Convergence of Translation Memory and Statistical Machine Translation

Philipp Koehn
University of Edinburgh
10 Crichton Street
Edinburgh, EH8 9AB
Scotland, United Kingdom
pkoehn@inf.ed.ac.uk

Jean Senellart
Systran
La Grande Arche
1, Parvis de la Défense
92044 Paris, France
senellart@systran.fr

Abstract

We present two methods that merge ideas from statistical machine translation (SMT) and translation memories (TM). We use a TM to retrieve matches for source segments, and replace the mismatched parts with instructions to an SMT system to fill in the gap. We show that for fuzzy matches of over 70%, one method outperforms both SMT and TM baselines.

1 Introduction

Two technological advances in the field of automated language translation, translation memory (TM) and statistical machine translation (SMT), have seen vast progress over the last decades, but they have been developed very much in isolation. The reason for this is that different communities played a role in each technology's development.

TMs are a tool for human translators. Since many translation needs are highly repetitive (translation of updated product manuals, or several drafts of legislation), being able to find existing translations of segments of the source language text, alleviates the need to carry out redundant translation. In addition, finding close matches (so-called fuzzy matches), may dramatically reduce the translation workload. Various commercial vendors offer TM software and the technology is in wide use by translation agencies.

Instead of building machine translation systems by manually writing translation rules, SMT systems (Koehn, 2010) are built by fully automatically analyzing translated text and learning the rules. SMT has been embraced by the academic and commercial research communities as the new dominant paradigm in machine translation. Almost all recently published papers on machine translation are published on new SMT techniques. The methodology has left the research labs and become the basis of successful companies such as Language Weaver and the highly visible Google and Microsoft web translation services. Even traditional rule-based companies such as Systran have embraced statistical methods and integrated them into their systems (Dugast et al., 2007).

The two technologies have not touched much in the past not only because of the different development communities (software suppliers to translation agencies vs. mostly academic research labs). Another factor is that TM and SMT have recently addressed different translation challenges. While TM have addressed the need of translation agencies to produce high-quality translations of often repetitive material, SMT has set itself the challenge of open domain translations such as news stories and is mostly satisfied with translation quality that is good enough for gisting, i.e., transmitting the meaning of the source text to a target language speaker (consider web page translation or information gathering by intelligence agencies).

Currently, SMT receives increasing attention by translation agencies, who would like to employ the technology in a workflow of first automatic translation and then human post-editing. One possible user scenario is to offer a human translator a fuzzy match from a TM, or an SMT translation, or both. What to show may be decided by an automatic classifier (Simard and Isabelle, 2009; Soricut and Echihabi, 2010; He et al., 2010) or may be based on fuzzy match score or SMT confidence measures (Specia et al., 2009).

In this paper, we argue that the two technologies have much more in common that commonly perceived. We present a method that integrates TM and SMT and that outperforms either technology for fuzzy matches of more than 80%. In a second method, we encode TM matches as very large translation rules and outperform all other methods for fuzzy match ranges over 70%.

2 XML Method

The main idea of our method is as follows: If we are able to find a sufficiently good fuzzy match for a given source segment in the TM, then we detect the location of the mismatch in source and target of the retrieved TM segment pair, and let the SMT system translate the mismatched area. We use the capability of the Moses SMT decoder (Koehn et al., 2007) to use XML markup to specify required translations for the matched parts of the source sentence, hence forcing it to only translate the unmatched part.

Recent work has explored similar strategies. Motivated by work in EBMT, Smith and Clark (2009); Zhechev and van Genabith (2010) use syntactic information to align TM segments. Then they create an XML frame to be passed to Moses. Both show weaker performance (on different data sets) than we report here. Smith and Clark (2009) never overcomes the SMT baseline. Zhechev and van Genabith (2010) only beat the SMT system in the 90-99% fuzzy match range.

The work by Biçici and Dymetman (2008) is closer to our approach: They align the TM segments using GIZA++ posterior probabilities to indentify the mismatch in the target and add one non-contiguous phrase rule to their phrase-based decoder. They show significant improvements over both SMT and TM baselines, but their SMT seems to perform rather badly — it is outperformed by raw TM matches even in the 74-85% fuzzy match range.

2.1 Example

To illustrate the process (see also Figure 1), let us first go over one example. Let us say that the following source segment needs translation:

The second paragraph of Article 21 is deleted .

The TM does not contain this source segment, but it contains something very similar:

The second paragraph of Article 5 is deleted .

In the TM, this segment is translated as:

À l’ article 5 , le texte du deuxiéme alinéa est supprimé .

