vs. 59.1%).

A major reason is computational cost. The most inner loop in the algorithm compares words. This is optimized by representing words as integers. However, if we allow case-insensitive matching, this simple method does not work anymore. We do precompute approximate word matches and store matching words identifiers in a hash map, but still the ratio of searches that do not complete in 100ms increases from 6.5% to 9.7%. By extending the allowable time to 1 second, the accuracy gap is reduced to 0.1%.

4.3 Approximate Word Matching

When a word in the user translation differs from a word in the decoder search graph only by a few letters, then it should be considered a lesser error than substitutions of completely different words. Such word differences may be due to casing, morphological variants, or spelling inconsistencies.

We compute word dissimilarity by computing

Max. Dissimilarity	100ms Max.	1000ms Max.
baseline	59.1%	59.5%
30%	60.2%	61.0%
20%	60.4%	61.3%
10%	60.6%	61.5%

Table 4: Counting substitutions between similar words as half an error. Dissimilarity is measured as letter edit distance

Min Stem / Max Suffix	100ms	1000ms
baseline	59.1%	59.5%
4 / 3	59.4%	60.1%
3 / 3	59.5%	60.2%
2 / 3	59.5%	60.3%

Table 5: Counting substitutions between morphological variants as half an error. Morphological variance is approximated by requiring a minimum number of initial letters to match and a maximum of final letters to differ.

the ratio of letter edit operations to the length of the shorter word.[3] We now set a threshold for maximum dissimilarity, under which mismatched words are considered only half the edit cost of other edit operations.

Table 4 shows that we get significantly higher word prediction accuracy than with the baseline approach (up to 60.6% vs. 59.1%), and the best performance with a 10% threshold. We observe the same computational problem as in the previous section (about 9.2% first pass failures, vs. 6.5%), reflected in a higher accuracy gap for 100ms and 1000ms time limits.

4.4 Stemmed Matching

We suspected that the main benefit of approximate word matching is the better handling of morphological variants. In Spanish, this mainly constitutes itself as different word endings. Thus, we redefine our word dissimilarity measure by consider words similar, if they agree in at least a number of leading letters (presumably the stem), and may differ in at most a number of trailing letters (presumably the morpheme).

Table 5 shows that this is successful in increasing the word prediction rate (59.5% vs. 59.1%) but not as much as with the more general approximate word matching in the previous section (recall: 60.6%).

[3]For instance, if a 6 letter word and a 4 letter word can be matched with two deletions and one substitution, then the dissimilarity score is $\frac{3}{4} = .75$.

#	Method	Word Acc.	Letter Acc.
1	baseline	56.0%	75.2%
2	1+matching last word	59.0%	80.6%
3	2+case insensitive	58.7%	80.4%
4	2+dissimilarity 10%	60.5%	80.6%
5	2+stem 2/3	59.4%	80.5%
6	4+desperate	60.5%	84.5%

Table 6: Extending the approach to word completion. Impact of refinements of letter prediction accuracy with additional desperate word matching against the entire vocabulary.

5 Word Completion

Besides word prediction, word completion is also a useful feature in an interactive translation tool. When the machine translation system decides for *college* over *university*, but the user types the letter *u*, it should change its prediction.

To enable word completion in the canonical algorithm, we allow matching of the final user word (if not followed by a space character) as a prefix of any word as a zero cost operation. The predicted suffix that is returned to the user then starts with the remaining letters of the word in the path.

Table 6 shows that the refinements that helped sentence completion also benefit word completion. From a baseline accuracy of 75.2% correctly predicted letters, we reach up to 80.6%. Note that the baseline word prediction accuracy is slightly lower (56.0% vs. 56.1%) than in the previous experiments, since the previously correctly matched last word may be mistaken as the prefix of another word.

We add an additional refinement to this task: If the potentially incomplete final word of the user prefix cannot be found in the predicted path, then we explore the entire vocabulary from the unpruned search graph for completions. If multiple words match, the one with the highest path score is used. This *desperate* word completion method gives significant gains (84.5% over 80.6%).

6 Conclusion and Future Work

We observe most improvements by a focus on the last word of the user prefix and approximate word matching. This suggests that there may be additional gains by a stronger focus on the tail of the user prefix. Also, the findings from the time/productivity tradeoffs indicate that more time efficient algorithms and implementations should be explored.

Acknowledgements

This work was supported under the CASMACAT project (grant agreement N° 287576) by the European Union 7th Framework Programme (FP7/2007-2013).

References

Alabau, V., Sanchis, A., and Casacuberta, F. (2011). Improving on-line handwritten recognition using translation models in multimodal interactive machine translation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Techologies*, pages 389–394, Portland, Oregon, USA. Association for Computational Linguistics.

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(1).

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

Civera, J., Cubel, E., Lagarda, A. L., Picó, D., González, J., Vidal, E., Casacuberta, F., Vilar, J. M., and Barrachina, S. (2004). From machine translation to computer assisted translation using finite-state models. In Lin, D. and Wu, D., editors, *Proceedings of EMNLP 2004*, pages 349–356, Barcelona, Spain. Association for Computational Linguistics.

Cubel, E., Khadivi, S., Lagarda, A., Ney, H., Toms, J., Vidal, E., and Vilar, J.-M. (2009). Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1).

Foster, G., Langlais, P., and Lapalme, G. (2002). User-friendly text prediction for translators. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 148–155, Philadelphia. Association for Computational Linguistics.

González-Rubio, J., Ortíz-Martinez, D., Benedí, J.-M., and Casacuberta, F. (2013). Interactive machine translation using hierarchical translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 244–254, Seattle, Washington, USA. Association for Computational Linguistics.

Koehn, P. (2009). A process study of computer-aided translation. *Machine Translation*, 23(4):241–263.

Koehn, P. and Haddow, B. (2012). Towards effective use of training data in statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 363–367, Montreal, Canada. Association for Computational Linguistics.

Langlais, P., Foster, G., and Lapalme, G. (2000a). Transtype: a computer-aided translation typing system. In *Proceedings of the ANLP-NAACL 2000 Workshop on Embedded Machine Translation Systems*.

Langlais, P., Foster, G., and Lapalme, G. (2000b). Unit completion for a computer-aided translation typing system. In *Proceedings of Annual Meeting of the North American Chapter of the Association of Computational Linguistics (NAACL)*.

Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann.

Och, F. J., Zens, R., and Ney, H. (2003). Efficient search for interactive statistical machine translation. In *Proceedings of Meeting of the European Chapter of the Association of Computational Linguistics (EACL)*.

Sanchis-Trilles, G. and Ortiz-Martínez, D. (2014). Efficient wordgraph pruning for interactive translation prediction. In *Annual Conference of the European Association for Machine Translation (EAMT)*.

Sanchis-Trilles, G., Ortiz-Martínez, D., Civera, J., Casacuberta, F., Vidal, E., and Hoang, H. (2008). Improving interactive machine translation via mouse actions. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 485–494, Honolulu, Hawaii. Association for Computational Linguistics.