Stone Soup Translation[*]

Paul C. Davis and Chris Brew
Department of Linguistics
Ohio State University
222 Oxley Hall
1712 Neil Ave.
Columbus, OH 43210-1298
USA
{pcdavis, cbrew}@ling.ohio-state.edu

Abstract

The automated translation of one natural language to another, known as *machine translation*, typically requires successful modeling of the grammars of the two languages and of the relationship between them. Rather than hand-coding these grammars and relationships, some machine translation efforts have begun to employ statistical methods, where the goal is to learn from a large amount of training examples of accurate translations. This work has also been extended to probabilistic finite-state approaches, most often via transducers. In this project, a novel combination of finite-state devices is employed. The model proposed, which consists of two probabilistically *linked* automata, is more flexible than a transducer model, giving increased ability to handle word order differences. In addition to the model and algorithms for its construction and use, we present several increased-coverage techniques, including methods for extracting partial results from the model. We present preliminary results for a test corpus of English to Spanish translations, which suggest the model may serve as a base for rudimentary translation, when used in conjunction with these extensions.

1 Introduction

Creating a translation system for natural languages often involves not only successfully modeling the grammars of the languages, but also modeling the relationship between the two languages. These models may be hand-crafted, but in recent years many data-driven approaches, i.e., those that learn from translation examples, have been explored. Data-driven approaches are appealing because of their lack of reliance on sophisticated human linguistic input, as well as for their robustness, since the graceful degradation of many of these systems makes them more tolerant of noisy input.

Within this area of data-driven machine translation, are machine translation (MT) approaches which are purely statistical. Probably the best known example of statistical

[*]Thanks to Philip Resnik, for making multilingual Bibles available for research via the University of Maryland Parallel Corpus Project, to Dan Melamed and Franz Josef Och for discussions regarding word alignment resources, and to Hiyan Alshawi regarding evaluation metrics. This work benefited from the comments of Bob Kasper, Erhard Hinrichs, Detmar Meurers, Dale Russell, and three anonymous reviewers. Paul C. Davis is the recipient of a Motorola University Partnerships in Research Grant. Portions of this research were also funded by an Ohio State University Summer GRA in Cognitive Science.

machine translation (SMT) is Brown et al. (1993), which models the alignment between words in two languages. There are, however, a number of SMT approaches which seek to impose additional structure on the language and translation models via the use of finite-state devices. These range from techniques which attempt to replicate the pure SMT approach by means of composed transducers, such as Knight & Al-Onaizan (1998), to those which use subsequential transducers on limited domain translation tasks, such as Vilar et al. (1999) , and finally to those which employ more powerful finite-state devices, such as weighted head transducers (Alshawi et al. 2000), to model the hierarchical syntactic structure of sentences via dependency trees. What most of the finite-state MT approaches share is the use of transducers as the underlying mechanism for translation. The use of transducers and finite-state devices in general is motivated by the fact that they are well understood, and can be used quite efficiently, notwithstanding their known limitations in terms of expressivity.

the linked automata model Transducers, however, have the unfortunate quality of creating an ordering asynchrony between the two languages. For example, we might represent the relationship between word-aligned English and French sentences shown in figure 1 with the transducer show in figure 2.

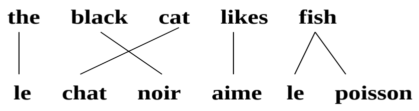

Figure 1: An example English and French word alignment

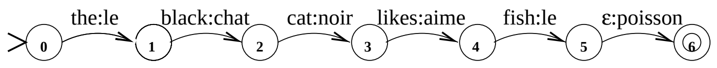

Figure 2: A transducer as a very simple translation system

This asynchrony means that one language imposes its linear ordering constraints on another and that the natural notion of a correspondence between words or word sequences in a translation is obscured. There exist methods for dealing with such asynchronies, such as using special symbols to indicate the original order (Vilar et al. 1999), but in this paper we present a more direct translation model: We represent the alignments between the words of *bitexts* (pairs of corresponding source and target sentences) literally, while still representing the relative ordering of the source and target language sentences (see figure 3).

