AN APPROXIMATION ALGORITHM FOR WORD-REPLACEMENT USING A BI-GRAM LANGUAGE MODEL

Jing He *Hongyu Liang*

Tsinghua University
Institute for Theoretical Computer Science
Beijing, China

ABSTRACT

This paper presents an approximation algorithm for word-replacement under a bi-gram language model. Words replacement is an key step in the decoding part of statistical machine translation. However, the word or phrase replacement step is often done at the same time with the target language generating process in machine translation while our algorithm focus on the special replacement model without the target language generating process. We firstly make a reduction from the famous NP-Complete problem *Hamiltonian Path Problem* to the word-replacement problem. Then we apply the approximation algorithm for Hamiltonian Path on the word-replacement problem, which gives us a good performance in the designed experiment.

Index Terms— word-replacement, statistical machine translation, NP-hard, Hamiltonian Path Problem, approximation algorithm

1. INTRODUCTION

Statistical method is widely used in natural language processing such as machine translation and part-of-speech tagging. The most usually used model in statistical machine translation is the source-channel model that is built by Brown et al.(1993)[1]. They assumed that in translation between English and French, English strings are generated according to some stochastic process and then transformed stochastically into French strings. Thus to translate French to English, it's needed to search for an English source string that is mostly likely according to the English language model[7] and the channel model. This kind of translation is called decoding. And usually a decoding process in statistical machine translation is combined by two sub-process[2][6]: generating the words or phrases of the target language and deciding the right order of the words or phrases to get a right target language sentence. For some language pairs, such as English and Japanese, the word-replacing problem is really hard to solve as the target word order differs significantly from the source word order and we get little information about the target word order from the source sentence.

Follow the result by Kevin[4] who studied the decoding complexity in word-replacement, this paper focuses on the word-replacement sub-process in machine translation with the source language model. We assume that the words of the target language is generated. However, as the grammar is totally different of the two languages(for example, English and Japanese), we need to replace all target words. We prove that in bi-gram language model[7], the problem is NP-hard using a reduction from the famous NP-complete problem[3] *Hamiltonian Path Problem*. And then we apply an approximation algorithm for finding out the solution of *Hamiltonian Path Problem* to our model and get a considerably good result from the data set of 704 scrambling English sentences(from http://www.nlp.org.cn).

Although, our experiment is done on the word-based replacement of a English sentence, however, our reduction and solving method can also be applied in the statistical machine translation whose working unit is phrase[6] if we replace the word "words" by "phrases" in our model description, reduction proof and algorithm process.

2. THE WORD-REPLACEMENT MODEL DESCRIPTION

In a word-replacement model, we have some (English) words together with a bi-gram source model which figures the dependency of the occurrence of words. Then, given any disordered sentence, we rearrange the words to meet the bi-gram source mode best so that we can approximately reconstruct the original sentence. In a more theoretical sense, we can formalize this idea into a search problem as follows.

Word Replacement Problem
Input:

1. A set of k words $U = \{w_i \mid 1 \leq i \leq k\}$.
2. A bi-gram source model with $k^2 + k$ parameters:
   $lm(w_i \mid w_j), 1 \leq i, j \leq n$;
   $lm(w_i \mid boundary), 1 \leq i \leq k$.

This work was supported in part by the National Natural Science Foundation of China Grant 60553001, and the National Basic Research Program of China Grant 2007CB807900,2007CB807901.

978-1-4244-5076-3/09/$26.00 ©2009 IEEE

3. A set of words $S \subseteq U$ without any order.

Output: An ordered tuple $T = (s_{i_1}, s_{i_2}, \ldots, s_{i_n})$ such that

1. $(i_1, i_2, \ldots, i_n)$ is a permutation of $\{1, 2, \ldots, n\}$.
2. The value

$$P(T) = lm(w_1 \mid boundary) \cdot \prod_{i=2}^{n} lm(w_i \mid w_{i-1})$$

achieves its maximum.

3. REDUCTION

We reduce the *Hamiltonian Path Problem* to the *Word Replacement Problem*[4]. It is well known that the search version of the former problem is **NP**-hard[3].

