Using a large monolingual corpus to improve translation accuracy

Radu Soricut, Kevin Knight, and Daniel Marcu
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
{radu, knight, marcu}@isi.edu

Abstract. The existence of a phrase in a large monolingual corpus is very useful information, and so is its frequency. We introduce an alternative approach to automatic translation of phrases/sentences that operationalizes this observation. We use a statistical machine translation system to produce alternative translations and a large monolingual corpus to (re)rank these translations. Our results show that this combination yields better translations, especially when translating out-of-domain phrases/sentences. Our approach can be also used to automatically construct parallel corpora from monolingual resources.

1 Introduction

Corpus-based approaches to machine translation usually begin with a bilingual training corpus. One approach is to extract from the corpus generalized statistical knowledge that can be applied to new, unseen test sentences. A different approach is to simply memorize the bilingual corpus. This is called *translation memory* [1], and it provides excellent translation quality in the case of a "hit" (i.e., a test sentence to be translated has actually been observed before in the memorized corpus). However, it provides no output in the more frequent case of a "miss".

While it is often unlikely that a test sentence will be found in a limited bilingual training corpus, it is much more likely that its *translation* will be found in a vast monolingual corpus. For example, note that the English sentence "She made quite a good job." does not appear on Google's Web index. But "She did quite a good job." does appear. If both sentences are suggested to us as translations, we can therefore automatically prefer the latter simply because this string has been observed before in a large monolingual corpus of English text. Similarly, on the basis of the frequency of a phrase, we may prefer as translation of the French phrase "elle a beaucoup de cran" the English phrase "she has a lot of guts" as opposed to, say, "it has a lot of guts", even if both phrases are found in a large monolingual corpus (Altavista's Web index)[1]. Their frequencies, however, differ by a factor of seven, and we can prefer the former translation on

[1] "The road from Angkor Wat ... it has a lot of guts to call itself a road."

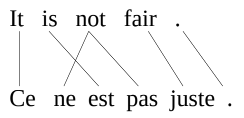

Fig. 1. Sample word alignment.

that basis. A similar idea has been proposed by Grefenstette [2] as an approach to lexical choice for machine translation. We take this idea a step further and propose the use of a vast monolingual corpus to validate full translations.

In this paper we introduce an alternative approach to machine translation that operationalizes the intuitions above. Our method uses a statistical machine translation system to produce alternative translations and a large monolingual corpus to (re)rank these alternative translations. We contribute to the field of machine translation in two ways:

- We introduce algorithms capable of counting the number of occurrences of a large number of translations (about 10^{300}) in a large monolingual sequence/corpus (about 1 billion words of English text).
- We show empirically that the accuracy of a statistical MT system can be improved if translated phrases/sentences are re-ranked according to their likelihood of occurring in a large monolingual corpus.

The algorithms we introduce can be also used to automatically construct parallel corpora starting from a translation table and a large monolingual corpus.

2 IBM Model 4

In this paper we use the IBM Model 4 [3]. For our purposes, the important feature of this model is that, for any given input French sentence, we can compute a large list of potential English translations (of order 10^{300} or even larger, see Section 3.2). IBM Model 4 revolves around the notion of word alignment over a pair of sentences (see Figure 1). A word alignment assigns a single English string position to each French word.

The word alignment in Figure 1 is shorthand for a hypothetical stochastic process by which an English string gets converted into a French string. There are several steps to be made. First, every English word is assigned a fertility. We delete from the string any word with fertility zero, we duplicate any word with fertility two, etc. Second, after each English word in the new string, we may insert an invisible NULL element with probability p_1 (typically about 0.02). The NULL element ultimately produces "spurious" French words. Next, we perform a word-for-word replacement of English words (including NULL) by French words, according to certain *translation probabilities* $t(f_j \mid e_i)$ (which together form a *translation table*, or T-table). Finally, the French words are permuted according to certain *distortion probabilities*.