The mismatch between our source segment and the TM source segment is the word *21* or *5*, respectively. By letting the SMT system translate the true source word *21*, but otherwise trusting the target side of the TM match, we construct the following (simplified) XML frame:

<xml translation="*À l’ article*"/> *21* <xml translation="*, le texte du deuxiéme alinéa est supprimé .*"/>

Or, to use a more compact formalism for the purposes of this paper:

<À l’ article> 21 <, le texte du deuxiéme alinéa est supprimé .>

The XML frame consists of specified translations (e.g., *À l’ article*) and source words (e.g., *21*). The decoder is instructed to use the specified translations in its output. The remaining source words are translated as usual, by consulting a phrase translation table, and search for the best translation according to various scoring functions including a language model.

In our example, the SMT decoder produces the following output:

À l’ article 21 , le texte du deuxiéme alinéa est supprimé.

A perfectly fine translation.

2.2 Fuzzy Matching

The first processing step is to retrieve the best match from the TM. Such fuzzy matches are measured by a fuzzy match score, and the task is to find the best segment pair in the TM under this score.

There are several different implementation of the fuzzy match score, and commercial products typically do not disclose their exact formula. However, most are based on the string edit distance, i.e., the number of deletions, insertions, and substitutions needed to edit the source segment to the TM segment.

Our implementation of the fuzzy match score uses word-based string edit distance, and uses letter string edit distance as a tie breaker. We define the fuzzy match score as:

$$\text{FMS} = 1 - \frac{\text{edit-distance}(\text{source}, \text{tm-source})}{\max(|\text{source}|, |\text{tm-source}|)}$$

For our example, the word-based string edit distance is 89% (one substitution for 9 words), and the letter-based string edit distance is 95% (one substitution and one deletion for 37 letters, not counting spaces).

2.3 Word Alignment of TM

The mismatch between the source segment and the TM source segment is easy to detect. As with our fuzzy match metric, we compute the string edit distance between the two, which detects the inserted, deleted and substituted words.

A harder problem is to determine which target target words are affected by the mismatch. For this, we need a word alignment between the source words and the target words in the TM segment.

Word alignment is a standard problem in SMT, and many algorithms have been proposed and implemented. Most commonly used is the implementation of the IBM Models (Brown et al., 1993) in the toolkit GIZA++ (Och et al., 1999), combined with symmetrization heuristics. This toolkit is also used by the Moses SMT training pipeline. So, if we build an SMT system on the TM data, then the word alignment falls out as a by-product.

2.4 Construction of XML Frame

We now have all the necessary ingredients to construct the XML frame that we will pass to the SMT decoder. See Figure 1 for an illustration of our example. Given the string edits and the word alignment, we can track the mismatched source word to the TM target word.

We construct the XML frame by subtraction. All of the TM target segment is passed as a specified translation to the SMT decoder, except for the subtraction of the mismatch. The TM target word aligned to the mismatched source word is not part of a specified translation, and instead the source word is inserted in their place.

A mismatch may consist of a sequence of multiple words in source, TM source, or TM target. We treat such a block the same way we treat single words: they are removed as a block from the TM target and the source block is inserted.

There may be multiple non-neighboring mismatched sequences. We treat each separately and perform the subtraction process for each.

2.5 Special Cases

There are a number of special cases (also illustrated in Figure 2):

Pure insertion: If the source segment has an inserted block that is not present in the TM source segment, we add these source words to the XML frame. The location of the words is after the TM target word aligned to the TM source word just prior to the insertion point.

Pure deletion: If the TM source segment has additional words, then these are removed from the specified translation in the XML frame.

Unaligned mismatched words: If the mismatched source words are unaligned, then we find the closest aligned previous source word and specify them after this target position.

Non-contiguous alignments: If a TM source segment has non-contiguous alignments to the TM target segment, we first try to resolve this by splitting up blocks. If this does not help, then the mismatched source words are placed at the position of the first aligned TM target block.

The basic principles of the heuristic are: all mismatched source words are inserted; all TM target words aligned to mismatched TM source words are removed; if the alignment to the TM target words fails, then go to the previous TM source word and follow its alignment.

The detailed pseudo-code of the algorithm is given in Figure 3.

