Volume 3, No. 1, January 2012
Journal of Global Research in Computer Science
ISSN-2229-371X
RESEARCH PAPER
Available Online at www.jgrcs.info

DESIGN OF PHRASE-BASED DECODER FOR ENGLISH-TO-SANSKRIT TRANSLATION

Mr.Sandeep Warhade*[1] and Mr.Prakash R.Devale[2]

[1*]Research Scholar, Deptt. of IT, Bharati Vidyapeeth Deemed University College of Engineering, Pune-India.
sandeepwarhade@gmail.com

[2]Professor & Head, IT Deptt., Bharati Vidyapeeth Deemed University College of Engineering, Pune-India.
prdevale@bvucoep.edu.in

***Abstract*—**This paper describes the Phrase-Based Statistical Machine Translation Decoder. Our goal is to improve the translation quality by enhancing the translation table and by preprocessing the source language text. research. We discuss the major design objective for the decoder, its performance relative to other SMT decoders, and the steps we are taking to ensure that its success will continue.

Keywords- Phrase-based Statistical Machine Translation, English-to-Sanskrit Translation.

INTRODUCTION

Phrase-based translation has been one of the major advances in statistical machine translation [2] in recent years and is currently one of the techniques which can claim to be state-of-the-art in machine translation. Phrase-based models are a development of the word based models as exemplified by the [2]. In phrase-based translation, contiguous segments of words in the input sentence are mapped to contiguous segments of words in the output sentence.

In SMT, we are given a source language sentence, s, which is to be translated into a target language sentence, t. The goal of machine translation, t^, is to find the translation, t^ , which is defined as:

$$\hat{t} = \arg\max_t p(t \mid s)$$

where p (t | s) is the probability model. The argmax implies a search for the best translation t^ in the space of possible translations t. This search is the task of the decoder, which we will concentrate on in this paper.

There have been numerous implementations of phrase-based decoders for SMT prior to our work. Early systems such as the Alignment Template System (ATS) [3] and Pharaoh [4] were widely used and accepted by the research community. ATS is perhaps the crossover system, in that word classes were translated as phrases but the surface words were translated word by word. Pharaoh substituted the word classes with surface words, thereby discarding the use of word classes in decoding altogether.

Pharaoh was released in 2003 as a pre compiled binary with documentation. This severely limited the extent to which other researchers can study and enhance the decoder. Without access to the decoder source code research was generally restricted to altering the input, augmenting it with extra information, or modifying the output or re-ranking the n-best list output.

The main contribution of this paper is to show how we have created an extensible decoder, has acceptable run time performance compared to similar systems, and the ease of use and development that has made it the preferred choice for researchers looking for a phrase-based SMT decoder.

SMT SYSTEM FRAMEWORK

Phrase-based statistical machine translation approaches continue to dominate the field of machine translation. The translation service makes use of state-of-the-art phrase-based SMT systems within the framework of feature-based exponential models containing the following features:

a. Phrase translation probability
b. Inverse phrase translation probability
c. Lexical weighting probability
d. Inverse lexical weighting probability
e. Phrase penalty
f. Language model probability
g. Simple distance-based distortion model
h. Word penalty

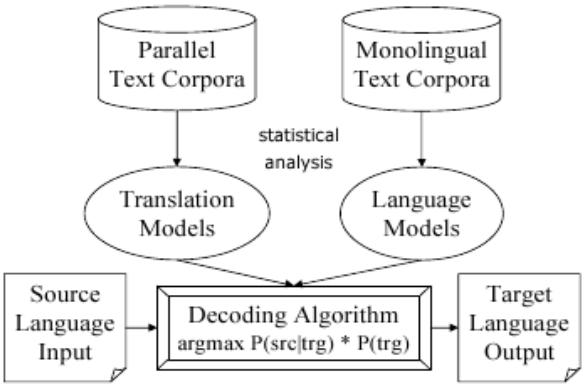

Figure 1: SMT Framework

The basic framework within which all the MT systems were constructed is shown in Figure
Translation examples from the respective bilingual text corpus are aligned in order to extract phrasal equivalences and to calculate the bilingual feature probabilities. Monolingual features like the language model probability are trained on mono lingual text corpora of the target language whereby standard word alignment and language modeling tools were used.