3 Multiple string matching against large sequences

From a computer science perspective, the problem we are trying to solve is simple: we are interested in determining the number of occurrences of a set of strings/translations $\{t_1, t_2, \ldots, t_n\}$ in a large sequence/corpus S. When n and S are small, this is a trivial problem, and tools such as *grep* can easily solve the problem. Unfortunately, for large n, the problem becomes extremely challenging.

3.1 Naive approaches

Simple *grep*

We ignore for the moment that we need to search for n strings. Even if one tries to search for all the occurrences of *one* string t_i in a corpus of 1 billion-words, *grep* will take about 30 minutes. Unfortunately, we do not have to search for 1 string, but for about 10^{300}. Searching sequentially using *grep* is clearly infeasible.

Regular expressions and *egrep*

Another idea is to represent all the possible strings/translations in a regular expression. Given that these translations share a lot of information, one can expect that the resulting regular expression will be much more compact than a sequential enumeration of all possible translations. We developed an algorithm to compactly represent all the possible translations as a regular expression. For a 6-word French sentence, the regular expression that subsumes all its possible translations into English takes roughly 4 Mbytes. Such huge expressions cannot be processed by *egrep*.

Querying the Web

Another attractive solution seems to be that of querying directly the Web. After all, the Web can provide us access to the ultimate monolingual corpus; and search engines like Google answer queries in only a few mili-seconds. Unfortunately, one cannot do searches of large regular expressions on the Web, and therefore even if one search takes 1 ms, 10^{300} searches take an infeasible amount of time.

3.2 Multiple string matching using FSAs

In order to solve the multiple string matching problem, we decided to expand on a solution proposed initially by Knight and Al-Onaizan [4], which proposes a finite state acceptor (FSA) to compactly represent possible English translations of a French sentence. If the IBM Model 4 is used as translation model, then such an FSA has to account for all the stochastic steps described in Section 2. Fortunately, one can build small acceptors (FSAs) and transducers (FSTs) that account for each of these steps separately, and then compose them together to obtain an acceptor which accounts for all of them.

Representing multiple translations as FSAs

In the framework of IBM Model 4 we start with an English string and perform several steps to probabilistically arrive at a French string. When translating/decoding, we need to perform all the steps described in Section 2 in reverse order to obtain the English strings that may have produced the French sentence.

Assume that we are interested in representing compactly all English translations of the French phrase "un bon choix". Since French and English have different word orders, we first need to generate all possible permutations of the French words. An FSA that accomplishes this task is presented in Figure 2(b).

The mapping between French and English words is often ambiguous. When translating from French into English, we can translate "un" as "a", "an", or even as NULL. We can build an FST to take into account the multiple translation possibilities. Given that we actually build probabilistic transducers, the probabilities associated with these possibilities can be incorporated. The T-table (Section 2) can be used to build a simple transducer: it has only one state and has one transition for each entry in the T-table (a highly simplified version is shown in Figure 2(a)). Composing this FST with the previous FSA results in an FSA modeling both the different word order and the word translation ambiguity phenomena (Figure 2(c)).

The story gets more complicated as one has to add new transducers for the other steps discussed in Section 2. For example, our French phrase "un bon choix" can be translated as "good choice" in English. Our model accomplishes this by considering the word "un" to be the translation of a NULL English word. A simple two-state automaton is used to model the NULL word insertions (see Figure 2(d)).

Finally, fertility also needs to be modeled by an FSA. In Figure 1, for example, the English word "not" is mapped into both "ne" and "pas". This can be simulated by using the fertility 2 of "not" to first multiply it (i.e., create "not not" on the English side), and then translating the first one as "ne" and the second one as "pas". A simple FSA (not shown here) can be used to model word fertilities.

