Decoding Complexity in Word-Replacement Translation Models

Kevin Knight[*]
University of Southern California

Statistical machine translation is a relatively new approach to the longstanding problem of translating human languages by computer. Current statistical techniques uncover translation rules from bilingual training texts and use those rules to translate new texts. The general architecture is the source-channel model: an English string is statistically generated (source), then statistically transformed into French (channel). In order to translate (or "decode") a French string, we look for the most likely English source. We show that for the simplest form of statistical models, this problem is NP-complete, i.e., probably exponential in the length of the observed sentence. We trace this complexity to factors not present in other decoding problems.

1. Introduction

Statistical models are widely used in attacking natural language problems. The *source-channel* framework is especially popular, finding applications in part-of-speech tagging (Church, 1988), accent restoration (Simard, 1998), transliteration (Knight and Graehl, 1998), speech recognition (Jelinek, 1998), and many other areas. In this framework, we build an underspecified model of how certain structures (such as strings) are generated and transformed. We then instantiate the model through training on a database of sample structures and transformations.

Recently, (Brown et al., 1993) built a source-channel model of translation between English and French. They assume that English strings are produced according to some stochastic process (source model) and transformed stochastically into French strings (channel model). To translate French to English, it is necessary to find an English source string that is likely according to the models. With a nod to its cryptographic antecedents, this kind of translation is called *decoding*. This paper looks at decoding complexity.

2. Part-of-Speech Tagging

The prototype source-channel application in natural language is part-of-speech tagging (Church, 1988). We review it here for purposes of comparison with machine translation.

Source strings comprise sequences of part-of-speech tags like *noun*, *verb*, etc. A simple source model assigns a probability to a tag sequence $t_1 \cdots t_m$ based on the probabilities of the tag pairs inside it. Target strings are English sentences, e.g., $w_1 \cdots w_m$. The channel model assumes each tag is probabilistically replaced by a word (e.g., *noun* by *dog*) without considering context. More concretely, we have:

- v total tags

[*] Information Sciences Institute, Marina del Rey, CA 90292

- A *bigram* source model with v^2 parameters of the form $\mathrm{b}(t|t)$, where $\mathrm{P}(t_1 \cdots t_m)$ $\sim \mathrm{b}(t_1|\text{boundary}) \cdot \mathrm{b}(t_2|t_1) \cdot \ldots \cdot \mathrm{b}(t_n|t_{m-1}) \cdot \mathrm{b}(\text{boundary}|t_m)$
- A *substitution* channel model with parameters of the form $\mathrm{s}(w|t)$, where $\mathrm{P}(w_1 \cdots w_m|t_1 \cdots t_m) \sim \mathrm{s}(w_1|t_1) \cdot \mathrm{s}(w_2|t_2) \cdot \ldots \cdot \mathrm{s}(w_m|t_m)$
- an m-word text annotated with correct tags
- an m-word unannotated text

We can assign parts-of-speech to a previously unseen word sequence $w_1 \cdots w_m$ by finding the sequence $t_1 \cdots t_m$ that maximizes $\mathrm{P}(t_1 \cdots t_m|w_1 \cdots w_m)$. By Bayes' Rule, we can equivalently maximize $\mathrm{P}(t_1 \cdots t_m) \cdot \mathrm{P}(w_1 \cdots w_m|t_1 \cdots t_m)$, which we can calculate directly from the b and s tables above.

Three interesting complexity problems in the source/channel framework are:

1. Can source and channel parameters be induced from annotated text efficiently?
2. Can optimal decodings be produced efficiently?
3. Can source and channel parameters be induced from unannotated text efficiently?

Problem (1) is solved in $\mathrm{O}(m)$ time for part-of-speech tagging—we simply count tag pairs and word/tag pairs, then normalize. Problem (2) seems to require enumerating all $\mathrm{O}(v^m)$ potential source sequences to find the best, but can actually be solved in $\mathrm{O}(mv^2)$ time with dynamic programming (Viterbi, 1967). Problem (3) can also be attacked by exploring the space of potential source sequences, using the estimation-maximization (EM) algorithm (Baum, 1972; Dempster et al., 1977). (Merialdo, 1994) describes such an induction algorithm that presumably runs in $\mathrm{O}(mv^2)$ time (per EM iteration); he obtains additional savings by restricting the channel to consider only the word/tag connections consistent with a given part-of-speech dictionary.

