EBMT for SMT: A New EBMT-SMT Hybrid

James Smith
Oxford University Computing Laboratory

Stephen Clark
University of Cambridge Computer Laboratory

Abstract

We propose a new framework for the hybridisation of Example-Based and Statistical Machine Translation (EBMT and SMT) systems. We add new functionality to Moses to allow it to work effectively with an EBMT system. Within this framework, we investigate the use of two types of EBMT system. The first uses string-based matching, and we investigate several variations, but find that the hybrid system is unable to match the pure SMT system. The second is a syntax-based EBMT system which uses dependency trees to compare inputs to the example base, and we show that this system is consistently better than the string-based approach. We find that, while the SMT system still performs better overall, the syntax-based hybrid does perform particularly well for some examples.

1 Introduction

In this paper, we investigate a new framework for combining EBMT and SMT. Our framework consists of an EBMT system which translates the parts of the sentence for which it is confident, followed by an SMT system which fills in the gaps and produces a final translation. A key component of our hybrid system is the open-source phrase-based SMT system, Moses [6]. Moses provides one feature of particular interest to us. It can take advantage of external knowledge through an XML interface which allows translations to be specified for segments of the input. Moses accepts the translations for the parts of the sentence which are XML tagged, and uses its statistical models to translate the remainder of the sentence and recombine the phrases. We build our hybrid EBMT-SMT system using this feature of Moses. Given an input, our EBMT system translates the parts for which it is confident, and then passes the marked-up sentence to Moses. Moses translates the remainder of the sentence using SMT, combines all of the translations, and produces the output.

The intuition behind this approach is based on the known strengths and weaknesses of EBMT and SMT. While EBMT is not particularly effective at providing wide coverage due to data-sparseness constraints, when a suitable example is found it can provide accurate translations. SMT is much better at covering a wide range of inputs, but is unable to directly take advantage of examples in its training data which closely resemble the input. By combining the EBMT and SMT paradigms, the EBMT system is able to exploit good examples in the training data, and then the SMT system can use its generalised models to translate the remainder. This has the additional benefit that the EBMT system does not need to be responsible for the recombination/generation stage, as this is handled by the SMT system.

The focus of our experimentation was on the matching phase. The next two sections describe the two approaches we investigated for matching: a string-based approach and a syntax-based approach. We also investigated the use of semantic information from WordNet (as part of the string-based approach), and used a filtering stage to allow examination of larger data sets.

2 String-Based Matching

Distance Metrics Our first set of experiments employ a "linguistically-light" strategy, focusing on finding matches which share exact strings with the input. The first of the matching algorithms is the **Levenshtein distance** (or edit distance) [8], for which there is a well-known dynamic programming algorithm. Given an input sentence, we want to find the most similar sentence, according to the edit distance, in the parallel corpus. If the distance between the input and the best example is sufficiently small we should be able to reuse parts of the target language side of the example to translate the input.

The second metric is a comparison of **common substrings**. The simplest approach would just count the number of matching substrings, but longer matches are of greater benefit than smaller ones. Hence, we allow each substring to contribute the square of its length to the score (i.e. a unigram contributes 1, a bigram 4, a trigram 9, and so on). Wer use a dynamic programming algorithm, similar to that for the Levenshtein distance, to efficiently calculate this similarity score.

Our final similarity metric deviates a little from the "linguistically-light" strategy by introducing some **semantic knowledge** into the Levenshtein distance algorithm, by varying the cost of substitution based on semantic similarity of words. Semantic similarity measures have been used in various forms throughout EBMT, e.g. [10], [16], [11] and [15]. We use the noun hierarchy from WordNet [4], with the following similarity measure:

$$d(w_1, w_2) = \frac{\text{distance from deepest of } w_1 \text{ and } w_2 \text{ to lowest common ancestor}}{\text{distance from deepest of } w_1 \text{ and } w_2 \text{ to root}}$$

This measure accounts for how deep in the hierarchy a noun is, and gives a value between 0 and 1 which indicates the relative depth of the lowest common ancestor in the tree. We use this value as the cost of a noun substitution in the Levenshtein distance calculation.

Filtering the Corpus EBMT relies on examples similar to the input existing in the example base in order to make a good translation. Hence we would like the example base to be as large as possible to maximise the chances of finding a good match; the difficulty is that calculating the Levenshtein distance or common substrings per input for every example becomes prohibitive as the size of the example base grows. We solve this by filtering the example base using *n*-gram counts. Every *n*-gram of length 1 to 5 is indexed, with a record of the sentences in which each *n*-gram occurs. The filter is applied as the first step of the matching process, as a fast but crude pruning method. The complete example base is filtered down to the examples most likely to be relevant (according to how many *n*-gram are shared between the input and example), and then the more time-consuming metrics are applied only to these examples.