Rather than using transducers, we move to *linked automata*, pairs of automata where the transitions are linked to one another by a weighted partial function. The alignment given in figure 1 can be incorporated directly into the linked automata model as shown in figure 3, where dotted arrows represent the alignments. The two automata constitute separate but linked language models for the two languages, and the linking function (or translation model) captures the crucial relationships between words on each side of the translation relation.

This linked automata model is extremely simple, and has limited descriptive power, since, for example, automata can only represent regular languages and not context-free

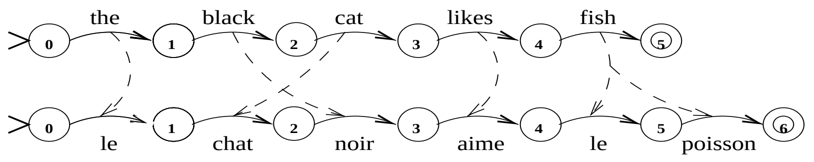

Figure 3: A bitext as represented in the linked automata MT model

languages, while natural languages are standardly assumed to be at least context-free. But by using techniques for approximation and generalization, and heuristics for working with partial results, the model may indeed prove to be adequate for translation tasks.[1] The model thus also bears a relationship to another type of data-driven MT, example-based machine translation, since when working with partial results, decisions need to be made with regard to where segments should be matched and how their translations should be recombined (see, for example, Nirenburg et al. (1993)).

In the remainder of this paper, we will, in section 2, briefly describe the training process for the model, including our use of probabilities and their motivation. In section 3 we describe the translation process, detailing how the linked automata model is used. In section 4 we outline methods we use for increased generalization as well as size-reduction of the system. In the final section, we describe our evaluation methodology, and present preliminary results and future research directions.

2 Training

A linked automata translation system consists of a source language automaton, a target language automaton, and an alignment table. The automata are standard, consisting of a finite set of states, Q, a start-state q_0 (numbered *0* in figures), a set of final states, $F \subseteq Q$ (shown in double circles), and a set of transitions, T. Each of the states are uniquely numbered so that we have a means to identify them. Transitions between states take the following form $< q_b, q_e, w, p >$, where q_b is the *begin state*, q_e is the *end state*, w is the label (w is for *word*), which may be empty (i.e., an epsilon-transition), and p is a probability associated with the transition. A string is recognized in the usual manner (i.e., we can recognize a string of words if there exists a path labeled with those words in the same order from the start-state to a final state). We also require that the automata be acyclic, otherwise we would not know how to order the words in a translation (see section 3).

The alignment table is a (partial) function from the power set of source transitions to a set of sets of pairs of target transitions and probabilities. For example, in figure 3, the source transition $< 2, 3, cat >$[2] would be aligned in the table with the target transition $< 1, 2, chat >$. Suppose that this alignment comprised 17 percent of the overall alignments, and further that $< 2, 3, cat >$ was never aligned with anything else during the training process. Then we would expect to see the following table entry for

[1]It is for this reason that we make reference to *Stone Soup*, after the story of a man who claimed he could make delicious soup with only a stone, and then proceeded to add other ingredients until he was able to amaze people with the soup's good taste.

[2]We show transition probabilities only when necessary.

this alignment: $(<2,3,cat>) \rightarrow \{((<1,2,chat>)\ \ .17)\}$. If we ignore the probabilities in the alignments, the translation model simply nondeterministically provides a set of translations for any input string which can be recognized.[3]

probabilities There are three major considerations regarding probabilities in the model: (1) the probabilities of the language models (the two automata); (2) the probabilities in the table (the alignment probabilities); and (3) the interaction between the two. Probabilities in the automata are implemented following Vilar et al. (1999). First, for any given state, we need to know the probability that it is final. The second probability, the transition probability, is defined so that for every state q, the sum of the probabilities of the transitions departing from q and the probability that q is final is equal to one. Thus, the probability of a string being accepted by the path of transitions: $<q_0, q_1, w_1, p_{0,1}>, <q_1, q_2, w_2, p_{1,2}>, \ldots, <q_{n-1}, q_n, w_n, p_{n-1,n}>$ is defined as:

(1) $P_{is\_final}(q_n) \prod_{i=1}^{n} p_{i-1,i}$

Thus the language model assigns a nonzero probability only to strings which are accepted. We are able to estimate both the state and transition probabilities by using the frequency of each transition during training.