3 Experiments

We carried out experiments using two data sets: an English–Italian commercial product manual corpus

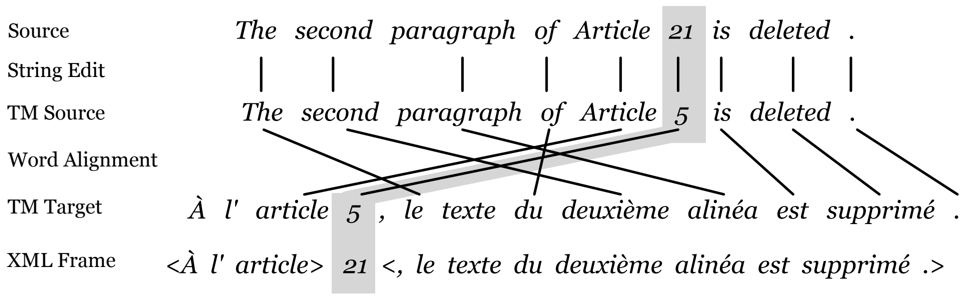

Source
String Edit
TM Source
Word Alignment
TM Target
XML Frame
The second paragraph of Article 21 is deleted .
The second paragraph of Article 5 is deleted .
À l' article 5 , le texte du deuxième alinéa est supprimé .
<À l' article> 21 <, le texte du deuxième alinéa est supprimé .>

Figure 1: **Construction of the XML frame:** The mismatched source word is tracked to the TM target word.

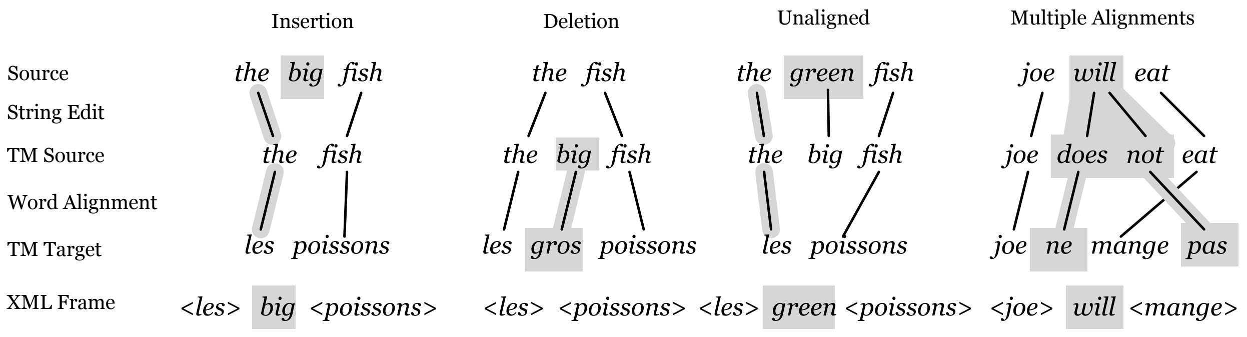


Figure 2: **Special Cases:** The basic principles are: all mismatched source words are inserted, all TM target words aligned to mismatched TM source words are removed, if the alignment to the target words fails, go to previous word and follow its alignment.

Acquis	Corpus	Test
segments	1,165,867	4,107
English words	24,069,452	129,261
French words	25,533,259	135,224

Product	Corpus	Test
segments	83,461	2,000
English words	1,038,762	24,643
French words	1,110,284	26,248

Table 1: Statistics of the corpus used in experiments

(Product) and the English–French part of the publicly available JRC-Acquis corpus[1] (Acquis), for which we use the same test set as Koehn et al. (2009). See Table 1 for basic corpus statistics.

The Acquis corpus is a collection of laws and regulations that apply to all member countries of the European Union. It has more repetitive content than the parallel corpora that are more commonly used in machine translation research. Still, the commercial Product corpus is more representative of the type of data used in TM systems. It is much smaller (around a million words), with shorter segments (average 12 words per segments).

We are especially interested in the performance of the methods for sentences for which we find highly scoring fuzzy matches, since we do not expect fuzzy matches to be very useful otherwise. See Table 2 for statistics on the subsets based on fuzzy match ranges. The sentences with 100% fuzzy match are much shorter, but otherwise there is no strong correlation between fuzzy match score and sentence length. Note that we do not break out a 100% fuzzy match subset for the Product corpus, since the test data is drawn from a TM, so we do not expect to find valid 100% matches (since we exclude the test data from the TM used in training and testing).

3.1 SMT Training

We train SMT systems using the Moses toolkit. We use a standard setup with lexicalized reordering and 5-gram language model.