3 Syntax-Based Matching

Our syntax-based matching requires inputs and examples to be represented as dependency trees; hence we need a source language syntax parser. We use the C&C parser [3], which produces dependency graphs made up of grammatical relations. It also has the advantage of being relatively fast, which is important for us as we need to parse a large example base. The parser is based on Combinatory Categorial Grammar (CCG) [14]. One of the key features of CCG is its ability to capture long-range dependencies, which can be exploited by the EBMT module and potentially give it an advantage over the phrase-based system. Previous work using CCG for SMT includes [5].

Modifications to Moses Moses' standard XML features do not provide all of the functionality we require for the syntax-based matching. Syntax-based matches are not required to be contiguous strings in the input sentence, as was the case in the string-based system. Hence we are more likely to need to specify several smaller phrase translations as part of a single noncontiguous match than one large contiguous one. These smaller phrases are all related, having come from the same match, so we do not want Moses to pick and choose between them; it should use either all of them or none of them. We may wish to specify several competing translations from different examples, and allow Moses to choose the most suitable. These competing translations may translate different but overlapping parts of the input.

```
<span english="homme aime lire" span="2,4"/>
the clever man likes reading books
```

Figure 1: A simple example output from the string-based EBMT system.

When Moses discards a particular phrase translation in favour of another, we need to ensure that it also discards any other translation table entries which came from the same match.

A second problem we encounter with Moses arises from one-to-many word alignments. As an example, consider an aligned sentence pair in which an English token *not* is aligned to the French tokens *ne* and *pas*, which are separated in the French sentence. If this alignment is used to translate *not*, we have a dilemma in terms of which word to substitute for it in the translation. Since Moses' translation table (and by extension the XML interface) only allow us to specify a direct phrase-for-phrase translation, we must either discard one of *ne* and *pas*, or specify the translation as *ne pas*.

To solve these problems, we added new functionality to Moses, by introducing a new XML tag, `<linked>`, which specifies a connection between the phrases within it. Any phrase translations grouped within a `<linked>` tag must either all be used, or none of them used. This solves the first problem mentioned above. Further, within a `<linked>` tag, we allow multiple tags with the same span. These will all be used if the `<linked>` group is used.

Matching Our syntax-based EBMT system works by parsing both the input and the source language examples into dependency trees, using the C&C parser. We find the best dependency tree matches for the input in the example base. Various syntax matching structures have been used in other SMT or EBMT systems [9, 7, 13]. We define the best match as the biggest connected dependency sub-tree in the example base which exactly matches a dependency sub-tree in the input, where an exact match requires matching words and dependency types. The matching process begins by comparing all of the dependencies in the input to those in the example. If there are any matches, we seed the matching sub-tree with one of the dependency matches, and then grow the tree on both sides of the match as much as possible, until we have found the maximal matching sub-tree. We repeat this process for every example[1] and return the biggest matches.

Similar to the constraints used in the string-based system, we can specify a minimum threshold for the size of a match. In the syntax-based system there are two ways of measuring this: the number of dependency matches, and the number of words which must be matched contiguously (as part of a larger dependency tree match).

4 Transfer to Target Language

Having found matching examples, the next step is to translate the input. Our aim here is not to translate the entire input sentence, but rather just the parts for which we have good matches. The SMT system will handle the remainder of the input, and deal with any boundary friction issues. In this phase we make use of word alignments generated by GIZA++ [12], which are also used by the SMT system.

The transfer modules for the string-based and syntax-based systems are similar. The matches provided by the syntax-based system are in tree form, rather than a flat sequence. We flatten them into tuples

[1]In practice, we filter the example base and only attempt matching on sentences which share at least one dependency match with the input.

```
<linked>
  <span1 english="homme intelligent" span="1,2"/>
  <span2 english="grands livres" span="6,7"/>
</linked>
<linked>
  <span3 english="aime lire" span="3,4"/>
  <span4 english="grands livres" span="6,7"/>
</linked>

the clever man likes reading really big books
```

Figure 2: An example using the <linked> tag to specify overlapping translation candidates.

containing an input word and the source word to which it was matched in the tree. This list of word pairs is then sorted based on the original string position of the input word. We then process the list in order, extracting contiguous input phrases matched by the example. This process is continued until all of the matched input phrases have been collected.