Hamiltonian Path Problem (Search version)

Input:

1. A directed graph $G = (V, E)$.
2. A weight function $f : E \rightarrow \mathbf{R}$

Output: A Hamiltonian path with minimum weight, where the weight of a path is defined as the sum of weights of all the edges in that path.

Now we describe the reduction. Given a digraph $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$, and a weight function f, we construct a set of n words $U = \{w_1, \ldots, w_n\}$ together with the bi-gram source model as follows:

1. $\forall 1 \leq i \leq n,\ lm(w_i \mid boundary) = 1$.
2. $\forall 1 \leq i, j \leq n,\ lm(w_i \mid w_j) = 2^{-f((w_j, w_i))}$.

Let the disordered set S be U itself. Now, we have constructed an instance of *Word Replacement Problem*. It suffices to show that if a sentence $T = (w_{i_1}, \ldots, w_{i_k})$ achieves the maximum probability according to our previous definition, then the corresponding Hamiltonian path $p = (v_{i_1}, \ldots, v_{i_n})$ has the minimum weight. Note that

$$\begin{aligned} P(T) &= lm(w_{i_1} \mid boundary) \cdot \prod_{i=2}^{n} lm(w_{i_j} \mid w_{i_{j-1}}) \\ &= \prod_{i=2}^{n} 2^{-f((w_{i_{j-1}}, w_{i_j}))} \\ &= 2^{-\sum_{j=2}^{n} f((v_{i_{j-1}}, v_{i_j}))} \\ &= 2^{-f(p)} \end{aligned}$$

So it follows straightforwardly from the fact that $f(x) = 2^{-x}$ is monotone decreasing. We have proved that the *Word Replacement Problem* is **NP**-hard.

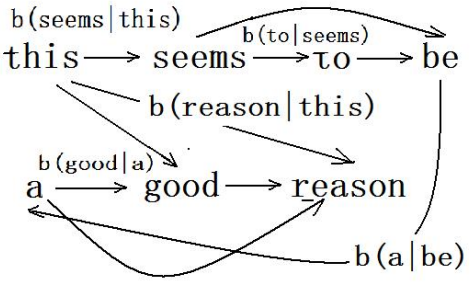

Fig 1: The idea of reduction

4. THE REPETITIVE NEAREST NEIGHBOR ALGORITHM FOR OUR PROBLEM

To search for the minimum Hamiltonian path, we can apply the exhaustive search algorithm that requires O($k^2 n!$) time, where k is total number of the English words and n the size of the graph constructed by the reduction. The complexity to find out the correct order grows exponentially. Due to our experimental result, when the number of words in a sentences is larger than 10, the running time of the program to search for a best ordered sentence is totally intolerable.

We introduce the *repetitive nearest neighbor algorithm* into our program to efficiently search for a "nearly-optimal", or an approximating solution. There are counter-examples showing that the algorithm is not necessarily optimal.

4.1. The nearest neighbor algorithm

The nearest neighbor algorithm runs as follows: 1. Start from a certain vertex.

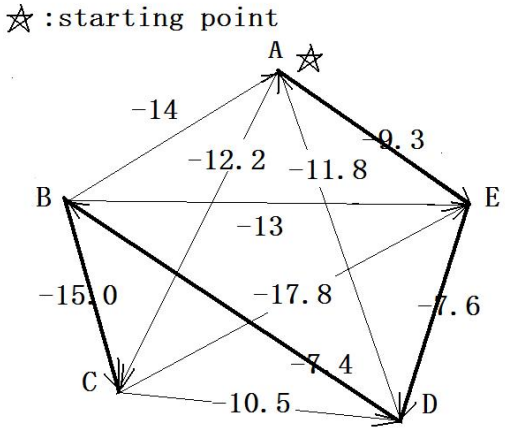

Fig 2: The nearest neighbor algorithm

2. Whenever you are at a vertex, pick the nearest unvisited vertex that is a neighbor of the current vertex. In case of a tie,

randomly pick one.
3. When all the vertex are visited, return the total path weight.
In figure(2), we start from A, pick E that is nearest to A, pick D that is nearest to E, and then follows by B and C. The approximate path is marked with the boldface.