Probabilities in the table, which we call *alignment probabilities*, are based on the alignments of words in bitexts from the training corpus. In order to find the correct alignment of a bitext, we first look for the best alignments of its component sequences. Our goal in the table is to represent this connection probabilistically, since sequences can be aligned in more than way. To do this, we count. That is, during the automata and table construction process, every time a sequence in the source automaton is aligned with a sequence in the target, we increment the table entry for this alignment. For example, an alignment between a source transition sequence, STS_i, and a target transition sequence, TTS_k, would result in changing the pair (where c is a count) $<TTS_k, c>$ to $<TTS_k, c+1>$.[4]

As will be described in section 3, translation consists in part of the transitions of the source automaton "activating" the transitions of the target automaton, via the links in the table. If all of the alignments in the table were equally likely, we could rely on the target automaton's transition probabilities alone to choose among competing sequences of transitions when looking for the most probable translation. Since, however, alignment probabilities vary, we rely on them to tell us how much to weight each target transition probability, i.e., for any given activated target transition, we multiply its probability by the probability of the alignment which generated it.[5] In this manner, each transition sequence in the target may be evaluated both in terms of its likelihood in the target language model and as having been generated via the alignment model. For example, a target transition sequence $<q_0, q_1, w_1, p_{0,1}>, <q_1, q_2, w_2, p_{1,2}>, \ldots,$

[3]In the alignment table presentation, we describe the translation process from source language to target language. The alignments are in fact, bidirectional, in the sense that the table is easily inverted, allowing for translation in either direction.

[4]The presentation here is slightly simplified. We normalize the probabilities by the total number of alignments for the whole table.

[5]Multiplication is not the only option here, but has the merit of simplicity.

$< q_{n-1}, q_n, w_n, p_{n-1,n} >$ with alignment probabilities $a_1, a_2, \ldots, a_n$ will have the overall translation probability:

(2) $P_{is\_final}(q_n) \prod_{i=1}^{n} a_i p_{i-1,i}$

construction The linked automata model is constructed from word-aligned bitexts. For the results reported in this paper, we used English and Spanish versions of the Judeo-Christian Bible.[6] We experimented with two different automatic word-aligners: a rather poor-performing word-aligner we created for the task, and the higher quality (word alignments of the) Giza^{++} translation model (Och & Ney 2000).[7] Construction from the word-aligned bitexts is straightforward: From a file of bitext triples (a source sentence, a target sentence, and a representation of the alignment between them), for each triple, add the appropriate states and transitions to the source automaton so that the source sentence can be recognized, perform the analogous additions to the target automaton with respect to the target sentence, and for each alignment pair of the form: $((source\_tran_i ... source\_tran_j)\ (target\_tran_k ... target\_tran_l))$, add an entry to the table for the source sequence if one does not already exist. Then, within this entry for the source transition sequence, increment the count for the target transition sequence by one, after first creating it if necessary. The top-level system construction algorithm is shown in figure 4. One version of the construction process additionally permits merging (see section 4) for generalization.

```
For each bitext triple $< source\_string,\ target\_string,\ aligned\_pairs >$
{ source_wordnum-to-transition_hash = add_to_fsa(source_fsa, source_string)
  target_wordnum-to-transition_hash = add_to_fsa(target_fsa, target_string)
  for each aligned pair $< source\_wordnums, target\_wordnums >$
  { add_to_alignment_table(
        (get_trans_seq(source_wordnums, source_wordnum-to-transition_hash))
        (get_trans_seq(target_wordnums, target_wordnum-to-transition_hash)))}}
```

Figure 4: Linked automata model construction algorithm from aligned bitexts

3 The Translation Process

Translation consists of three stages: (I) processing the source sentence; (II) retrieving the target transition sequences to be activated from the table; and (III) using these target transitions to find all allowable paths through the target automaton (i.e., all the translations). Continuing with the *the black cat likes fish* translation example (shown earlier as a portion of a linked automata translation model in figure 3), we illustrate the translation algorithm in figure 5.