3. Substitution Ciphers

Before we turn to translation, it is also useful to consider simple substitution ciphers, in which a plaintext message like `HELLO WORLD` is transformed into a ciphertext message like `EOPPX YXAPF`, via a fixed letter-substitution table. As with tagging, we can assume an alphabet of v source tokens, a bigram source model, a substitution channel model, and an m-token coded text.

If the coded text is annotated with corresponding English, then building source and channel models is trivially $\mathrm{O}(m)$. Comparing the situation to part-of-speech tagging:

- (Bad news.) Cryptanalysts rarely get such coded/decoded text pairs and must employ "ciphertext-only" attacks using unannotated training data.
- (Good news.) It is easy to train a *source* model separately, on raw unannotated English text that is unconnected to the ciphertext.

Then the problem becomes one of acquiring a channel model, i.e., a table $\mathrm{s}(f|e)$ with an entry for each code-letter/plaintext-letter pair.

Figure 1 shows a naive EM implementation that runs in $\mathrm{O}(mv^m)$ time. At each iteration, the algorithm revises its hypothesis $\mathrm{s}(f|e)$ so as to increase the probability of the observed corpus P(f). Figure 2 shows an efficient $\mathrm{O}(mv^2)$ EM implementation, based on dynamic programming (Baum, 1972) that accomplishes the same thing. Once the $\mathrm{s}(f|e)$ table has been learned, there is a similar $\mathrm{O}(mv^2)$ algorithm for optimal decoding. Such methods can break English letter-substitution ciphers of moderate size.

```
Given coded text f of length $m$, a plaintext vocabulary of $v$ tokens, and a source model b:

  1. set the $s(f|e)$ table initially to be uniform
  2. for several iterations do:
     a. set up a count table $c(f|e)$ with zero entries
     b. $P(f) = 0$
     c. for all possible source texts $e_1 \cdots e_m$ ($e_i$ drawn from vocabulary)
          compute $P(e) = b(e_1 \mid \text{boundary}) \cdot b(\text{boundary} \mid e_m) \cdot \prod_{i=2}^{m} b(e_i|e_{i-1})$
          compute $P(f|e) = \prod_{j=1}^{m} s(f_j|e_j)$
          $P(f) \mathrel{+}= P(e) \cdot P(f|e)$
     d. for all source texts e of length $m$
          compute $P(e|f) = \frac{P(e) \cdot P(f|e)}{P(f)}$
          for $j = 1$ to $m$
            $c(f_j|e_j) \mathrel{+}= P(e|f)$
     e. normalize $c(f|e)$ table to create a revised $s(f|e)$
```

Figure 1
A naive application of the EM algorithm to break a substitution cipher. It runs in $O(mv^m)$ time.

```
Given coded text f of length $m$, a plaintext vocabulary of $v$ tokens, and a source model b,

  1. set the $s(f|e)$ table initially to be uniform
  2. for several iterations do:
     a. set up a $c(f|e)$ table with zero entries
     b. for $i = 1$ to $v$
          $Q[i,1] = b(e_i \mid \text{boundary})$
     c. for $j = 2$ to $m$
          for $i = 1$ to $v$
            $Q[i,j] = 0$
            for $k = 1$ to $v$
              $Q[i,j] \mathrel{+}= Q[k,j-1] \cdot b(e_i|e_k) \cdot s(f_{j-1}|e_k)$
     d. for $i = 1$ to $v$
          $R[i,m] = b(\text{boundary} \mid e_i)$
     e. for $j = m-1$ to $1$
          for $i = 1$ to $v$
            $R[i,j] = 0$
            for $k = 1$ to $v$
              $R[i,j] \mathrel{+}= R[k,j+1] \cdot b(e_k|e_i) \cdot s(f_{j+1}|e_k)$
     f. for $j = 1$ to $m$
            for $i = 1$ to $v$
              $c(f_j|e_i) \mathrel{+}= Q[i,j] \cdot R[i,j] \cdot s(f_j|e_i)$
     g. normalize $c(f|e)$ table to create a revised $s(f|e)$
```

Figure 2
An efficient $O(mv^2)$ algorithm that accomplishes the same thing as Figure 1.