As an example, consider the following sentences:

(1) Input: the clever man likes reading books
Source: the man likes reading big books

First, we generate a list of pairs of word matches between the input and the source side of the example ((i, j) indicates a match between word a_i from the input and word b_j from the source). This list is sorted:

(2) {(0,0), (2,1), (3,2), (4,3), (5,5)}

Next, we extract all contiguous sequences of matches. From the matched pairs in (2) we get the following sequences:

	{(2,1), (3,2), (4,3)}	man likes reading
(3)	{(0,0)}	the
	{(5,5)}	books

We now need to look at both sides of the example to perform the translation. For this example, we assume that the source and target sides of the example are aligned. For each matched sequence in (3), we collect all of the aligned tokens in the target language. The tokens are subsequently ordered based on their original ordering in the example, and then contiguous sequences are extracted from these. In this example, for the source sequence *man likes reading*, we will extract *homme aime lire* as the only target sequence.[2] The final step is to format the target phrases for presentation to the SMT system, using Moses' XML interface. As small matches are generally not well translated by the EBMT system, we set a threshold on the minimum size of match to be considered. Using a threshold of 3, the above example will result in the output from the EBMT system shown in Figure 1. The `english` attribute represents the translated phrase (following the convention of denoting the target using *English*). The `span` attribute specifies the word range in the source language to which the translation applies.

Each contiguous matched phrase in this list will have its translation represented by a Moses XML tag in the output. For each contiguous phrase, we look up the target language words to which it is aligned.

[2]In some cases, a single source phrase may result in multiple target phrases. Handling this situation required significant modifications to the SMT system, and is discussed in §3.

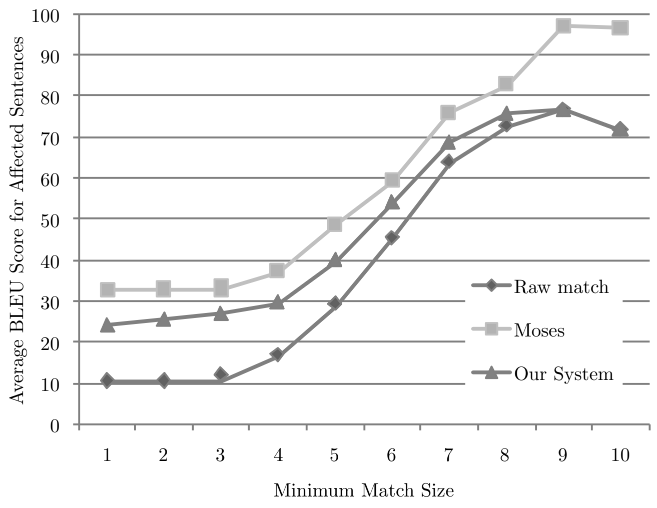

Figure 3: Performance of the Levenshtein string-based system, Moses, and a baseline on EuroParl data.

These target words are sorted by their original sentence position, and then processed to extract contiguous target phrases. If the target words for an input phrase do not form a single contiguous match, then one tag will be produced for each target phrase, with all of them referencing the same input word range as their translated span. When all of the matched input phrases have been translated, the entire collection of target phrases are contained within a <`linked`> tag. This process is repeated for each of the match trees, producing a series of <`linked`> tag groups, one for each tree. The resulting string is then passed to the SMT system. An example of <`linked`> usage is shown in Figure 2.

5 Results

All of our results are for translating from English into French, using Europarl data. The accuracy measure is the BLEU score.

String-Based System We compared the performance of the various distance metrics presented in §2, starting with the Levenshtein metric and the common substrings metric. The Levenshtein distance consistently and significantly outperforms the common substrings metric, despite the fact that the common substrings algorithm favours contiguous matches more heavily than the Levenshtein distance. The common substrings approach performs little better than using the filter (as described in §2) alone. This is due to its definition of sentence similarity, which has a lot in common with the filter. In contrast, the Levenshtein distance takes a different approach to the filter, and the combination of the two is effective.

The semantic similarity module did tend to have a minor positive effect on the translation score, suggesting that in some limited cases it was selecting more relevant examples. The improvement was small though, and did not justify the additional processing cost such a semantic similarity module incurs.

