Gmail **Hieu Hoang <hieuhoang@gmail.com>**

Your AMTA 2016 Submission (Number 2)

1 message

mtresearchers@amtaweb.org <mtresearchers@amtaweb.org> 16 August 2016 at 14:57
To: hieuhoang@gmail.com

Dear Hieu Hoang:

On behalf of the AMTA 2016 Program Committee, I am delighted to inform you that the following submission has been accepted to appear at the conference:

Fast, Scalable Phrase-Based SMT Decoding

The Program Committee worked very hard to thoroughly review all the submitted papers. Please repay their efforts, by following their suggestions when you revise your paper.

When you are finished, you can upload your final manuscript at the following site:

https://www.softconf.com/amta2016/papers/

You will be prompted to login to your START account. If you do not see your submission, you can access it with the following passcode:

2X-H5D7P7H9C6

Alternatively, you can click on the following URL, which will take you directly to a form to submit your final paper (after logging into your account):

https://www.softconf.com/amta2016/papers/user/scmd.cgi?scmd=aLogin&passcode=2X-H5D7P7H9C6

The reviews and comments are attached below. Again, try to follow their advice when you revise your paper.

Congratulations on your fine work. If you have any additional questions, please feel free to get in touch.

Best Regards,
Lane and Spence
AMTA 2016

==========
AMTA 2016 Reviews for Submission #2
==========

Title: Fast, Scalable Phrase-Based SMT Decoding

Authors: Hieu Hoang, Nikolay Bogoychev, Lane Schwartz and Marcin Junczys-Dowmunt
==========
REVIEWER #1
==========

Reviewer's Scores

Appropriateness: 5
Originality: 3
Substance: 4
Soundness / Correctness: 4
Clarity: 3
Meaningful Comparison: 4
Impact of Ideas / Results: 4
Impact of Accompanying Software: 5
Impact of Accompanying Dataset / Resource: 1
Recommendation: 5
Reviewer Confidence: 4

Comments

This paper describes a number of optimizations that can be made to speed up phrase-based decoding, in particular in multi-threading scenarios. These include customized memory pools, alternate methods for organizing stacks, and incorporating lexicalized reordering information directly into the phrase table. The resulting decoder is much faster than Moses and its performance continues to scale as more threads are added.

This is good work, and likely to be highly relevant to many. The fact that the code will be released open source is very important, and a major plus. Furthermore, the paper is a good fit for AMTA with its industry focus.

With the positives out of the way – the paper itself is not that great. In general, it feels rushed and not very academic. So, although I feel strongly that this paper should be accepted based on the strength of the work, I hope the authors will consider the following criticisms when preparing the next version:

The basic premise, multi-thread decoding, is never clearly defined. People familiar with Moses will understand that we are discussing one-thread-per-sentence, embarrassingly parallel decoding (hence why it is so surprising that performance does not scale with more cores). This should be spelled out, as it is easy to imagine more complicated schemes (such as multiple threads working to decode a single sentence).

The prior work section mentions most of the relevant prior work, but it feels more like a list than a discussion. It does not do a good job of relating prior work to the work done in this paper. In particular, it would be very useful to better specify which of Moses's search optimizations (such as, for example, the incremental decoding described in Heafield et al. 2014) are included in this new decoder.

It would be helpful to provide an explanation for why folding the Lexicalized Reordering model into the phrase table (and therefore, dodging an extra lookup) seems to have such a big impact on scalability (threads help beyond 16).

Section 3's first paragraph mentions "lexicalized re-ordering model using the compact data structure" – the phrase "compact data structure" does not appear previously in the paper, and should be defined.

Table 2: the 637 MB in this table is jarring, would we change it to 0.6GB?

Please put figures and tables at the top or bottom, not in the middle, of pages.

I do not understand what I am supposed to take away from Figures 3 and 4. This pseudo-code doesn't improve my understanding of how decoding with ministacks differs from traditional cardinality stacks. In particular, there is no mention of how ministacks impact pruning (which seems to be really important), nor is there an illustration of how they change the handling of the distortion limit (as suggested at the end of paragraph 5 of Section 4.2. As it stands, I do not know what the pseudo-code adds to the paper.

Figure 5 hardly looks like a ringing endorsement for any of the stack strategies, and were I presented with it, my instinct would be to stick with the status quo (cardinality stacks). The paragraph discussing Figure 5 should close with a clear statement of which strategy was decided upon and why.

Figure 6: shouldn't we be looking at model score on the y-axis if we want to consider this decoder as a drop-in replacement? BLEU is fairly unreliable when comparing similar search strategies.

I couldn't find the n-gram order of the language models for either the Arabic or French experiments. This seems like it would be important when understanding decoding speed or replicating these experiments.

==========
REVIEWER #2
==========

Reviewer's Scores

Appropriateness: 5
Originality: 3
Substance: 4
Soundness / Correctness: 4
Clarity: 4
Meaningful Comparison: 5
Impact of Ideas / Results: 2
Impact of Accompanying Software: 1
Impact of Accompanying Dataset / Resource: 1
Recommendation: 4
Reviewer Confidence: 4

Comments

This is a nice engineering work on putting Moses more practical by carefully trading off the speed and memory especially for multi core settings. The idea is rather simple; do not directly allocate memory from heaps, but use own memory pool for each thread. Better caching for phrase tables combined with probing based access. More lexicalized reordering parameters into phrase tables. trading off the stack configuration, whether to differentiate by coverages or by a combination of coverage with the last covered position. Experimental results indicates better performance under multiple threads that might help reducing latencies during service.

I like this kind of engineering contribution. However, the ideas here a somewhat known but not explicitly published. Nevertheless, this papers deserves acceptance, I think.

==========
REVIEWER #3
==========

Reviewer's Scores

Appropriateness: 4
Originality: 3
Substance: 3
Soundness / Correctness: 3
Clarity: 3
Meaningful Comparison: 3
Impact of Ideas / Results: 3
Impact of Accompanying Software: 1
Impact of Accompanying Dataset / Resource: 4
Recommendation: 3
Reviewer Confidence: 4

Comments

This paper focuses on the fast decoding for phrase-based SMT.
It proposes four main techniques: memory management, stack configuration, rule caching and the unification of phrase-table and lexical reordering table.
Together with using the multicores, these techniques lead to one order of speedups for decoding.

Overall, I think this paper provides more contributions from engineering than from research.
There are some issues in this paper.

1. As presented in Section 1, it seems that the contribution of this paper is the decoding speedups on multicore servers, but the four techniques are not clearly related

to the multicore servers. Parallelization over multiple sentences is trivial for decoding, but it would be very nice if this work parallelizes over one sentence.

2. Some of these techniques are originally from other papers, for example, rule caching etc.

3. In Section 2, there are many paragraphs including only one sentence. Actually, these paragraphs can be easily re-organized for better presentation.

--
AMTA 2016 - https://www.softconf.com/amta2016/papers