4. Machine Translation

In our discussion of substitution ciphers, we were on relatively sure ground—the channel model we assumed in decoding is actually the same one used by the cipher-writer for encoding. That is, we know that plaintext is converted to ciphertext, letter by letter, according to some table. We have no such clear conception about how English gets converted to French, although many theories exist. (Brown et al., 1993) recently cast some simple theories into a source-channel framework, using the bilingual Canadian parliament proceedings as training data. We may assume:

- v total English words.
- A bigram source model with v^2 parameters.
- Various substitution/permutation channel models.
- A collection of bilingual sentence pairs, with maximum sentence lengths m.
- A collection of monolingual French sentences, each m words or fewer.

Bilingual texts seem to exhibit English words getting substituted with French ones, though not one-for-one and not without changing their order. These are important departures from the two applications discussed earlier.

In main channel model of (Brown et al., 1993), each English word token e_i in a source sentence is assigned a "fertility" ϕ_i which dictates how many French words it will produce. These assignments are made stochastically according to a table $\mathrm{n}(\phi|e)$. Then actual French words are produced according to $\mathrm{s}(f|e)$ and permuted into new positions according to a distortion table $\mathrm{d}(j|i,m,l)$. Here, j and i are absolute target/source word positions within a sentence, and m and l are target/source sentence lengths.

Inducing n, s, and d parameter estimates is easy if we are given annotations in the form of word *alignments*. An alignment is a set of connections between English and French words in a sentence pair. In (Brown et al., 1993), alignments are asymmetric—each French word is connected to exactly one English word.

Word-aligned data is usually not available, but large sets of unaligned bilingual sentence pairs do sometimes exist. A single sentence pair will have l^m possible alignments—for each French word position $1 \ldots m$, there is a choice of l English positions to connect to. A naive EM implementation will collect n, s, and d counts by considering each alignment, but this is expensive. (By contrast, part-of-speech tagging involves a single alignment, leading to $O(m)$ training). Lacking a polynomial reformulation, (Brown et al., 1993) decide to collect counts only over a subset of likely alignments. To bootstrap, they require some initial idea of what alignments are reasonable, so they begin with several iterations of a simpler channel model (called Model 1) that has nicer computational properties.

In the following description of Model 1, we represent an alignment formally as a vector $a_1, \ldots, a_m$, with values a_j ranging over English word positions $1 \ldots l$.

```
Model 1 Channel
Parameters: $\epsilon(m|l)$ and $\mathrm{s}(f|e)$.
Given a source sentence e of length $l$:

  1. choose a target sentence length $m$ according to $\epsilon(m|l)$
  2. for $j = 1$ to $m$, choose an English word position $a_j$ according to the uniform
     distribution over $1 \ldots l$
  3. for $j = 1$ to $m$, choose a French word $f_j$ according to $\mathrm{s}(f_j|e_{a_j})$
  4. read off $f_1 \cdots f_m$ as the target sentence
```

```
Given a collection of sentence pairs:

  1. collect estimates for the $\epsilon(m|l)$ table directly from the data
  2. set the $s(f|e)$ table initially to be uniform
  3. for several iterations do:
     a. set up a count table $c(f|e)$ with zero entries
     b. for each given sentence pair e, f with respective lengths $l, m$:
          for $a_1 = 1$ to $l$
            for $a_2 = 1$ to $l$          /* select connections for a word alignment */
              ...
                for $a_m = 1$ to $l$
                  compute $P(a_1, \ldots, a_m \mid e,f) = \frac{P(f, a_1, \ldots, a_m|e)}{P(f|e)} = \frac{\prod_{j=1}^{m} s(f_j|e_{a_j})}{\sum_{a'_1=1}^{l} \sum_{a'_2=1}^{l} \cdots \sum_{a'_m=1}^{l} \prod_{j=1}^{m} s(f_j|e_{a'_j})}$
                  for $j = 1$ to $m$
                    $c(f_j|e_{a_j})$ += $P(a_1 \ldots a_m|e,f)$
     c. normalize $c(f_j|e_i)$ table to create new $s(f_j|e_i)$
```

Figure 3
Naive EM training for the Model 1 channel model.

Because the same e may produce the same f by means of many different alignments, we must sum over all of them to obtain P(f|e):

$$P(f|e) = \epsilon(m|l)\ \frac{1}{l^m} \sum_{a_1=1}^{l} \sum_{a_2=1}^{l} \cdots \sum_{a_m=1}^{l} \prod_{j=1}^{m} s(f_j|e_{a_j})$$

Figure 3 illustrates naive EM training for Model 1. If we compute P(f|e) once per iteration, outside the "for a" loops, then the complexity is $O(ml^m)$ per sentence pair, per iteration.