We compared our string-based hybrid system to two other systems. The first was a low baseline, in which the translation for each input is taken to be the target language half of the best matching example. The second was the Moses SMT system. The results are shown in Figure 3. The Levenshtein string-based system consistently outperforms the baseline, but does not score as well as Moses. Moses is not as good at ordering the EBMT-translated phrases correctly as it is with those it produces itself. This can sometimes result in phrases being out of place in the output. Further, Moses is less affected by bad word

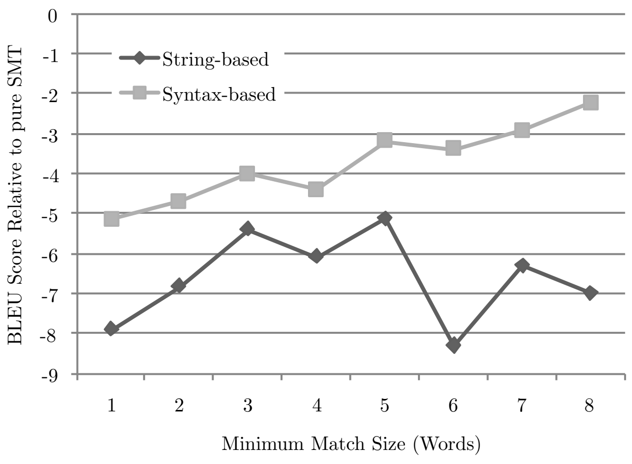

Figure 4: Performance of the string-based and syntax-based systems relative to that of the pure SMT system on EuroParl data.

alignments than our system. The alignments between the source and target languages are not perfect, and where Moses is likely to pick the most commonly occurring alignment over the whole corpus, the EBMT component has to rely on the alignment specific to the chosen example.

There is also a fair amount of overlap between the techniques used by a string-based EBMT system and a phrase-based SMT system. While the EBMT system is better able to take advantage of long matches, and always uses matches from the same example (reducing issues of boundary friction), the SMT paradigm is better overall at translating contiguous strings thanks to its policy of selecting the most common translation in each case.

Syntax-Based System Disappointingly, our syntax-based system does not manage to achieve the same performance measured in BLEU points as the SMT system. Across all of the testing data, our system generally fell about 3-4 points short of Moses. More encouragingly though, the syntax-based system consistently outperforms the string-based hybrid system. Figure 4 shows a chart of the BLEU scores (as percentages) of both the syntax-based and string-based systems relative to Moses. The syntax-based system has a minimum dependency match threshold of 3 dependencies, and moving towards the right of the graph the minimum word match threshold is increased in both systems, showing the effect that the threshold has on the hybrid system.

The dependency threshold determines the minimum size that a sub-tree match must satisfy in order to be used in a translation. Figure 5 shows the results of three sets of experiments, with an increasing dependency threshold from left to right. The three data sets represent fixed word thresholds of one, three, and five words. Performance for each system is reported by BLEU score (as a percentage) measured relative to the SMT system. All three systems show relative stability up to about three dependencies, after which we see a general performance decline relative to the SMT system. There are two reasons for this. As the dependency threshold increases, we begin to restrict the system to operation only on sentences for which large matches can be found. A large dependency match is likely to imply a large contiguous string match, which is where the SMT system performs well. To compound the problem, the move to higher dependency thresholds introduces data sparsity issues into our system. This results in the EBMT component providing far fewer options to Moses, and relying too heavily on the quality of one or two matches.

The word threshold determines the minimum number of contiguous words which must be present

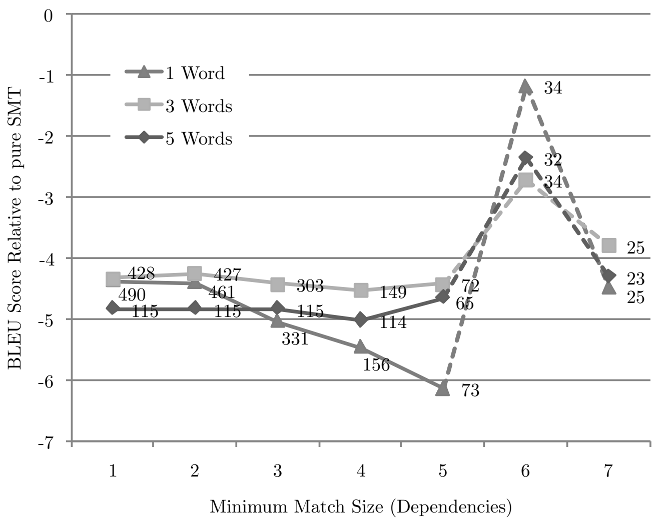

Figure 5: The effect of changing the minimum dependency match threshold in the syntax-based system. Data labels show the number of sentences out of a sample set of 500 for which the EBMT component was able to provide part of the translation. The final two data points are indicative of the instability caused by data sparsity with high threshold settings.

as part of the matched sub-tree in order for it to be used for translation. Performance peaked at word thresholds of between three and five. At lower thresholds, the EBMT module allowed smaller (less reliable) matches to be included in the output. For mid-range word thresholds, the threshold is large enough to cut out small phrase translations which are better left to the SMT system, but still allows the syntax-based matching to be effective. With high thresholds, we are restricting ourselves to operation on sentences to which SMT is well suited, and where the syntactic information we utilise adds little useful information.

