Introduction
We will provide an overview of the paper by Germann, giving a description of the methods employed. The key claim is that significant speedup in translation is possible without a significant reduction in translation accuracy, using a variation on a currently popular statistical approach to translation.

Goals
The author’s goal is to present an improvement to a popular statistical approach to machine translation. They claim practical reduction from O(n^6) to O(n) or O(k*n^2) where k is so small for most n that k*n^2 approaches the values for n. They do so while minimizing a loss in the quality of translations.

Core Problem
As explained in the Background section, finding the translation candidate S that maximizes pr(S|T) for a given T is NP-complete, and so guaranteed optimal decoders are not used, and approximate solutions are used instead. Such approximate solutions result in 5% to 10% error but return answers in reasonable time, and are within a few percentage points of the optimal algorithm performance.

Approach
The searching for a translation that maximizes the probability that it is a correct translation is NP-complete. The Germann used several methods to address the key problem of NP-completeness:

- integrating (hypothesis evaluation) into hypothesis creation
- tiling improvements over the translation hypothesis at the end of each search iteration
- imposing restrictions on the amount of word reordering during decoding

Background
Most work in statistical machine translation use word replacement models and conventions laid out in Brown et al. (1993). The Models are referred to as IBM models 1 to 5.

The IBM model views the translation problem as decoding a message from a noisy channel that undergoes a set of standard transformations. The goal is to undo these transformations.

The Brown model assumes that every sentence(T) in one language is a possible translation for any sentence(S) in another, a cross product. They can then assign a probability pr(T|S) to each pair (S,T), the probability a translator would generate T as a translation of S. Pr(T|S)
should be very small for non-translations and very high for commonly accepted translations.

The goal then is stated as: Given $T$ in the target language, find the sentence $S$ from which the translation produced $T$. Error is minimized by picking the $S$ most probable given $T$. So they choose $S$ to maximize $Pr(S|T)$. Using Bayes,

$$Pr(S|T) = \frac{(Pr(S)*Pr(T|S))}{Pr(T)}$$

Since the denominator does not depend on $S$, one can just choose the $S$ to maximize $Pr(S)*Pr(T|S)$

$Pr(S) =$ the language model probability of $S$
$Pr(T|S) =$ translation probability of $T$ given $S$

Brown suggests viewing $Pr(T|S)$ as suggesting words in the source that may have produced the words we observed, and $Pr(S)$ as suggesting an order to place the source words.

So one needs a method of computing the source language probabilities $Pr(S)$, the translation probabilities $Pr(T|S)$ and a way to search among possible $S$ for the one that maximizes $Pr(S)*Pr(T|S)$

In computing the language and translation models each version of IBM considers different parameters:

- IBM 1: does not consider positional information and sentence length
- IBM 2: considers sentence length and word position
- IBM 3, 4, 5: fertility in translation

An example of entries for $Pr(T|S)$ for French-English:

- people (gens, 0.25) (les, 0.16) (personnes, 0.1) …
- years (ans, 0.38) (années, 0.31) (depuis, 0.12), …

**Details of the Germann method**

Moving from Brown to Germann notation requires a minor change in notation:

$S \Rightarrow e$ (Source – English)
$T \Rightarrow f$ (Translation – Foreign)

$p(e)$ is computed using an n-gram language model

The basic process is:

- for each English word $e_i$, a fertility $n_i$ is chosen
  - $n_i$ is the fertility of $e_i$
  - fertility is the number of words $e_i$ will translate into

- Each word $e_i$ is replaced by $n_i$ foreign words

- The linear order of the foreign words are rearranged

- $n_0$ spurious words are inserted into the foreign text
  - $n_0$ depends on length $l$ of the English string

Each transformation has a probability called the **translation model probability** and is the product of the individual probabilities of each change.

**Alignment probability** = 

$$(\text{translation model probability}) * (\text{language model probability of } E)$$

Computing the alignment probability of a whole alignment is linear in
Germann shows how to compute the alignment probability incrementally and thereby reduce the evaluation cost to constant time.

**Decoding**

Greedy Decoding is a hill climbing algorithm. It does not work one word at a time.

*start with a complete gloss of the input sentence
* align each input word f with word e that maximizes the inverse probability \( t(e|f) \) (or in Brown pr(S|T))
  - uses \( t(f|e) \)

The decoder tries out changes to the alignment:

- translation of a word
- insertion of NULL words
- Removing NULL words
- Joining e words
- Swapping E regions

For each "search iteration" the algorithm evaluates each possible change.

G1 - considers one operation at a time
G2 - considers up to 2 operations, or inserts one NULL word + word translation change before evaluation

At end of each iteration, decoder locks in the changes that gives the biggest gain in alignment probability then repeats search iteration, until no more improvements can be made.

The “greedy” part of the Germann decoder is shown in the modification of the operations used in the search process:

- CHANGE word translation: only examines the top 10 possible translations, making CHANGE a \( O(10n) \) operation.
- INSERT null word: limited to once for each English word, making it \( O(n) \)
- ERASE null word: again \( O(n) \)
- JOIN two English words: given [...e_saved... e_doomed...], removes e_doomed and all f – words linked to e_doomed are attached to e_saved. By checking \( t(f|e_{saved}) \), which for the most part will be zero, allows quick pruning, changing a \( O(n^2) \) process to a much smaller though unknown value.
- SWAP two regions in the English string: reduced by limiting the size of the regions (S) and distance (d) to limit the possible swaps to \( d*s*s \)

The key optimization occurs from maintaining a local context and thus reducing the search space to optimizing islands in the search space. By searching for all changes in a local context and keeping track of the best ones, then hill-climbing them, Germann is able to reduce a \( O(n^4) \) problem to a chain of smaller \( O(n^3) \) problems with small coefficients. This gives the piratical behavior of being almost linear.

**Comments**

Primary reductions comes from changing the swapping from \( O(n^4) \) to \( d*s*s \). In an evaluation processing 100 short Chinese news stories total processing time went from 440 CPU hours to 40 minutes. Swapping accounted for 98% of the search steps, but gave only 5% of the improvements. Limiting swapping/reordering is a common translation optimization.
The author points out that since BLEU is based on n-grams it is not possible to measure the effects of long range swapping.

Overall, the improvements to the decoder make it possible to do multiple searches for each sentence, giving an overall better decoder.

The author proposes using multiple randomized searches per sentence. To me this brings to mind using a genetic algorithm, or grammatical evolution system instead of a purely randomized search process. This would allow continued use of the hill climbing paradigm, while allowing imposition of resource limitations on the processing of each sentence (say give the best 10 translations in 2 seconds).

Questions

- How valid are the assumptions on which they base their complexity estimate?
- Do the restrictions over restrict?
- Are they "Turing complete" for the space?
- Would other search processes be appropriate?
- Would there be any benefit from attempting to learn optimal operations?

References


Software

http://www.isi.edu/natural-language/people/germann/software/ReWrite-Decoder/index.html