More efficient $O(lm)$ training was devised by (Brown et al., 1993). Instead of processing each alignment separately, they modify the algorithm in Figure 3 as follows:

```
b. for each given sentence pair e, f of respective lengths $l, m$:
     for $j = 1$ to $m$
       $sum = 0$
       for $i = 1$ to $l$
         $sum$ += $s(f_j|e_i)$
       for $i = 1$ to $l$
         $c(f_j|e_i)$ += $s(f_j|e_i)$ / $sum$
```

This works because of the algebraic trick that the portion of P(f|e) we originally wrote as $\sum_{a_1=1}^{l} \sum_{a_2=1}^{l} \cdots \sum_{a_m=1}^{l} \prod_{j=1}^{m} s(f_j|e_{a_j})$ can be rewritten as $\prod_{j=1}^{m} \sum_{i=1}^{l} s(f_j|e_i)$.

We next consider decoding. We seek a string e that maximizes P(e|f), or equivalently maximizes P(e) · P(f|e). A naive algorithm would evaluate all possible source strings, whose lengths are potentially unbounded. If we limit our search to strings at most twice the length m of our observed French, then we have a naive $O(m^2 v^{2m})$ method:

```
Given a string f of length $m$
  1. for all source strings e of length $l \leq 2m$:
      a. compute $\mathrm{P(e)} = \mathrm{b}(e_1 \mid \text{boundary}) \cdot \mathrm{b}(\text{boundary} \mid e_l) \cdot \prod_{i=2}^{l} \mathrm{b}(e_i|e_{i-1})$
      b. compute $\mathrm{P(f|e)} = \epsilon(m|l)\ \frac{1}{l^m} \prod_{j=1}^{m} \sum_{i=1}^{l} \mathrm{s}(f_j|e_i)$
      c. compute $\mathrm{P(e|f)} \sim \mathrm{P(e)} \cdot \mathrm{P(f|e)}$
      d. if P(e|f) is the best so far, remember it
  2. print best e
```

We may now hope to find a way of reorganizing this computation, using tricks like the ones above. Unfortunately, we are unlikely to succeed, as we now show. For proof purposes, we define our optimization problem with an associated yes/no decision problem:

M1-OPTIMIZE
Given a string f of length m and a set of parameter tables (b, ϵ, s),
return a string e of length $l \leq 2m$ that maximizes P(e|f), or equivalently maximizes
$\mathrm{P(e)} \cdot \mathrm{P(f|e)} = \mathrm{b}(e_1 \mid \text{boundary}) \cdot \mathrm{b}(\text{boundary} \mid e_l) \cdot \prod_{i=2}^{l} \mathrm{b}(e_i|e_{i-1}) \cdot$
$\epsilon(m|l)\ \frac{1}{l^m} \prod_{j=1}^{m} \sum_{i=1}^{l} \mathrm{s}(f_j|e_i)$

M1-DECIDE
Given a string f of length m, a set of parameter tables (b, ϵ, s), and a real number k,
does there exist a string e of length $l \leq 2m$ such that $\mathrm{P(e)} \cdot \mathrm{P(f|e)} > \mathrm{k}$?

We will leave the relationship between these two problems somewhat open and intuitive, noting only that M1-DECIDE's intractability does not bode well for M1-OPTIMIZE.

Theorem. M1-DECIDE is NP-complete.

To show inclusion in NP, we only need to nondeterministically choose e for any problem instance and verify that it has the requisite $\mathrm{P(e)} \cdot \mathrm{P(f|e)}$ in $\mathrm{O}(m^2)$ time. Next we give separate polynomial-time reductions from two NP-complete problems. Each reduction highlights a different source of complexity.

4.1 Reduction 1 (from Hamilton Circuit Problem)

The Hamilton Circuit Problem (Hopcroft and Ullman, 1979) asks: given a directed graph G with vertices labeled $0, \ldots, n$, does G have a path that visits each vertex exactly once and returns to its starting point?

We transform any Hamilton Circuit instance into an M1-DECIDE instance as follows. First, we create a French vocabulary $f_1, \ldots, f_n$, associating word f_i with vertex i in the graph. We create a slightly larger English vocabulary $e_0, \ldots, e_n$, with e_0 serving as the "boundary" word for source model scoring. Ultimately, we will ask M1-DECIDE to decode the string $f_1 \cdots f_n$.