Parallel Corpora:

SMT treats translation as a machine learning problem. This means that we apply a learning algorithm to a large body of previously translated text, known variously as a parallel corpus, parallel text, bitext, or multitext. The learner is then able translate previously unseen sentences. With an SMT toolkit and enough enough parallel text, we can build an MT system for a new language pair within a very short period of time. be built for a wide variety of language pairs within similar time frames. The accuracy of these systems depends crucially on the quantity, quality, and domain of the data, but there are many tasks for which even poor translation is useful [5].

Language Model:

Statistical language modeling is the science (and often art) of building models that estimate the prior probabilities of word strings. Language modeling has many applications in natural language technology and other areas where sequences of discrete objects play a role, with prominent roles in speech recognition and natural language tagging (including specialized tasks such as part-of-speech tagging, word and sentence segmentation, and shallow parsing). As pointed out in [6], the main techniques for effective language modeling have been known for at least a decade, although one suspects that important advances are possible, and indeed needed, to bring about significant breakthroughs in the application areas cited above—such breakthroughs just have been very hard to come by [7,8].

Translational Model:

Translational models allow us to enumerate possible structural relationships between pairs of strings. However, even within the constraints of a strict model, the ambiguity of natural language results in a very large number of possible target sentences for any input source sentence. Our translation system needs a mechanism to choose between them.

This mechanism comes from modeling: parameterization. We design a function that allows us to assign a real-valued score to any pair of source and target sentences. The general forms of these models are similar to those in other machine learning problems. There is a vast number of approaches;, for more detail, the reader is referred to a general text on machine learning, such as [9].

Decoding Model:

Now that we have a model and estimates for all of our parameters, we can translate new input sentences. This is called decoding. We call this the decision rule.

The phrase-based decoder we developed for purpose of comparing different phrase-based translation models employs a beam search algorithm, similar to the one by [10]. The Sanskrit output sentence is generated left to right in form of partial translations (or hypotheses).

We start with an initial empty hypothesis. A new hypothesis is expanded from an existing hypothesis by the translation of a phrase as follows: A sequence of untranslated foreign words and a possible English phrase translation for them is selected. The English phrase is attached to the existing English output sequence. The foreign words are marked as translated and the probability cost of the hypothesis is updated.

The cheapest (highest probability) final hypothesis with no untranslated foreign words is the output of the search. The hypotheses are stored in stacks. The stack contains all hypotheses in which foreign words have been translated. We recombine search hypotheses as done by [11]. While this reduces the number of hypotheses stored in each stack somewhat, stack size is exponential with respect to input sentence length. This makes an exhaustive search impractical.

Thus, we prune out weak hypotheses based on the cost they incurred so far and a future cost estimate. For each stack, we only keep a beam of the best hypotheses. Since the future cost estimate is not perfect, this leads to search errors. Our future cost estimate takes into account the estimated phrase translation cost, but not the expected distortion cost.

We compute this estimate as follows: For each possible phrase translation anywhere in the sentence (we call it a translation option), we multiply its phrase translation probability with the language model probability for the generated English phrase. As language model probability we use the unigram probability for the first word, the bigram probability for the second, and the trigram probability for all following words.

Given the costs for the translation options, we can compute the estimated future cost for any sequence of consecutive foreign words by dynamic programming. Note that this is only possible, since we ignore distortion costs. Since there are only such sequences for a foreign input sentence of length , we can pre-compute these cost estimates beforehand and store them in a table.

During translation, future costs for uncovered foreign words can be quickly computed by consulting this table. If a hypothesis has broken sequences of untranslated foreign words, we look up the cost for each sequence and take the product of their costs.

The beam size, e.g. the maximum number of hypotheses in each stack, is fixed to a certain number. The number of translation options is linear with the sentence length. Hence, the time complexity of the beam search is quadratic with sentence length, and linear with the beam size.

PHRASE-BASED SMT SYSTEM

We decided to develop the decoder as a English to Sanskrit translator. Java is chooses for it's rich library and multi-platform support would have been useful.