We begin with a string of source words, *S*, and attempt to recognize *S* in the source automaton. If successful, we collect the transitions of the most probable source transition sequence (STS). In the second stage, we take the STS and use it to select

[6]We chose to use the Bible because of its electronic availability and the ease of sentence alignment.

[7]Better quality word alignment usually leads to better translations, as shown with the results in table 1. We measured the quality of word-alignments independently against a hand-aligned corpus.

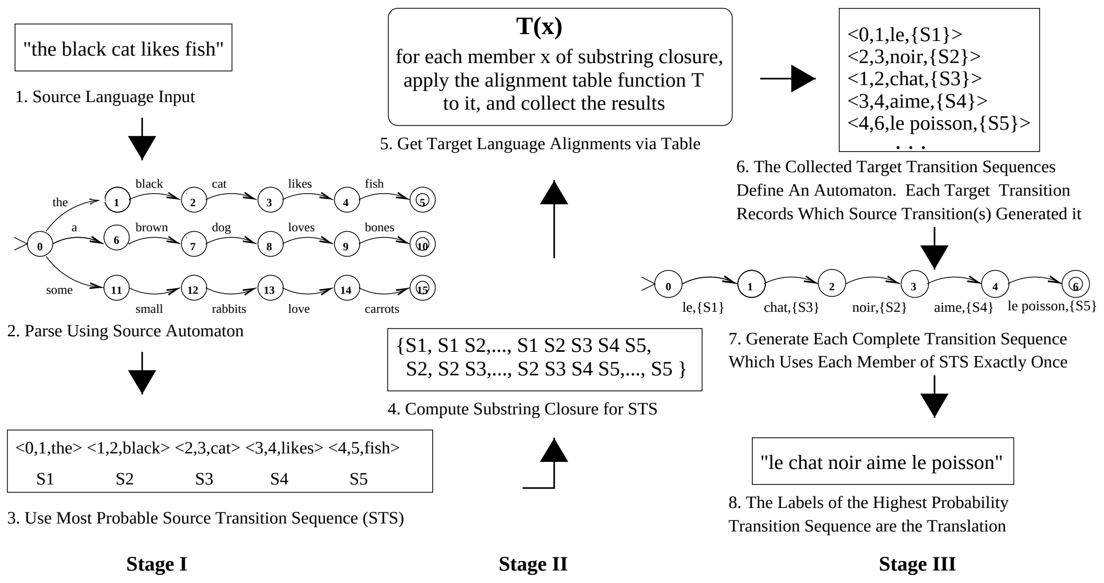
"the black cat likes fish"
1. Source Language Input
the
black
cat
likes
fish
a
brown
dog
loves
bones
some
small
rabbits
love
carrots
2. Parse Using Source Automaton
<0,1,the> <1,2,black> <2,3,cat> <3,4,likes> <4,5,fish>
S1 S2 S3 S4 S5
3. Use Most Probable Source Transition Sequence (STS)
Stage I
T(x)
for each member x of substring closure, apply the alignment table function T to it, and collect the results
5. Get Target Language Alignments via Table
{S1, S1 S2,..., S1 S2 S3 S4 S5, S2, S2 S3,..., S2 S3 S4 S5,..., S5 }
4. Compute Substring Closure for STS
Stage II
<0,1,le,{S1}>
<2,3,noir,{S2}>
<1,2,chat,{S3}>
<3,4,aime,{S4}>
<4,6,le poisson,{S5}>
. . .
6. The Collected Target Transition Sequences Define An Automaton. Each Target Transition Records Which Source Transition(s) Generated it
le,{S1}
chat,{S3}
noir,{S2}
aime,{S4}
le poisson,{S5}
7. Generate Each Complete Transition Sequence Which Uses Each Member of STS Exactly Once
"le chat noir aime le poisson"
8. The Labels of the Highest Probability Transition Sequence are the Translation
Stage III