The step-wise composition of these automata is shown in Figure 2. (Only very few possible translations are illustrated, and the probabilities are not included in the picture for readability.) For a given French sentence f, the final result of these operations is a non-deterministic FSA with epsilon transitions, which we generically call FSA_f^0. For a 6-word French sentence f such as "elle me a beaucoup appris .", the finite state acceptor we automatically generate has 464 states, 42139 arcs, and takes 1,172 Kbytes. The total number of paths (without cycles) is 10^{328}. There are a number of advantages to this representation:

- FSA_f^0 enumerates all possible English translations of f (according to the translation model).
- FSA_f^0 reflects the goodness of each translation e_i as assessed by the statistical model used to generate it; as Knight and Al-Onaizan [4] have shown, the probability of a path e through the FSA corresponds to the IBM-style transition model probability $P(f|e)$ [3].

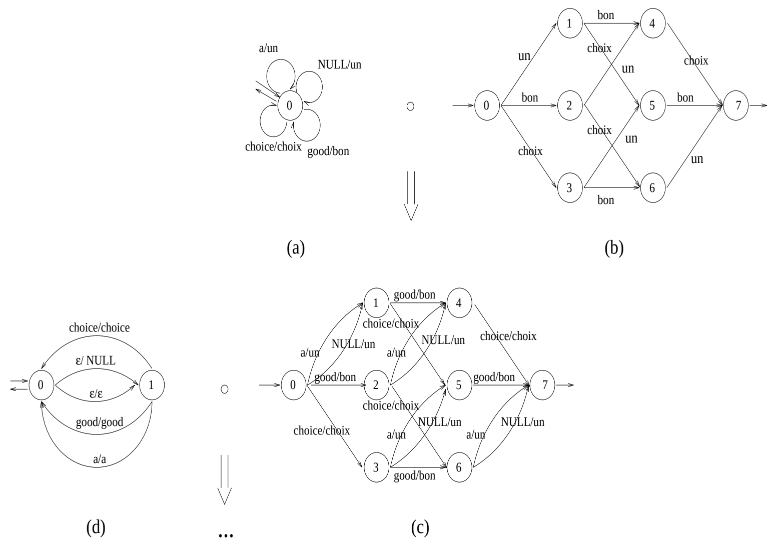

Fig. 2. Step-wise composition of FSAs/FSTs.

– FSA_f^0 can be used as a binary classifier for English strings/translations ("yes" if string e is a possible translation of f; "no" otherwise).

A finite state machine built in this manner operates as a rudimentary statistical machine translation system. Given a French sentence f, it can output all its English translations e_i and their IBM-4 translation probabilities (modulo distortion probabilities).

Matching FSAs against large sequences

In the previous section, we have shown how to automatically build, for a given French sentence f, a finite state acceptor FSA_f^0 that encodes all possible English translations of f. The next step is to use FSA_f^0 to find all the occurrences of the possible English translations of f in a large monolingual corpus. In order to be able to perform the string matching operations, we need to modify the monolingual corpus such that all the English words unknown to FSA_f^0 are replaced by UNK in the monolingual corpus. The acceptor FSA_f^0 needs also to be slightly modified to account for the UNK token, and we call the resulted acceptor FSA_f^1. We do not describe these modifications here due to lack of space.

A summary of all the operations is presented in Figure 3. From a French sentence f, using the parameters of a statistical translation model, a finite state acceptor FSA_f^0 is built. FSA_f^0 is further modified to yield FSA_f^1. A large English corpus is taken sentence by sentence and modified such that all English words not known by FSA_f^0 are replaced by UNK. Each modified sentence is matched against FSA_f^1, and for each sentence accepted by FSA_f^1 we store the string matched, and

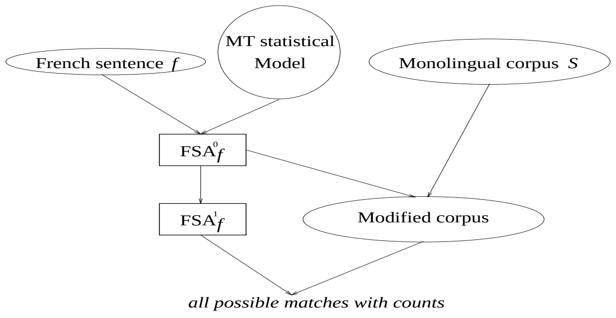

Fig. 3. Multiple string matching against a large corpus

also keep a count of each appearance. We end up with all possible translations of f that also occur in the corpus S, and their counts. The number of observed translations of f decreases from an order of magnitude of 10^{300} as proposed by the translation model to an order of magnitude of 10^3-10^6.