We create channel model tables as follows:

$$\mathrm{s}(f_j|e_i) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \qquad \epsilon(m|l) = \begin{cases} 1 & \text{if } l = m \\ 0 & \text{otherwise} \end{cases}$$

These tables ensure that any decoding e of $f_1 \cdots f_n$ will contain the n words $e_1, \ldots, e_n$ (in some order).

We now create a source model. For every pair (i, j) such that $0 \leq i, j \leq n$:

$$\mathrm{b}(e_j|e_i) = \begin{cases} 1/\mathrm{n} & \text{if graph G contains an edge} \\ & \text{from vertex } i \text{ to vertex } j \\ 0 & \text{otherwise} \end{cases}$$

Finally, we set k to zero. To solve a Hamilton Circuit problem, we transform it as above (in quadratic time), then invoke M1-DECIDE with inputs b, ϵ, s, k, and the string $f_1 \cdots f_m$.

If M1-DECIDE returns yes, then there must be some string e with both P(e) and P(f|e) nonzero. The channel model lets us conclude that if P(f|e) is nonzero, then e contains the n words $e_1, \ldots, e_n$ in some order. If P(e) is nonzero, then every bigram in e (including the two boundary bigrams involving e_0) has nonzero probability. Because each English word in e corresponds to a unique vertex, we can use the order of words in e to produce an ordering of vertices in G. We append vertex 0 to the beginning and end of this list to produce a Hamilton circuit. The source model construction guarantees an edge between each vertex and the next.

If M1-DECIDE returns no, then we know that every string e includes at least one zero value in the computation of either P(e) or P(f|e). From any proposed Hamilton circuit—i.e., some ordering of vertices in G—we can construct a string e using the same ordering. This e will have P(f|e) = 1 according to the channel model. Therefore, P(e) = 0. By the source model, this can only happen if the proposed "circuit" is actually broken somewhere. So no Hamilton circuit exists.

Figure 4 illustrates the intuitive correspondence between selecting a good word order and finding a Hamilton circuit. We note that (Brew, 1992) discusses the NP-completeness of a related problem, that of finding some permutation of a string that is acceptable to a given context-free grammar. Both of these results deal with decision problems. Returning briefly to optimization, we recall another circuit task called the Traveling Salesman Problem. It introduces edge costs d_{ij} and seeks a minimum-cost circuit. By viewing these edge costs as log-probabilities, we can cast the Traveling Salesman Problem as one of optimizing P(e), that is, of finding the best source word order in Model 1 decoding.

4.2 Reduction 2 (from Minimum Set Cover Problem)

The Minimum Set Cover Problem (Garey and Johnson, 1979) asks: given a collection C of subsets of finite set S, and integer n, does C contain a cover for S of size $\leq n$, i.e., a subcollection whose union is S? We now transform any instance of Minimum Set Cover into an instance of M1-DECIDE, using polynomial time.

This time, we assume a rather neutral source model in which all strings of a given length are equally likely, but we construct a more complex channel.

We first create a source word e_i for each subset in C, and let g_i be the size of that subset. We create a table $\mathrm{b}(e_i|e_j)$ with values set uniformly to the reciprocal of the source vocabulary size (i.e., the number of subsets in C).

Assuming S has m elements, we next create target words $f_1, \ldots, f_m$ corresponding to each of those elements, and set up channel model tables as follows:

$$\mathrm{s}(f_j|e_i) = \begin{cases} 1/g_i & \text{if the element in S corresponding to } f_j \\ & \text{is also in the subset corresponding to } e_i \\ 0 & \text{otherwise} \end{cases}$$