[1]http://wt.jrc.it/lt/Acquis/ (Steinberger et al., 2006)

```
Input: match path $M$,
    input $\mathbf{i}$,
    TM source $\mathbf{s}$,
    TM target $\mathbf{t}$,
    alignment $\mathbf{a}$
Output: xml frame
 1: global $M$, $\mathbf{i}$, $\mathbf{s}$, $\mathbf{t}$, $\mathbf{a}$, insertion
 2: $pos_i$ = 0;
 3: $pos_s$ = 0;
 4: matching = false;
 5: matched-target($[0; |\mathbf{t}|]$) = true // matched words
 6: for all edit $e \in M$ do
 7:   if matching AND e != 'match' then
 8:     // beginning of mismatch
 9:     $start_i$ = $pos_i$;
10:     $start_s$ = $pos_s$;
11:     matching = false;
12:   else if !matching AND e=='match' then
13:     // end of mismatch
14:     mismatch($start_i$,$pos_i$-1, $start_s$,$pos_s$-1)
15:     matching = true;
16:   end if
17:   $pos_i$++ unless edit == 'deletion'
18:   $pos_s$++ unless edit == 'insertion'
19: end for
20: if !matching then
21:   mismatch($start_i$,$pos_i$,$start_s$,$pos_s$)
22: end if
23: return construct-xml()

function mismatch:
Input: $start_i$,$end_i$, $start_s$,$end_s$
 1: // remove use of affected target words
 2: for all source positions $s \in [start_s; end_s]$ do
 3:   for all target positions $t \in \mathbf{a}(s)$ do
 4:     matched-target(t) = false;
 5:   end for
 6: end for
 7: // find insertion position
 8: $ins_t$ = min( $\{t \in \mathbf{a}([start_s; end_s])\}$ )
 9: while $ins_t$ undefined do
10:   $ins_t$ = min( $\{t \in \mathbf{a}(--start_s)\}$ )
11: end while
12: if $ins_t$ undefined then
13:   $ins_t$ = -1 // sentence start
14:   $ins_t$ = $|\mathbf{i}|$ if $end_i == |\mathbf{i}|$ // end
15: end if
16: // locate inserted words
17: insertion[$ins_t$] = $\mathbf{i}[start_i;end_i]$

function construct-xml:
 1: xml = ""
 2: included = false
 3: $start_t = -1$
 4: for all target positions $t \in [0; |\mathbf{t}|[$ do
 5:   if !included AND matched-target($t$) then
 6:     $start_t = -1$
 7:     included = true
 8:   else if included AND (!matched-target($t$) OR
      insertion[$t$]) then
 9:     if $start_t \geq 0$ then
10:       xml += "<xml translation="
11:       xml += $\mathbf{t}[start_t, t]$
12:       xml += "> x </xml>"
13:     end if
14:     included = false
15:   end if
16:   xml += insertion[$t$]
17: end for
18: return xml
```

Figure 3: **Algorithm to construct the XML frame:** The main function first detects all mismatched sequences in the string edit distance path. Mismatched sequences (function **mismatch**) trigger de-activation of aligned target words (variable *matched-target*) and an insertion record for mismatched input words (variable *insertion*). During the XML construction, matched target words are included as markup (lines 10–12 in function **construct-xml**), in addition to inserted input words (line 16, ibid.).

Acquis	Sentences	Words	W/S
100%	1395	14,559	10.4
90-99%	433	12,775	29.5
80-89%	154	5,347	34.7
70-79%	250	6,767	27.1

Product	Sentences	Words	W/S
95-99%	230	3,006	13.1
90-94%	225	2,968	13.2
85-89%	177	2,000	11.3
80-84%	185	1,950	10.5
75-79%	152	1,350	8.9
70-74%	98	987	10.1

Table 2: Composition of the test subsets based on fuzzy match scores (letter-based string edit distance).

To increase word-level matches between input and TM, we use an aggressive tokenization scheme that separates out hyphens as separate tokens and splits up letter/number combinations (e.g., *XR300* becomes *XR* $\sim$@$\sim$ *300*).

To improve word alignment quality of the Product system, we add training data from Europarl when running GIZA++. We only use the data to obtain better word alignments, we do not use it in the translation model or as TM.

When using the XML frames, we specify its use to the decoder as *exclusive*, i.e., the decoder may not override it with its own translations.

3.2 Basic Method

We compare our method using XML frames against two baselines: (a) the basic SMT system, and (b) using TM matches unmodified. The latter is only reasonable, if we find sufficiently good fuzzy matches.

We expect the methods to perform differently in different match ranges. If we find a very close match, we would expect the TM to produce a very good translation, and hopefully the XML method an even better one. If no close match exists, we expect the baseline SMT system to outperform the other methods.

This expectation is confirmed by our experiments. In Figure 4, we plot the BLEU scores for subsets of the test sets where we have a find a fuzzy match of at least 70%. Our XML method performs best for sentence which have fuzzy matches of at least 80%, SMT is best below this threshold.