The decoder is fully compatible with Pharaoh 1.2 in the algorithms that are implemented, input files (configuration file, translation table, language models) and command line.

Algorithm used for decoding:

Here the algorithm we described used for our beam search. For each number of foreign word covered, a hypotheses stack is created. The initial hypothesis is placed in the stack for hypothesis with no foreign words covered. Starting with this hypothesis, new hypothesis are generated by committing phrasal translations that covered previously unused foreign words. Each derived hypothesis is placed in stack based on the number of foreign words it covers.

```
initialize hypothesesStack[0..nf];
create initial hypothesis hype_init;
```



```
add to stack  hypothesesStack[0]
for i=0 to nf-1:
for each hyp in  hypothesesStack[i]:
for each new_hyp that can be derived from hyp:
nf[new_hyp] = no. of foreign words covered by
new_hyp
add new_hyp to  hypothesesStack[nf[new_hyp];
prune hypothesesStack[nf[new_hyp];
find new hypothesis best_hyp in
hypothesisStack[nf];
output best path that leads to best_hyp;
```

We proceed through this hypothesis stack, going through each hypothesis in the stack, deriving new hypothesis for this hypothesis and placing them into new appropriate stack. After a new hypothesis placed into stack, the stack may have to be pruned by threshold or histogram pruning, if it has become too large. In the end, the best hypothesis of the ones that cover all foreign words is the final state of the best translation. We can read off the English words of the translation by following the back links in each hypothesis.

Data structure in phrase-base model:

In addition to efficient algorithms for decoding, we also need efficient data structures and strategies to store a model with millions of parameters. and strategies to store a model with millions of parameters. A common data structure in phrase-based models is the phrase table. This structure enumerates all of the phrase pairs in the model and maps them to their phrase translation probabilities. In practice, these phrase pairs may be extracted from parallel corpora containing tens or hundreds of millions of words. Even the most conservative extraction heuristics will produce a huge number of phrases under this condition. Efficient algorithms for storage and access are necessary for practical decoding algorithms. Since the quantity of available parallel data is always increasing, this is particularly important.

To generate a translation table for each pair of languages starting from a sentence-aligned parallel corpus, we used a modified version of the Moses training software. The software also required GIZA++ word alignment tool [12]. We generated for each phrase pair in the translation table 5 features: phrase translation probability (both directions), lexical weighting [4] (both directions) and phrase penalty (constant value).

Design Goals

This decoder mainly implements the phrase-based decoding in order to improve its performance, optimizations focused in three areas:

a. minimize the cost to build a hypothesis state (that is added to a stack)
b. minimize the cost of adding the hypothesis state to a stack
c. minimize the chance of building a hypothesis state that will not "stick" to its assigned stack (that will end up being thrown out from the stack, because it's out of the search parameters defined by the beam size and the beam threshold).

The configuration used for benchmarking:

i. Beam size: 50 hypotheses
ii. Distortion limit: 4 (Pharaoh's semantics)
iii. Max. phrase length: 5 words
iv. Beam threshold:
v. Translation table threshold: 10 .

Modularity:

Firstly, software modularity enables us to work on one component of the decoder without affecting other components. A modular design reduces the learning curve to understand the entire system if they are only developing a specific part.

Modularity also assists in the re-using of components by separating the implementation details from the module interface.

We takes advantage of JAVA support for object-oriented and generic programming to enable modularity. The simple application which currently comes with the decoder enables users to use the system via the command line .

Therefore, the current typical compilation of the decoder would combine the libraries from IRSTLM, SRILM, decoder, and decoder-cmd to create a binary executable.