Algorithm 1 The Nearest Neighbor Algorithm

Input: start point i, current path p
$CurrentPoint \leftarrow i$
set i to be visited
$max \leftarrow -100000$ (just a sufficiently large value)
for $j = 1$ to n **do**
  if j is unvisited AND the edge (i, j) is contained in G AND $max < weight(i, j)$ **then**
    $CurrentPoint \leftarrow j$
    $max \leftarrow weight(i, j)$
  end if
end for
set j to be visited
$p \leftarrow p \cup (i, CurrentPoint)$
if all nodes have been visited **then**
  return p
else
  $i \leftarrow CurrentPoint$
  Recursively run the algorithm with input (i, p).
end if

4.2. The repetitive nearest neighbor algorithm

The repetitive nearest neighbor algorithm starts from every node of the graph and applies the nearest neighbor algorithm. And then return the best of all the hamiltonian path starting differently as follows:

Algorithm 2 The Repetitive Nearest Neighbor Algorithm

$max \leftarrow -10000000$ (a sufficiently small number)
$maxPath \leftarrow \emptyset$
for $i = 1$ to n **do**
  Apply the **Nearest Neighbor Algorithm** with input $(i, \emptyset)$ to get a result p
  if $max < weight(p)$ **then**
    $maxPath \leftarrow p$
    $max \leftarrow weight(p)$
  end if
end for
return $maxPath$

5. EXPERIMENT

5.1. The bi-gram language model training

In order to build a reasonable bi-gram language model[7] to complete the experiment, we download the 3rd version of the Europarl corpus[5] which is extracted from the proceedings of the European Parliament. This data set is usually used as a base material in statistical machine translation contest or other research projects involved European language. And the English part is used to train an bi-gram English language model. There are about 0.307 million English sentences in the material. Thus the bi-gram language model built by SRILM[8] can reflect the properties of the English language.

5.2. The re-ordering experiment

How to test the accuracy of the algorithm is an key problem in our research as we have no standard database to test the accuracy. The replacement of words in sentences is a subprocess in statistical machine translation, especially in the decoding step. However, all the existing testing method and standard are designed to meet the test of the accuracy of the translation result but not the single subprocess of words replacement.

Thus, we designed a testing principle and method for our own purpose. First, we choose 704 English sentences from the data set of 1500 English-Chinese sentence pairs(from http://www.nlp.org.cn) as the standard answer as all the sentences make sense. Then, we randomly generate a words permutation to transfer the original sentences to the scrambling ones. For example, the original sentences is "sometimes you are overly frank" that has 5 words. We generate a permutation of length 5 $\{1\ 3\ 2\ 5\ 4\}$ to get a scrambling new sentence "sometimes are you frank overly" while the first one is used as an standard answer.

To test the distance between our answer and the standard sentence seems to be the same problem of determining the distance between two ordered array, while the inversion pairs that is used to measure the distance between two ordered array is brought into our experiments.
A pair of integers (i,j) is an inversion pair of some permutation σ

$$if\ \ i < j\ \ but\ \ \sigma(i)\ >\ \sigma(j) \tag{1}$$

For instance, from the correct sentence is "sometimes you are overly frank" we get a total order of the words that "sometimes" = 1, "you" = 2, "are" = 3, "overly" = 4, "frank" = 5. The output sentence by our program is "sometimes you are frank overly" = $\{1\ 2\ 3\ 5\ 4\}$ which has only one inversion pair (5,4).

On this way we can calculate the number of inversion pairs to measure the degree of disordering. By comparing the number of inversion pairs in the randomly scrambling sentences and the partially ordered sentences by our algorithm, we can see the effect of words replacement.

By contrast, we both test the approximate algorithm and the non-efficiency exhaustive search Hamiltonian algorithm to get an exact sentences with the maximum language model score.

The result of the two algorithms are listed in the table. To illustrated the result of our experiments, we randomly picked 15 sentences to illustrate the result of our approximate algorithm. The chart below shows the number of inversion pairs in

704 test cases	approximate algorithm	exact algorithm
better sentence percentage	70.74%	74.71%
total inversion pairs drop	1366	1852
average inversion pairs drop	1.94	2.63

Table 1: Inversion pairs drop

the randomly disordered sentences pairs and the output sentences of the approximately minimum Hamiltonian path. By our reduction, along the points in the minimum Hamiltonian path we can reconstruct the sentence with a maximum language model score.