3.3 Multiple Fuzzy Matches

When we described our method in Section 2, we skipped over some choices we have to make when constructing an XML frame:

1. If the fuzzy match score to the source is the same for several TM source segments, which TM match should be picked?
2. If the source has multiple targets, which target should be picked? Note that this happens typically in a parallel corpus such as the Acquis data set, less so in a real TM.
3. If we run the decoder with an XML frame what translation should be picked?

In the basic results, we answered these questions as follows:

1. randomly chosen from best matches according to letter string edit distance
2. most frequent
3. highest model score

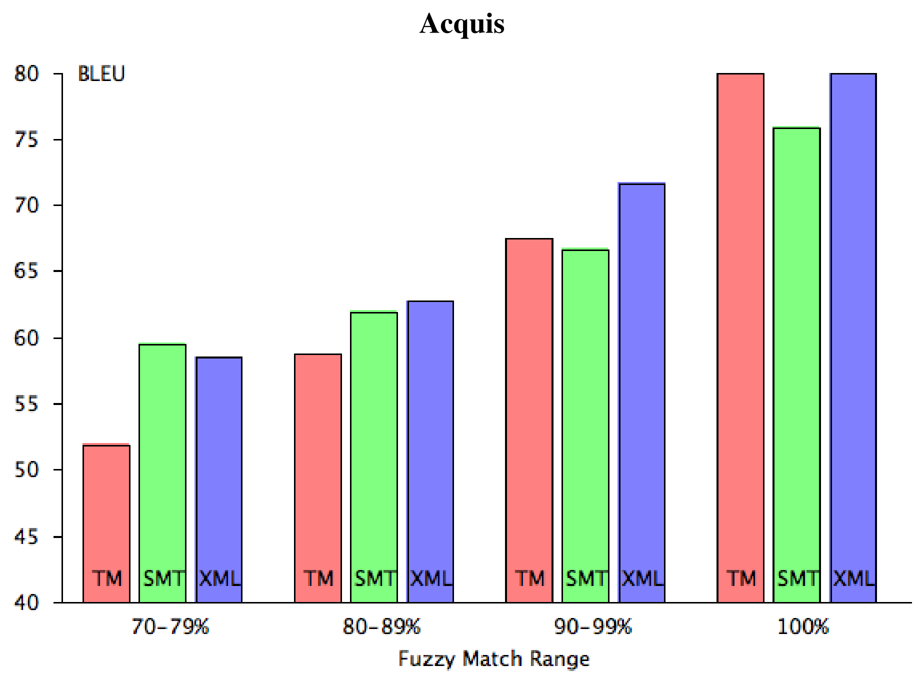
By choosing the most frequent target translation of a TM source segment, we do better than choosing one at random (for 100% matches the BLEU score is 79.9 vs. 78.7 on Acquis).

But would we also do better if we did not chose a TM match at random, but find some other criterion at a later stage to pick a translation. Maybe the SMT system offers scoring functions that would aid a decision? Experience in SMT research shows that an integrated approach with a global scoring function almost always outperforms a pipelined approach.

To keep with our three questions, we would like to now attempt the following answers:

1. all matches with best word string edit distance
2. all
3. language model score / full decoder score

Not only do we now use all source matches and all their target translations and a 10-best list of translations. We pick the best translation based on (a) the language model score and (b) the full decoder score. The full decoder score also contains the word count