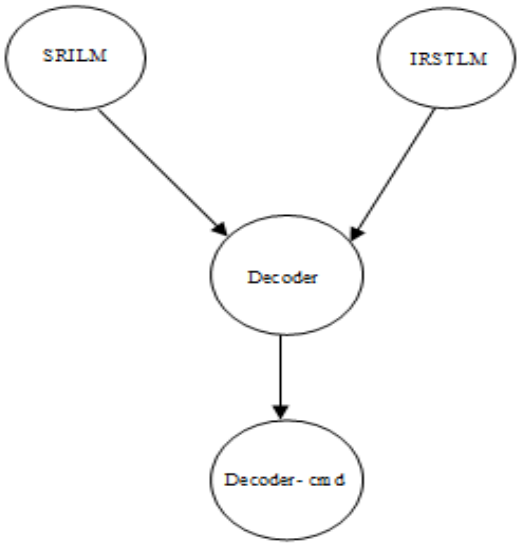

Figure 2 : Project Dependencies

The input into the decoder is simple string (sentence). Language models are abstracted to enable different implementations to be used and provide a framework for more complex models such as factored LM and the Bloom filter language model [13]. Similarly, phrase tables are abstracted to provide support for multiple implementations.

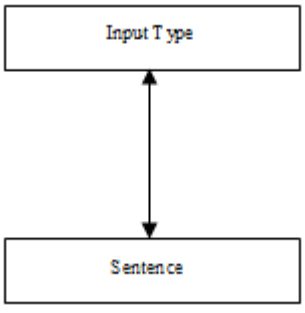

Figure 3 : Input Type

Parameters for running system.

There are optimizations related to the probability calculation. Standard data files (LM and translation table data files) will have all probabilities <= 1 (log-probabilities <= 0). Using this assumptions, decoder can optimize the decoding process,

eliminating up to 50% of the computation time (under the following parameters:

i. dl 4 -b 0.01 -ttable-limit 10 -s 50 -x-max-phrase-length 5 and with fast vocabulary-based language model).
ii. read (1) -- reads the input file instead of reading it from StdIn
iii. write (1) -- writes the input file instead of writting it to StdOut

CONCLUSION

This decoder delivers a very good baseline system. This is capable of estimating parameters over a large develop ment corpus in a reasonable time, thus it is able to generate highly relevant parameters. We have applied the sound software engineering principles and design to the implementation of the decoder which has enabled other researchers to use and extend its functionality. We believe this has been a major factor for the widespread adoption of Moses within the SMT community. We hope that the design of the decoder will enable it to maintain it leading edge status into the future.

REFERENCES

[1] Philipp Koehn, Hieu Hoang, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, Evan Herbst, "Moses: Open Source Toolkit for Statistical Machine Translation", Proceedings of the ACL 2007 Demo and Poster Sessions, pages 177–180, June 2007.

[2] Brown, P. F., J. Cocke, et al. (1990). "A statistical approach to machine translation."

[3] Och, F. J. and H. Ney (2004). "The alignment template approach to statistical machine translation." Computational Linguistics.

[4] Koehn, P. (2004). Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Models. AMTA.

[5] CHURCH, K. AND HOVY, E. 1993. Good applications for crummy machine translation. Mach. Transl. 8, 239–258.

[6] P. Clarkson and R. Rosenfeld, "Statistical language modeling using the CMU-Cambridge toolkit", in G. Kokkinakis, N. Fakotakis, and E. Dermatas, editors, Proc. EUROSPEECH, vol. 1, pp. 2707–2710, Rhodes, Greece, Sep. 1997.

[7] F. Jelinek, "Up from trigrams! The struggle for improved language models", in Proc. EUROSPEECH, pp. 1037–1040, Genova, Italy, Sep. 1991.

[8] R. Rosenfeld, "Two decades of statistical language modeling: Where do we go from here?", Proceedings of the IEEE, vol. 88, 2000.

[9] MITCHELL, T. M. 1997. Machine Learning. McGraw-Hill.

[10] Jelinek, F. (1998). Statistical Methods for Speech Recognition. The MIT Press.

[11] Och, F. J., Ueffing, N., and Ney, H. (2001). An efficient A* search algorithm for statistical machine translation. In Data-Driven MT Workshop.

[12] Franz Josef Och and Hermann Ney. 2003. A system atic comparison of various statistical alignment mod els. Computational Linguistics, 29(1):19–51.

[13] Talbot, D. and M. Osborne (2007). Smoothed Bloom filter language models: Tera-Scale LMs on the Cheap. EMNLP, Prague, Czech Republic.