Figure 5: The linked automata translation algorithm

target transitions from the transition alignment table. This is not, however, simply a matter of getting the value for each source transition from the table, since alignments are between sequences of transitions. In principle, any alignment type is allowed, e.g., 2 transitions to 1, 5 to 6, etc. So, we first compute the *substring closure* for the STS, which is simply the set of all substrings of a sequence, including the empty string. Then, for each member of the substring closure, we get the value from the table. If we think of the table as a function, T, we simply apply T to each member of the substring closure, and collect the results.[8] These collected results contain all the target transitions which will be activated, and lead us to the final translation stage.

At this point we have a set of activated target automaton transitions. Our strategy is to put these transitions together in all ways that the target language model allows, then to choose the best one. In our earliest implementation, we used a modified chart parser to put the activated transitions together. But this step lacked a crucial insight: The activated transitions, taken as a unit, are already connected. In fact, they define an automaton, one which is simply a much smaller part of the original target automaton. Imagine an automaton with thousands of transitions, where a small number of the transitions are chosen randomly. These transitions, along with the original start-state and final state information, form an automaton.

Thus, putting these activated transitions together is just a matter of generating every possible transition sequence from this smaller automaton, which is easily computed given the automaton's small size. In addition, we want to ensure that each source

[8]The values of the function in figure 5 (step 6) match those produced by following the dotted alignment arrows shown in figure 3. Note that using the substring closure precludes the use of *discontinuous alignments* with source words, i.e., those where the source words of the alignment do not all occur in sequence. This is intentional, since the word-alignment algorithms tested do not allow such alignments. Nothing in the model's architecture, however, needs to be changed to use them, and we will present an algorithm for translating with such alignments in a future paper.

word is used exactly once to generate each complete target transition sequence. We accomplish this by adding another component to transitions, called the source-word-store (sws), inspired by Johnston (1998), which contains a numerical representation of the source words which generated the transition. A complete parse, then, (i.e., a potential translation), is a sequence of transitions which begins at the start-state, ends at a final state, and has a *full* sws (i.e., one that covers all the source words). The best translation is the combined labels (i.e., the words) of the highest scoring sequence.

4 Generalization

The model as described is limited to only the exact bitexts on which it was trained. We use a number of techniques to increase its ability to generalize (as well as reduce its size, a nontrivial problem for this type of translation model), the most important of which is merging. We merge transitions in the automata (transition merging can alternatively be viewed as merging two pairs of states) only if the result would not cause us to "lose" any translations the model already covers. We refer to this requirement as preserving the *translation integrity*, and define it as:

(3) Let P, Q be the sets of strings over the sets of source and target words, respectively, $T \subseteq P \times Q$ be the set of training bitexts, and T' be the set of translations in the merged translation system, then translation integrity is:
$(\forall p)(\forall q)\ (< p, q > \in T \rightarrow < p, q > \in T')$

In order to satisfy this requirement (and others of our approach), we allow transitions to be merged only if they have the same label, the same translations, and their merging would not create a cycle. To complete a merge we make the appropriate adjustments to the table as well as the automata.

Although we do not describe merging or its consequences in detail here, we provide an example below in figure 6, which shows how coverage can be increased at the level of the source automaton via the merging of transitions. In this example, the automaton on the left recognizes the two sentences *the cat likes fish* and *a dog likes bones.* After merging the two transitions labeled *likes*, to get the automaton on the right side of the figure, the automaton can recognize the two original sentences, plus *the cat likes bones* and *a dog likes fish*, even though these two new sentences were not in the training data.

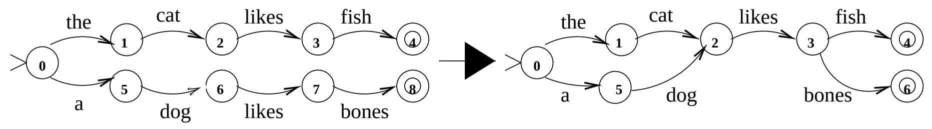

Figure 6: Merging on the automaton level

We also use four additional generalization techniques, which we regard as common sense heuristics: (1) *fragment processing*, (2) *unknown word fall-through*, (3) *partial source parsing*, and (4) *partial target parsing.* Fragment processing expands our coverage to process not only every training source sentence as a valid input, but also any substring of it, and similarly for the target language model. Unknown word fall-through means that rather than completely rejecting source sentences which contain words which were unseen during training, we let the unknown words fall through untranslated, and translate the words on either side of the unknown words, as if these word sequences were translated on their own.