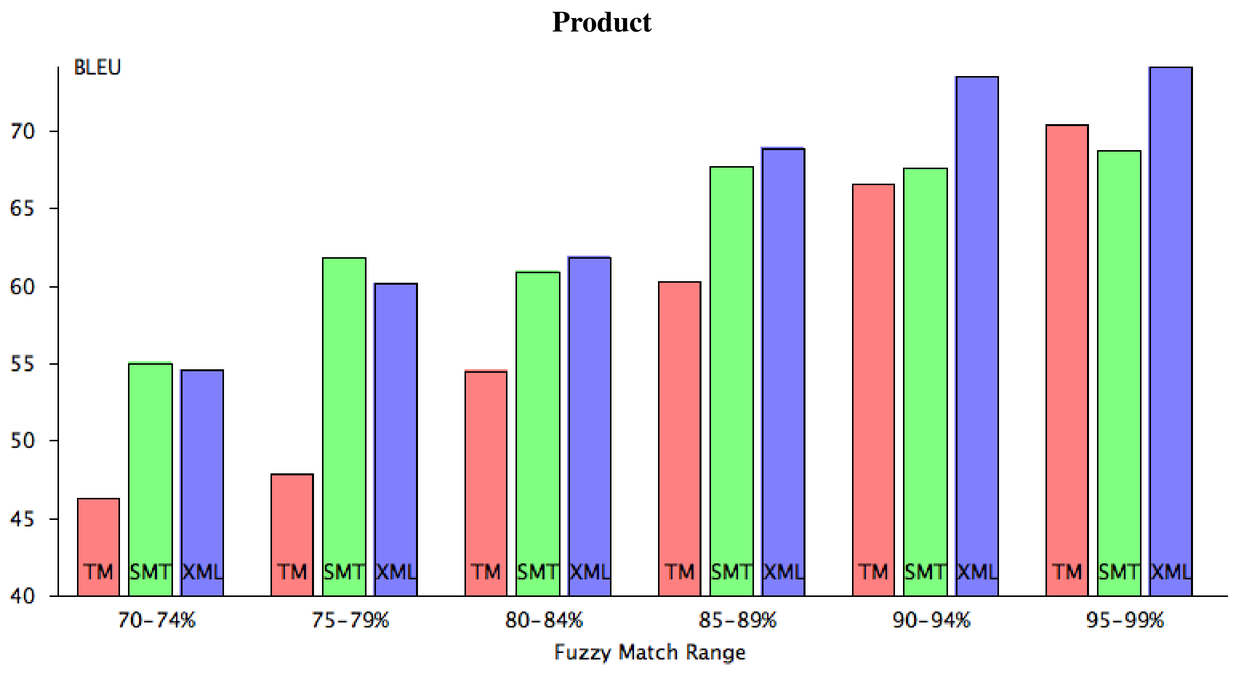


Figure 4: **Basic Results:** BLEU scores for different fuzzy match ranges. Our XML method performs best for sentence which have fuzzy matches of at least 80%, SMT is best below this threshold.

Model	70s	80s	90s	100
SMT	59.5	62.0	66.6	75.8
XML baseline	58.5	62.7	71.6	**79.9**
XML + LM score	59.0	**63.6**	71.2	77.7
XML + SMT score	**59.6**	63.3	**72.1**	66.7

Table 3: Using SMT scores to decide among multiple XML frames and translations

feature. Also, if there are different XML frames, different words are translated, hence we have different translation model scores.

Table 3 shows the results with these variations. Except for the 100% matches, one of the multiple XML frame methods outperforms the basic XML method and the SMT system, but the differences are not very large.

The decoder could also use additional scoring functions in its model, such as the fuzzy match scores. The XML frames could also be scored with methods used in the SMT system, such as lexical smoothing. But before we go further down the road, let us shortly reflect into which direction we are heading.

4 Fuzzy Matches as Very Large Rules

Readers familiar with the hierarchical model in SMT (Chiang, 2007) will notice the similarities between our XML frame construction and the construction of hierarchical phrase rules. Let us quickly review this model.

4.1 Hierarchical Phrase Rules

If we have the following sentence pair with monotone word alignment

- *the big fish*
- *les gros poissons*

we can create phrase-based translation rules from the following phrase pairs:

- (*the* ; *les*)
- (*the big* ; *les gros*)
- (*the big fish* ; *les gros poissons*)
- (*big* ; *gros*)
- (*big fish* ; *gros poissons*)
- (*fish* ; *poissons*)

Hierarchical phrase-based rules are constructed by removing a smaller phrase pairs from a larger phrase pairs. For instance, by taking the phrase pair:

(*the big fish* ; *les gros poissons*)

and removing the phrase pair

(*big* ; *gros*)

we create the rule

(*the* X *fish* ; *les* X *poissons*)

The symbol X is called a non-terminal, since the translation rule is viewed as a synchronous context-free grammar rule. In essence it is a place-holder for recursively nested sub-phrases.

Hierarchical rules require a different decoding algorithm that is typically drawn from syntactic parsing methods, but otherwise use a very similar training, tuning, and testing pipeline as traditional phrase-based models (Hoang et al., 2009).

4.2 TM Matches as Very Large Rules

How does that relate to our XML frames? Recall the XML frame that we constructed in Section 2.1:

<À l' article> 21 <, le texte du deuxiéme alinéa est supprimé .>

Instead of replacing the source sentence

The second paragraph of Article 21 is deleted .

with the XML frame, we can rewrite this frame as a hierarchical phrase rule and provide it to a hierarchical decoder:

(*The second paragraph of Article* X *is deleted .* ; *À l' article* X *, le texte du deuxiéme alinéa est supprimé .*)

In practice, hierarchical models do not use such large rules (and keep in mind, this particular rule is drawn from a relatively short sentence, thus containing few words and only one non-terminal). But this is purely due to scaling issues and concerns about the size of the rule table.