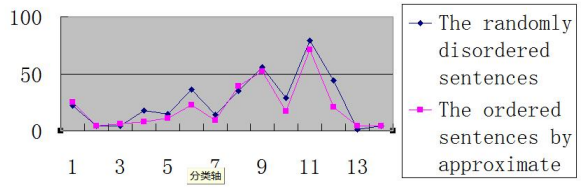

Fig 3: Approximate algorithm
(the number of inversion pairs in contrast)

The non-efficiency exhaustive search Hamiltonian algorithm performs better as the chart shown below

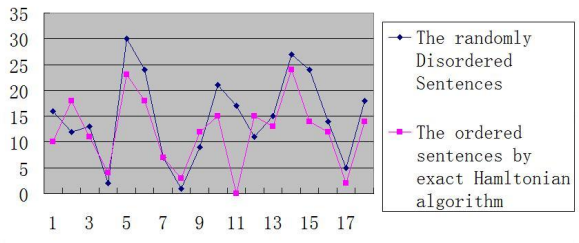

Fig 4: Exact algorithm
(the number of inversion pairs in contrast)

The exact algorithm performs better than the approximate algorithm, as it searches for the maximum Hamiltonian path.

We also randomly choose some example sentences to show the words replacement result.

6. CONCLUSIONS

In this paper, we study the problem of word replacement in decoding process in statistical machines translation. We implement the existing approximate algorithm for Hamiltonian path under the model and after experiment, the algorithm is proved to be reasonable and performs well in re-ordering the words to form a logical sentences.

However, the language model we used is bi-gram model. Which is more easier than the popular tri-gram model that better evaluate the sentence itself. In future, we want to apply the tri-gram model into the words replacement sentences. What's more, our algorithm and experiments are done independently from the decoding process, such as the target language words or phrases generating. We are about to combined

case	The scrambling sentences	The correct sentences
#1	had have them somebody must	somebody must have had them
#2	is pact a between this us	this is a pact between us
#3	paddocks fields their are mere	their fields are mere paddocks
#4	was he great she that knew pain in	she knew that he was in great pain
#5	party the rescue arrived had	the rescue party had arrived
case	The result of the approximate algorithm	The result of the exact algorithm
#1	somebody must have had them	somebody must have had them
#2	pact is a us this between	between us this is a pact
#3	their fields are mere paddocks	paddocks their fields are mere
#4	was he great she that knew pain in	she knew that he was in great pain
#5	the rescue had party arrived	rescue the party had arrived

Table 2: Examples

the two part: predicting the collection of words and deciding the correct order of the target sentences together to design and test new decoding algorithms.

7. REFERENCES

[1] Peter F. Brown, Stephen Della-Pietra, Vincent Della-Pietra, and Robert Mercer "The mathematics of statistical machine translation: Parameter estimation," in *Computational Linguistics*. 19(2): 263-311, 1993.

[2] Pi-Chuan Chang, Kristina Toutanova, "A Discriminative Syntactic Word Order Model for Machine Translation," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. pp. 9-16, Prague, Czech Republic, June 2007.

[3] Richard M. Karp, "Reducibility Among Combinatorial Problems," in *Complexity of Computer Computations*. pp. 85-103, New York: Plenum, 1972.

[4] Kevin Knight, "Decoding Complexity in Word-Replacement Translation Models," in *Computational Linguistics*. 25(4): 607-615, 1999.

[5] Phukuoo Koehn, "Europarl: A Parallel Corpus for Statistical Machine Translation," MT Summit, 2005.

[6] Philipp Koehn, Franz Josef Och, and Daniel Marcu 2003. "Statistical Phrases-Based Translation," HLT/NAACL, 2003.

[7] J M. Ponte and W B. Croft, "A Language Model Approach to Information retrieval," Research and Development in Information Retrieval: 275-281, 1998.

[8] Andreas Stolcks, "SRILM – An Extensible Language Modeling Toolkit," in *Proceedings of the International Conference on Spoken Language Processing*. vol. 2, pp. 901-904, Denver, 2002.