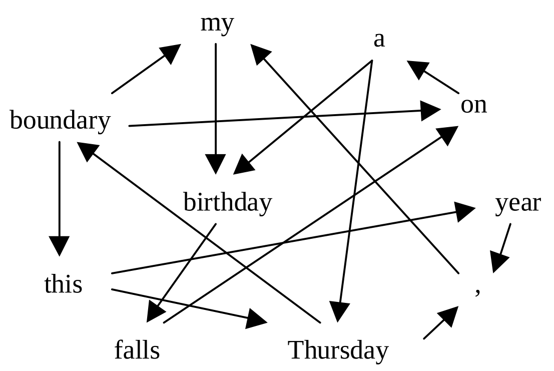

Figure 4
Selecting a Good Source Word Order is Like Solving the Hamilton Circuit Problem. If we assume that the channel model offers deterministic, word-for-word translations, then the bigram source model takes responsibility for ordering them. Some word pairs in the source language may be illegal. In that case, finding a legal word ordering is like finding a complete circuit in a graph. (In the graph shown above, a sample circuit is *boundary* $\rightarrow$ *this* $\rightarrow$ *year* $\rightarrow$ *comma* $\rightarrow$ *my* $\rightarrow$ *birthday* $\rightarrow$ *falls* $\rightarrow$ *on* $\rightarrow$ *a* $\rightarrow$ *Thursday* $\rightarrow$ *boundary*). If word pairs have probabilities attached to them, then word ordering resembles the finding the least-cost circuit, also known as the Traveling Salesman Problem.

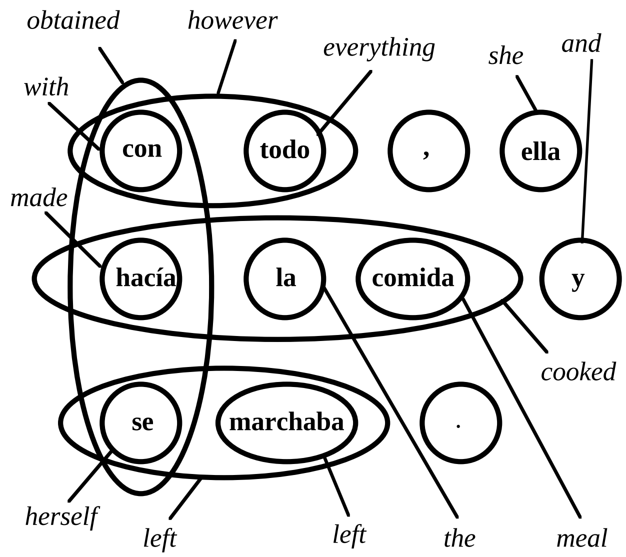

Figure 5
Selecting a Concise Set of Source Words is Like Solving the Minimum Set Cover Problem. A channel model with overlapping, one-to-many dictionary entries will typically license many decodings. The source model may prefer short decodings over long ones. Searching for a decoding of length $\leq n$ is difficult, resembling the problem of covering a finite set with a small collection of subsets. In the example shown above, the smallest acceptable set of source words is $\{$*and*, *cooked*, *however*, *left*, *comma*, *period*$\}$.

$$\epsilon(m|l) = \begin{cases} 1 & \text{if } l \leq n \\ 0 & \text{otherwise} \end{cases} \qquad \epsilon(m+1|l) = \begin{cases} 1 & \text{if } l > n \\ 0 & \text{otherwise} \end{cases}$$

Finally, we set k to zero. This completes the reduction. To solve an instance of Minimum Set Cover in polynomial time, we transform it as above, then call M1-DECIDE with inputs b, ϵ, s, k, and the words $f_1, \ldots, f_m$ in any order.

If M1-DECIDE returns yes, then some decoding e with P(e) $\cdot$ P(f|e) > 0 must exist. We know that e must contain n or fewer words—otherwise P(f|e) $= 0$ by the ϵ table. Furthermore, the s table tells us that every word f_j is covered by at least one English word in e. Through the one-to-one correspondence between elements of e and C, we produce a set cover of size $\leq n$ for S.

Likewise, if M1-DECIDE returns no, then all decodings have P(e) $\cdot$ P(f|e) $= 0$. Because there are no zeroes in the source table b, every e has P(f|e) $= 0$. To account for this zero, either (1) the length of e exceeds n, or (2) some f_j is left uncovered by the words in e. Because source words cover target words in exactly the same fashion as elements of C cover S, we conclude that there is no set cover of size $\leq n$ for S.

Figure 5 illustrates the intuitive correspondence between source word selection and minimum set covering.

5. Discussion

The two proofs in the last section point up separate factors in MT decoding complexity. One is word-order selection. But even if any word order will do, there is still the problem of picking a concise decoding in the face of overlapping bilingual dictionary entries. The former is more closely tied to the source model, and the latter to the channel model, though the complexity arises from the interaction of the two.