The issues can be resolved. In fact, Lopez (2007) presented a method to compute very large translation rules on the fly for hierarchical models. While these rules were limited to two non-terminals, they could contain any number of words — a very similar situation to our XML frames.

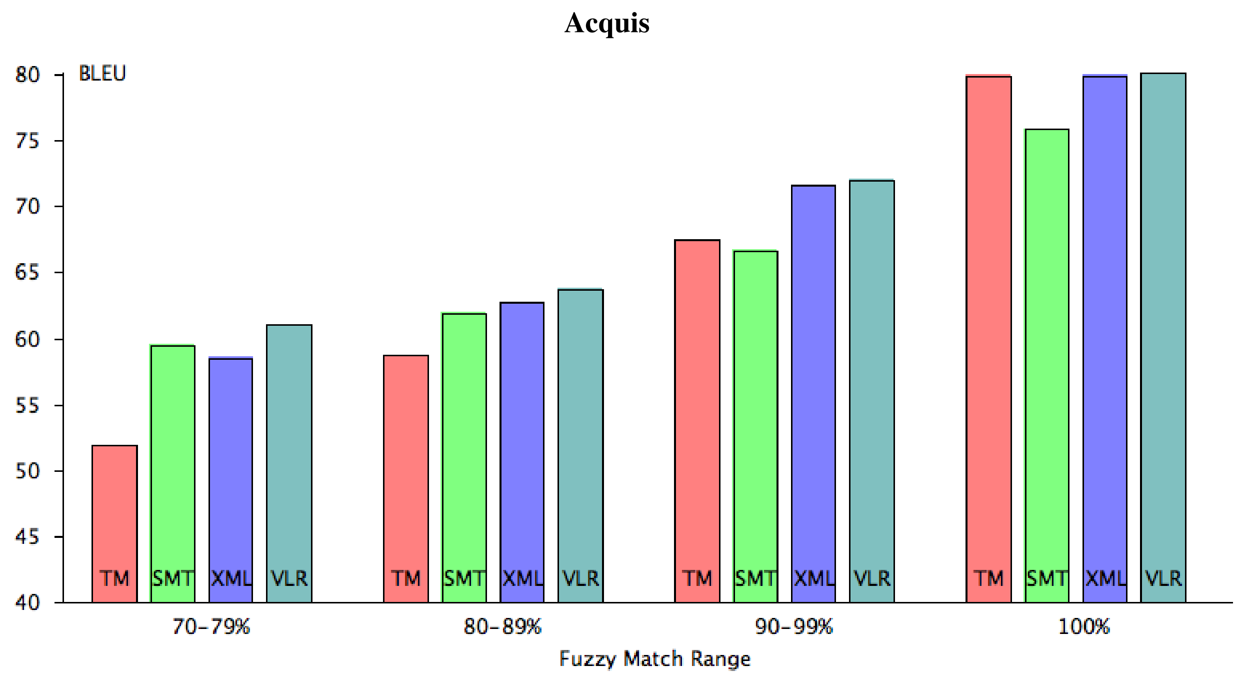


Figure 5: **TM matches as very large rules (VLR):** Encoding TM match frames as very large hierarchical grammar rules (VLR) outperforms all previous methods.

4.3 Results

We train a hierarchical phrase-based model with Moses, which has very similar performance as the phrase-based model used in the previous experiments. Instead of using XML frames, we create very large rules (VLR) that we store in a second translation table. Thus, the VLR rule table has its own feature functions. We also do this for the tuning data, hence optimizing the weighting between the two rule tables.

This approach uses

1. all TM matches with best word string edit distance
2. all target translations of each TM match
3. integrated scoring of VLR rules and basic translation rules

In other words, this is a fully integrated approach that uses all available choices at every stage.

Results are presented in Figure 5. The VLR approach outperms all other methods on all fuzzy match ranges. It even outperforms the SMT system on the 70-79% match range, where our XML method scores worse.

5 User Interaction

The translations generated by our methods may be presented to a human translator the same way currently TM matches are presented. However, we may decorate them additionally for added benefit. Since we know which parts of the translation were generated by the SMT system, we can highlight it to alert the translator to potential flaws.

We did not carry out a user study to demonstrate how much productivity increase can be expected along with the improved translation performance that we measured with the BLEU score. However, results suggest that these improvements correspond to a 5–10% increase in fuzzy match score.