Next is partial source parsing. Suppose that all the individual words in a source string have been seen before, but that we cannot recognize the string itself, i.e., we cannot find any path in the source automaton for the given string, even if we allow fragments. Once again, perhaps the most sensible thing to do is look for 'parts' of this source string which we can recognize, and to translate these parts individually in the linear order they occur. A straightforward greedy algorithm is to get the longest substring which we can recognize, translate it, and then do the same with the remaining parts. This approach, at its worst, devolves to a simple word-for-word translation, where the source string imposes its word ordering on the translated target words.

Partial target parsing is used at the last stage of the translation process to order unconnected transitions. This is necessary when no sequence is found which represents a complete transition path in the target automaton. This situation can arise as the result of merging and the other generalization techniques. The algorithm employed uses two heuristics. First, given two activated transitions A and B, if A precedes B in the automaton, then A precedes B in the resulting translation. We use the following second heuristic only if the first heuristic does not hold in either direction: If the distance from the start-state to A is less than the distance to B, then A precedes B in the translation. Using these techniques, we can guarantee that the system always produces something; a property that simplifies the task of evaluation.

5 Evaluation and Future Directions

There appear to be as many different evaluation methods for machine translation as there are machine translation methods. This arises from the fact that there is little agreement on how to define what a correct translation is, much less how to measure it, be the measurement automatic or human (White 2000). MT evaluation (MTE) has become a small research niche in its own right, but these efforts have yielded few uncontroversial results. It comes down to one of the fundamental truths of MTE: "There is no such thing as *the* correct translation" (King (1997:261), emphasis added).

In an evaluation of system such as that proposed in this paper, we are primarily interested in determining if the overall approach is feasible, as well as being able to assess to what degree the generalization methods improve performance. As such, automatic evaluations are suitable (White 2000), especially since small changes in the model may yield large changes in performance, and string-based evaluation metrics appear to be appropriate. We use *edit-distance*, the minimum number of insertions, deletions, or substitutions to convert one string to another (see Kruskal (1999)). In this case, the edit-distance is measured in terms of words.

Edit-distance returns a natural number, but the meaning of these numbers depends on the length of the sentences being tested. A measure that we can compare across test examples and that returns a value between 0 and 1 is more suitable. Following Alshawi et al. (2000), we use *simple-accuracy* (SA), and *translation-accuracy* (TA), as defined below, where I, D, and S are the number of insertions, deletions, and substitutions, respectively, I' and D' are insertions and deletions if transpositions, T, are taken into account, and R is the length of the reference translation.

(4) $\textit{simple-accuracy} = 1 - ((I + D + S)/R)$

(5) $\textit{translation-accuracy} = 1 - ((I' + D' + S + T)/R)$

Alshawi et al. (2000) view translation accuracy as a more appropriate measure for translation because transpositions of otherwise correct words count as one error (a transposition), rather than two (a deletion plus an insertion).[9]

As mentioned in section 2, we trained the model with two different sets of word-aligned bitexts (using 1,529 English and Spanish verses from the Bible): one set of low quality (set A in table 1), and one of better quality (the Giza^{++} alignments, set B in table 1). We report SA and TA results for both training sets, and, for brevity, report translation times and system size for training set A only. The 1,529 bitexts consisted of 38,364 English words, 2,443 of which were unique, and 34,779 Spanish words, 3,861 which were unique. To give an idea of the system size, for training set A, the source automaton consisted of 34,537 states in an unmerged system. Using the most conservative merging algorithm during construction, this size was reduced by 31%.[10] Clearly size considerations are an important issue for this type of translation model, and less conservative merging techniques will need to be tested.