We should note that Model 1 is an intentionally simple translation model, one whose primary purpose in machine translation has been to allow bootstrapping into more complex translation models (IBM's Models 2-5). It is easy to show that the intractability results also apply to these stronger "fertility/distortion" models; we assign zero probability to fertilities other than 1, and we set up uniform distortion tables.

Simple translation models like Model 1 find more direct use in other applications (e.g., lexicon construction (Wu and Xia, 1994), idiom detection (Melamed, 1997), psychological norms (Wettler and Rapp, 1993), and cross-language information retrieval), so their computational properties are of wider interest.

The proofs we presented are also based on a worst-case analysis. Real s, ϵ, and b tables may have properties that permit faster optimal decoding than the artificial tables constructed above. It is also possible to devise approximation algorithms like those devised for other NP-complete problems. To the extent that word ordering is like solving the Traveling Salesman Problem, it is encouraging substantial progress continues to be made on traveling-salesman algorithms. For example, (Pemberton and Zhang, 1996) and others describe methods for getting within two percent of the optimal tour, while (Applegate et al., 1998) demonstrate an optimal tour of 13,509 US cities. (The latter experiment relied on things like distance symmetry and the triangle inequality constraint, however, which do not hold in word ordering.) We also now have a better understanding of why some traveling-salesman instances are easy to solve optimally, and why some are hard (Zhang and Korf, 1996). So far, statistical translation research has either opted for heuristic beam-search algorithms (Brown et al., 1990; Wang and Waibel, 1997) or different channel models. For example, (Tillmann et al., 1997) avoid bag generation by preprocessing bilingual texts to remove word-order differences, while (Wu, 1996; Wang and Waibel, 1998) adopt channels that eliminate syntactically unlikely alignments.

Finally, expensive decoding also suggests expensive training from unannotated (monolingual) texts, which presents a challenging bottleneck for extending statistical machine translation to language pairs and domains where large bilingual corpora do not exist.

References

David Applegate, Robert Bixby, Vasek Chvatal, and William Cook. 1998. On the solution of traveling salesman problems. *Doc. Math. J. DMV Extra Volume ICM III*, page 645.

L. E. Baum. 1972. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process. *Inequalities*, 3.

Chris Brew. 1992. Letting the cat out of the bag: Generation for shake-and-bake MT. In *Proceedings of the International Conference on Computational Linguistics (COLING).*

Peter Brown, John Cocke, Stephen Della Pietra, Vincent Della Pietra, Fredrick Jelinek, John Lafferty, Robert Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2).

Peter Brown, Stephen Della-Pietra, Vincent Della-Pietra, and Robert Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2).

Ken Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the Conference on Applied Natural Language Processing.*

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(B):1–38.

Michael Garey and David Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Co, New York.

John Hopcroft and Jeffrey Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, MA.

Fred Jelinek. 1998. *Statistical Methods for Speech Recognition.* MIT Press, Cambridge, MIT.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4).

I. Dan Melamed. 1997. Automatic discovery of non-compositional compounds. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing.*

Bernard Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2).

Joseph Pemberton and Weixiong Zhang. 1996. Epsilon-transformation: Exploiting phase transitions to solve combinatorial optimization problems. *Artificial Intelligence*, 81.

Michel Simard. 1998. Automatic insertion of accents in French text. In *Proc. Third Conference on Empirical Methods in Natural Language Processing.*

C. Tillmann, S. Vogel, H. Ney, and A. Zubiaga. 1997. A DP-based search using monotone alignments in statistical translation. In *Proc. ACL.*

A. J. Viterbi. 1967. Error bounds for convolution codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269.

Y. Wang and A. Waibel. 1997. Decoding algorithm in statistical machine translation. In *Proc. ACL.*

Ye-Yi Wang and Alex Waibel. 1998. Modeling with structures in statistical machine translation. In *Proceedings of the COLING/ACL Conference.*

Manfred Wettler and Reinhard Rapp. 1993. Computation of word associations based on the co-occurrences of words in large corpora. In *Proceedings of the Workshop on Very Large Corpora (WVLC).*

Dekai Wu and Xuanyin Xia. 1994. Learning in English-Chinese lexicon from a parallel corpus. In *Proceedings of the First Conference of the Association for Machine Translation in the Americas.*

D. Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *Proc. ACL.*

Weixiong Zhang and Richard Korf. 1996. A study of complexity transitions on the asymmetric traveling salesman problem. *Artificial Intelligence*, 81.