6 Discussion

In terms of overall BLEU scores, the SMT system outperforms our hybrid. BLEU, however, is not without its flaws. The problems with BLEU and other automatic evaluation methods have been discussed many times, e.g. [1] and [2]. Both point out simple ways of manipulating the metric in order to produce better BLEU scores without improving translation quality. For example, BLEU does not make any requirements about the ordering of phrases, nor does it require that the matched *n*-grams all come from the same reference translation. [1] also shows results that cast doubt on the claim that BLEU judgements mirror human ones, giving examples from MT competitions in which the human ranking of systems is very different to BLEU's ranking. Further, the reliance on reference translations is fundamentally flawed in that there are generally many more ways of saying something than can be captured in a few references, so the system could produce a perfect translation which receives a low score because it was worded differently than the references.

The EBMT module is good at finding matches, and can often provide a range of choices to the SMT system. Unfortunately, the word alignments for these matches are often imprecise and can lead to incorrect translations. The pure SMT system solves this problem by using statistical models based on counts across the entire training corpus to select the most likely candidates. In contrast, the EBMT system is reliant on the examples it selects being well-aligned. To compound the problem, the EBMT system also relies on the accuracy of the CCG parser. While the parser's performance is very good in general, if a mistake is made on a specific example that we happen to use for translation, the EBMT module's performance is likely to suffer. The provision of multiple candidates to the SMT module does a little to assuage these effects, but the real problem lies in the EBMT system's inability to differentiate between good and bad candidates. Ultimately, this becomes the Achilles' heel for the framework we have proposed and used as the basis for our two hybrid systems. Without tighter integration of the statistical models into the EBMT procedure, the system is unable to perform at the same level as the SMT system.

In conclusion, the hybrid system was unable to consistently match the performance of the pure SMT system. However, we did find many examples in which the hybrid system outperformed the SMT system, and the approach is worthy of further investigation. The interface we have developed for integration with Moses is very flexible, and could be used to enable other MT systems, example-based or otherwise, to operate within the same framework.

References

[1] Chris Callison-Burch, Miles Osborne, and Philipp Koehn. Re-evaluating the Role of BLEU in Machine Translation Research. In *Proceedings of the EACL*, pages 249–256, 2006.

[2] David Chiang, Steve DeNeefe, Yee Seng Chan, and Hwee Tou Ng. Decomposability of Translation Metrics for Improved Evaluation and Efficient Algorithms. In *Proceedings of EMNLP*, pages 610–619, 2008.

[3] Stephen Clark and James R. Curran. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4):493–552, 2007.

[4] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication).* The MIT Press, 1998.

[5] Hany Hassan, Khalil Sima'an, and Andy Way. A syntactified direct translation model with linear-time decoding. In *Proceedings of EMNLP*, pages 1182–1191, Singapore, 2009.

[6] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the ACL Demos*, 2007.

[7] Philippe Langlais and Fabrizio Gotti. EBMT by tree-phrasing. *Machine Translation*, 20(1):1–23, 2006.

[8] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[9] Zhanyi Liu, Haifeng Wang, and Hua Wu. Example-based machine translation based on tree-string correspondence and statistical generation. *Machine Translation*, 20(1):25–41, 2006.

[10] Makoto Nagao. A framework of a mechanical translation between Japanese and English by analogy principle. In *Proceedings of the International NATO Symposium on Artificial and Human Intelligence*, pages 173–180, Lyon, France, 1984.

[11] Sergei Nirenburg, Constantine Domashnev, and Dean J. Grannes. Two Approaches to Matching in Example-Based Machine Translation. In *Proceedings of the 5th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 47–57, 1993.

[12] Franz Josef Och and Hermann Ney. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, 2003.

[13] Christopher Quirk and Arul Menezes. Dependency treelet translation: the convergence of statistical and example-based machine-translation? *Machine Translation*, 20(1):43–65, 2006.

[14] Mark Steedman. *The Syntactic Process.* MIT Press, Cambridge, MA, 2000.

[15] Eiichiro Sumita. Example-based machine translation using DP-matching between word sequences. In *Proceedings of Workshop on Data-driven Methods in Machine Translation*, pages 1–8, Toulouse, France, 2001.

[16] Eiichiro Sumita, Hitoshi Iida, and Hideo Kohyama. Translating with Examples: a new approach to machine translation. In *Proceedings of The 3rd International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language*, pages 203–212, Austin, Texas, 1990.