In table 1, we report our results for four different test suites (each consisting of ten English and Spanish bitexts), and a variety of increased-coverage options, listed so that tests using more of the options come later, for each test-suite. For a given test configuration, merged system results always follow unmerged system results. Test-suite 1 consisted of bitexts from the training sets and thus was not really a test, but rather a reality check, just to make sure that our system could handle the bitexts on which it was trained. Test-suite 2 consisted of bitexts from the Bible which were not in the training data, but contained no unknown source words. Test-suite 3 bitexts used unseen Bible sentences that contained unknown words (hence the most difficult of the tests). In test-suite 4, we took some of the training sentences and replaced a number of words in them with (known) words of the same lexical category, to create sentences not in the training data, but which were quite close (hence the easiest of the latter 3 tests). In general, the system performed as expected, showing significant improvement as the increased-coverage heuristics were added, but poor performance without them. Merging had the greatest effect for the most difficult tests, i.e., where additional generalization would be expected to be the most helpful. In general, rough translations of long (> 24 words), difficult sentences, could be accomplished within six

[9]Intuitively, SA can be thought of as measuring what percentage of words are correct and in the proper position in the translation. Note, however, that these measures do not strictly yield a positive number, since it is conceivable to have a translation which is wrong by more words than R. In such instances we treat the distance as R, thus SA and TA are 0.

[10]The merging reported on here was limited to singleton source transition sequences and alignments to singleton sets of target transition sequences, so significantly more merging is possible which is guaranteed to preserve the translation integrity.

TS	SrcL	Mrg	PSP	FRG	PTP	UFT	A-wa SA/TA	B-wa SA/TA	Time (sec)
1	20.3	no	no	no	no	no	1.00/1.00	1.00/1.00	0.4
1	20.3	yes	no	no	no	no	0.99/0.99	0.99/0.99	1.4
2	19.5	no	no	no	no	no	0.00/0.00	0.00/0.00	0.0
2	19.5	yes	no	no	no	no	0.00/0.00	0.00/0.00	0.0
2	19.5	no	yes	yes	no	no	0.16/0.16	0.39/0.39	4.5
2	19.5	yes	yes	yes	no	no	0.13/0.13	0.38/0.39	4.1
2	19.5	no	yes	yes	yes	no	0.31/0.31	0.40/0.40	4.5
2	19.5	yes	yes	yes	yes	no	0.28/0.28	0.39/0.40	4.3
3	24.3	no	no	yes	no	yes	0.02/0.02	0.02/0.02	0.2
3	24.3	yes	no	yes	no	yes	0.02/0.02	0.02/0.02	0.4
3	24.3	no	yes	yes	no	yes	0.16/0.17	0.33/0.33	5.6
3	24.3	yes	yes	yes	no	yes	0.21/0.21	0.36/0.36	5.4
3	24.3	no	yes	yes	yes	yes	0.22/0.23	0.34/0.34	5.6
3	24.3	yes	yes	yes	yes	yes	0.31/0.31	0.38/0.38	5.6
4	17.6	no	no	no	no	no	0.00/0.00	0.00/0.00	0.0
4	17.6	yes	no	no	no	no	0.00/0.00	0.00/0.00	0.0
4	17.6	no	yes	yes	no	no	0.66/0.66	0.51/0.53	1.1
4	17.6	yes	yes	yes	no	no	0.62/0.62	0.51/0.51	1.1
4	17.6	no	yes	yes	yes	no	0.88/0.88	0.80/0.82	1.2
4	17.6	yes	yes	yes	yes	no	0.84/0.85	0.80/0.81	1.2