All these operations can be performed using off-the-shelf software. We use a publicly available finite-state package[2] to perform the composition operations which yield FSA^0_f. We use the same package for the string matching problem: FSA^1_f is loaded into memory and matched sentence by sentence against the corpus, performing the acceptance test.

4 Time performance issues

The multi-string matching method described in Section 3 is capable of finding all translations of a given French sentence in a large monolingual, English corpus, but it needs to be performed in parallel in order to be usable. A parallel version of our algorithm, run on a Linux cluster of 192 nodes and 384 CPUs (Pentium 3, 766MHz & 866MHz), obtains linear speed-up in run-time with the increase in the number of processors. For a 6-word French sentence we obtain the following reductions in time as we increase the number of processors:

1 Processor:	30 hrs.	(1800 min.)
10 Processors:	3 hrs.	(180 min.)
100 Processors:	0.3 hrs.	(18 min.)

The parallelization method is straightforward: given the total number of sentences in the corpus, one has to assign an equal number of sentences to each processor, run in parallel the algorithm on each such corpus, and then collect and sum up the results.

[2] http://www.isi.edu/licensed-sw/carmel/index.html

5 Performance evaluation

In order to asses whether our translation method can improve the performance of IBM Model 4 we collected a corpus of 1.6 billion words of English by collecting various corpora made available by LDC[3]. The parameters of the statistical model were trained on 500,000 parallel sentences from the Hansard genre using GIZA[4]. We translated 101 in-domain French sentences (of length 6) taken from the Hansard genre, and 110 out-of-domain French sentences (of length 6) taken from Le Monde.

We translated these sentences using four different methods: two of them were previously published methods using only the parameters of the statistical model and different decoding algorithms – Stack and Greedy [5] – and were publicly available for downloading[5]. The other two methods were variations of the method using a Big monolingual Corpus (therefore called BC here): a pure BC method, and a semi-automated method BC^+ where we looked at the first 10 translations proposed by the BC method for each sentence and selected by hand the best one.

5.1 The BC method

For a given French sentence suppose there is a possible translation occurring in the monolingual corpus. For each such translation we store:

- The translation model probability which ignores distortions (TM^-)
- The number of occurrences in the monolingual corpus (NO)
- The alignment for each translation

The formula $NO \times TM^-$ is used for a pre-ranking from which the first 500 candidates are extracted. On each of these candidates we compute a language model probability (LM) using trigrams and also (using the alignment) a translation model probability which includes distortions (TM).

The last step is to re-rank the top 500 candidates using a formula $F(NO, LM, TM)$. We currently re-rank according to the formula $NO \times LM \times TM$. The formula F can be trained to yield optimal results, which we leave for future work.

5.2 Evaluation

Table 1 shows the number of perfect translations obtained on both corpora by each of the methods used. Here, “perfect” refers to a human-judged translation that transmits all of the meaning of the source sentence using flawless target-language syntax. The table also shows the number of untranslated/unscored sentences for each corpus.

The BC method, which we introduced in this paper, produces 43 perfect translations out of the 101 sentences of the in-domain corpus and returns zero

[3] http://www.ldc.upenn.edu/
[4] http://www.clsp.jhu.edu/ws99/projects/mt/
[5] http://www.isi.edu/natural-language/projects/rewrite/

	In-domain		Out-of-domain	
	perfect	unsc.	perfect	unsc.
Stack	40	0	16	0
Greedy	53	0	18	0
BC	43	7	24	33
BC$^+$	62	7	32	33

Table 1. Translation scores obtained by different methods.

	In-domain		Out-of-domain	
	G perf.	G err.	G perf.	G err.
BC perf.	31	12	10	14
BC err.	22	29	8	45

Table 2. Confusion matrices for the BC and Greedy methods.

translations for 7, which means 42.5% recall and 45.7% precision. Returning zero translations is a failure of our method which is discussed in Section 6. The performance of the Greedy method is higher for in-domain sentences (52.4% recall and precision).