On the Product and Acquis corpus, the BLEU score for our method on each of the 70-75% and 75-80% subsets is as good as a the TM score for the ranges with 10% higher fuzzy match. For subsets with higher fuzzy match scores, results are almost as good. We suspect that the BLEU scores actually underestimate these improvements, since they tend to be too literal and do not sufficiently appreciate alternate acceptable translations.

We would like to evaluate these user issues in forthcoming work.

6 Conclusion

We presented two methods that convergence ideas from TM and SMT. The first method constructed XML frames that provided specified translations for part of the segments (the matched part) and rely in the SMT decoder to fill in the remainder (the unmatched part). We showed experimentally on two data sets that we outperform those technologies for relatively high fuzzy match ranges (80-99%). Either XML or SMT always outperform TM matches.

The second method encodes these XML frames as very large hierarchical phrase rules, and uses these in a secondary rule table of a hierarchical phrase-based model. This method outperforms SMT on all tested fuzzy match ranges ($\geq$70%).

These results suggest that fuzzy TM matches can be seen as just another way to generate translation rules for an SMT system, and that there is little value is using TM without SMT.

We would like to evaluate the utility of our methods in user studies in future work.

Acknowledgement

This work was partly supported by the EuroMatrixPlus project funded by the European Commission (7th Framework Programme).

References

Biçici, Ergun and Marc Dymetman. 2008. Dynamic Translation Memory: Using Statistical Machine Translation to Improve Translation Memory. In Gelbukh, Alexander F., editor, *Proceedings of the 9th Internation Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, volume 4919 of *Lecture Notes in Computer Science*, pages 454–465. Springer Verlag.

Brown, Peter F., Stephen A. Della-Pietra, Vincent J. Della-Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation. *Computational Linguistics*, 19(2):263–313.

Chiang, David. 2007. Hierarchical Phrase-Based Translation. *Computational Linguistics*, 33(2):201–228.

Dugast, Loïc, Jean Senellart, and Philipp Koehn. 2007. Statistical Post-Editing on SYSTRAN's Rule-Based Translation System. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 220–223, Prague, Czech Republic, June. Association for Computational Linguistics.

He, Yifan, Yanjun Ma, Andy Way, and Josef van Genabith. 2010. Integrating N-best SMT Outputs into a TM System. In *Coling 2010: Posters*, pages 374–382, Beijing, China, August. Coling 2010 Organizing Committee.

Hoang, Hieu, Philipp Koehn, and Adam Lopez. 2009. A unified framework for phrase-based, hierarchical, and syntax-based statistical machine translation. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, pages 152–159, December.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Christopher J. Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

Koehn, Philipp, Alexandra Birch, and Ralf Steinberger. 2009. 462 Machine Translation Systems for Europe. In *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)*. International Association for Machine Translation.

Koehn, Philipp. 2010. *Statistical Machine Translation*. Cambridge University Press.

Lopez, Adam. 2007. Hierarchical Phrase-Based Translation with Suffix Arrays. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 976–985.

Och, Franz Josef, Christoph Tillmann, and Hermann Ney. 1999. Improved Alignment Models for Statistical Machine Translation. In *Proceedings of the Joint Conference of Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP-VLC)*, pages 20–28.

Simard, Michel and Pierre Isabelle. 2009. Phrase-based Machine Translation in a Computer-assisted Translation Environment. In *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)*. International Association for Machine Translation.

Smith, James and Stephen Clark. 2009. EBMT for SMT: A New EBMT-SMT Hybrid. In Forcada, Mikel L. and Andy Way, editors, *Proceedings of the 3rd International Workshop on Example-Based Machine Translation*, pages 3–10.

Soricut, Radu and Abdessamad Echihabi. 2010. TrustRank: Inducing Trust in Automatic Translations via Ranking. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 612–621, Uppsala, Sweden, July. Association for Computational Linguistics.

Specia, Lucia, Marco Turqui, Zhuoran Wang, John Shawe-Taylor, and Craig Saunders. 2009. Improving the Confidence of Machine Translation Quality Estimates. In *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)*. International Association for Machine Translation.

Steinberger, Ralf, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis, and Daniel Varga. 2006. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *5th Edition of the International Conference on Language Resources and Evaluation (LREC '06)*, pages 2142–2147.

Zhechev, Ventsislav and Josef van Genabith. 2010. Seeding Statistical Machine Translation with Translation Memory Output through Tree-Based Structural Alignment. In *Proceedings of the 4th Workshop on Syntax and Structure in Statistical Translation*, pages 43–51, Beijing, China, August. Coling 2010 Organizing Committee.