Table 1: Summary of test results *key: TS=test-suite, SrcL=mean source sentence length, Mrg=Merged, PSP=partial source parsing, FRG= fragment processing, PTP=partial target parsing, UFT=unknown word fall-through, A-wa=low quality word-alignment, B-wa=better quality word-alignment, SA/TA=simple-accuracy/translation-accuracy, Time=mean run time*

seconds.[11] The better quality training set, set B, led to better accuracies, as expected, except for test-suite 4. We view these results as an indication that the system may be a promising base for future development, but must note that overall accuracy remains low, with scores at .40 or below for test-suites 2 and 3. We see several reasons for the low accuracy, the most important of which is insufficient generalization. Only in test-suite 3 does merging improve accuracy. Also, we suspect that the training sets were too small, and that at this stage in development, the Bible may be too difficult a translation task. Lastly, incorrect word-alignment and less than optimal partial parsing algorithms are likely sources of many errors. More fundamentally, the model remains subject to many of the limitations faced by other finite-state models which do not fully take hierarchical structure into account, but we believe there remains ample room to see how far such a model can be pushed as a translation system.

future research The most immediate goal is to achieve more generalization (via merging) and thus better accuracy and smaller model size. Additional future plans to

[11] Testing was done on a 500MHz Sun Blade-100. The system performance was optimized to the point were translation was fast enough for frequent testing, but further improvement should be possible. Off-line tasks, such as training, were significantly longer, with training on 1529 bitexts ranging from 30 seconds (no merging) to 1 hour (with merging).

improve the coverage extend to a number of areas where we would make use of more linguistic information. We next expect to explore the use of part-of-speech (POS) tags. By using POS tags, we can both differentiate senses of words, and begin to make generalizations about categories. We are also considering adding special handling of function words, as well as presegmentation components, either morphological (lemmatization) or syntactic (partial parsing), as well as improving our algorithms for extracting partial results. Another idea is to allow the context of the translations to play a role. This could be accomplished by allowing the activation of used target transitions to slowly decay, rather than start anew with each translation task. With extensions such as these, we have additional ingredients to make a better translation soup.

References

Alshawi, Hiyan, Srinivas Bangalore & Shona Douglas: 2000, 'Learning dependency translation models as collections of finite-state head transducers', *Computational Linguistics*, **26**(1): 45–60.

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra & Robert L. Mercer: 1993, 'The mathematics of statistical machine translation: Parameter estimation', *Computational Linguistics*, **19**(2): 263–311.

Johnston, Michael: 1998, 'Unification-based multimodal parsing', in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, Montreal, Quebec, Canada, pp. 624–630.

King, Margaret: 1997, 'Evaluating translation', in Christa Hauenschild & Susanne Heizmann, eds., *Machine Translation and Translation Theory*, Berlin: Mouton de Gruyter, pp. 251–263.

Knight, Kevin & Yaser Al-Onaizan: 1998, 'Translation with finite-state devices', in David Farwell, Laurie Gerber & Eduard Hovy, eds., *Machine Translation and the Information Soup: Proceedings of the Third Conference of the Association of Machine Translation in the Americas, AMTA '98*, Berlin: Springer-Verlag.

Kruskal, Joseph: 1999, 'An overview of sequence comparison', in David Sankoff & Joseph Kruskal, eds., *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Stanford: CSLI Publications, Reissued edition with an introduction by John Nerbonne, pp. 1–44.

Nirenburg, Sergei, Constantine Domashnev & Dean J. Grannes: 1993, 'Two approaches to matching in example-based machine translation', in *Fifth International Conference on Theoretical and Methodological Issues in Machine Translation, TMI '93: MT in the next generation*, Kyoto, Japan, pp. 47–57.

Och, F. J. & H. Ney: 2000, 'Improved statistical alignment models', in *Proceedings of the 38th Annual Meeting of the ACL*, Hong Kong, China, pp. 440–447.

Vilar, Juan Miguel, Victor Manuel Jiménez, Juan Carlos Amengual, Antonio Castellanos, David Llorens & Enrique Vidal: 1999, 'Text and speech translation by means of subsequential transducers', in Andras Kornai, ed., *Extended Finite State Models of Language*, Cambridge: Cambridge University Press, pp. 121–139.

White, John S.: 2000, 'Contemplating automatic MT evaluation', in John S. White, ed., *Envisioning Machine Translation in the Information Future: 4th Conference of the Association for Machine Translation in the Americas, AMTA 2000*, Berlin: Springer, Lecture Notes in Artificial Intelligence #1934, pp. 100–108.