For the 110 out-of-domain sentences, the BC method produces 24 perfect translations and returns zero translations for 33, which means 21.8% recall and 31.1% precision. The Greedy method has a performance of only 16.3% recall and precision on this corpus. These results show that, although the Greedy method performs well on in-domain sentences, it is more dependent than the BC method on the genre on which the parameters are trained. The BC method depends less on the statistical parameters and therefore has a better performance for out-of-domain sentences.

The BC$^+$ method outperforms the Stack and the Greedy methods for both in-domain and out-of-domain sentences. It has 61.3% recall and 65.9% precision for in-domain sentences, and 29.1% recall and 41.5% precision for out-of-domain sentences. This proves that, although the formula used for re-ranking is perhaps not optimal, the method proposed here can significantly improve translation accuracy.

Another useful comparison is shown in Table 2. It indicates the amount of overlap for the perfect translations produced by the BC and Greedy methods. The confusion matrices prove that these two methods yield quite orthogonal results, i.e., their results are produced independently and the correct translations do not necessarily overlap. For example, the BC method produces 12 in-domain and 14 out-of-domain perfect translations. These translations are not found by the Greedy decoder because they have a lower probability according to the translation model.

6 Discussion

In this section we examine a common cause for failure in our system, and also briefly discuss the possible use of this translation method for other language pairs.

A major cause of failure for the BC method is the gaps in the corpus S. By the very idea of this method – trying to find phrases in S that are translations of the initial sentence – the algorithm can fail to find any such possible translation, returning zero proposed translations. This type of failure has several possible fixes. One is to keep increasing the size of the corpus S beyond 1 billion words of magnitude. Intuitively this gives our algorithm an increased chance of finding good translation proposals. Another possible fix is to incorporate the BC method, together with other translation methods, into a multi-engine system which combines the strengths of each individual method.

And yet another possible approach to fixing this type of failure is to find a reliable mechanism for splitting up sentences into "independent" sub-parts (such as clauses, or elementary textual units [6]), and then translate them individually. We suspect that such an approach would also allow for the method proposed here to scale up to longer sentences without loosing much in the translation accuracy.

The method presented here has the potential to work for other language pairs as well, as long as a large monolingual corpus of the target language and a translation table for the language pair are available. The only extra condition that seems to be required is that the monolingual corpus is comparable content-wise with the corpus from which the sentences to be translated are extracted.

7 Building new parallel corpora

Parallel corpora are expensive resources that are time-consuming to build by humans; yet, they are crucial for building high-performance statistical machine translation systems. Although as a translation mechanism our method is both time and resource consuming, we believe it can also be used to automatically construct parallel corpora quicker and cheaper. We hypothesize that new phrase/sentence pairs aligned by our method can be extracted and used for training, in order to improve the estimates of the parameters of a statistical model.

References

1. Sprung, R., ed.: Translating Into Success: Cutting-Edge Strategies For Going Multilingual In A Global Age. John Benjamins Publishers (2000)
2. Grefenstette, G.: The world wide web as a resource for example-based machine translation tasks. In: ASLIB, Translating and the Computer 21, London (1999)
3. Brown, P., Della Pietra, S., Della Pietra, V., Mercer, R.: The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics **19** (1993) 263–311
4. Knight, K., Al-Onaizan, Y.: Translation with finite-state devices. In: Proceedings of the 4th AMTA Conference. (1998)

5. Germann, U., Jahr, M., Knight, K., Marcu, D., Yamada, K.: Fast decoding and optimal decoding for machine translation. In: Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL'01), Toulouse, France (2001)
6. Marcu, D.: A surface-based approach to identifying discourse markers and elementary textual units in unrestricted texts. In: Proceedings of the COLING/ACL–98 Workshop on Discourse Relations and Discourse Markers, Montreal, Canada (1998)