

The 54th Annual Meeting of the Association for Computational Linguistics**ACL 2016****Author Response**

Title: Fast, Scalable Phrase-Based SMT Decoding

Authors: Hieu Hoang, Nikolay Bogoychev, Lane Schwartz, Kenneth Heafield and Marcin Junczys-Dowmunt

Instructions

The author response period has begun. The reviews for your submission are displayed on this page. If you want to respond to the points raised in the reviews, you may do so in the box provided below.

Please note: *you are not obligated to respond to the reviews.*

Review #1

APPROPRIATENESS: 4

CLARITY: 4

ORIGINALITY: 2

EMPIRICAL SOUNDNESS / CORRECTNESS: 4

THEORETICAL SOUNDNESS / CORRECTNESS: 3

MEANINGFUL COMPARISON: 2

SUBSTANCE: 2

IMPACT OF IDEAS OR RESULTS: 3

IMPACT OF ACCOMPANYING SOFTWARE: 4

IMPACT OF ACCOMPANYING DATASET: 1

RECOMMENDATION: 2

MENTORING: NO

AUTHOR RESPONSE: N/A

Comments

- Summary: The paper presents several improvements to the Moses decoder that result in strong improvements in scalability to a multi-threading environment. While I applaud the authors for their efforts and their willingness to share their code, scientifically the paper leaves much to be desired.

- Strengths: The results are strong: Improvements of up to a factor of 14.5 in decoding speed compared to Moses. If the code is released as a part of Moses, this is a great service to the community. Question: Will the code be part of Moses or do you plan to release your own toolkit?

- Weaknesses: The authors neglect to mention that some of the improvements have been investigated before and are part of open source toolkits other than Moses. The presentation is sloppy with plenty of errors, the authors do not offer explanations for the largest gains and in total I believe the amount of material is insufficient for a long paper.

- General Discussion:

- * the stack configuration you propose has been presented in (Zens & Ney, Improvements in Dynamic Programming Beam Search for Phrase-based Statistical Machine Translation, IWSLT 2008). It is implemented in the Jane decoder.
- * The Jane decoder also integrates the lex RO model into the phrase table
- * in 5.4, it is clear why your proposal increases speed. But you need to offer an intuition why the gain increases with more threads. In fact, your results indicate an error in your experiments: When you compare "integrated Lex RO" with "No Lex RO" and leave all other parameters unchanged, the former strictly performs more operations than the latter. It can therefore not be faster than using no RO model.
- * the "coverage" curve is missing in Fig. 5
- * Please clarify which of the models are fully stored in memory and which are read from disk on-demand. The model sizes you report should easily fit into memory of your 380G machine, which contradicts the first sentence in 2.3: "For any realistic sized phrase-based model to be used in an online situation, memory and loading time constraints requires to use load-on-demand phrase-table implementations."
- * You are using tiny language models in your experiments, which skews the results in your favor. I believe with realistic language model sizes, your improvements would be overpowered by the LM computation times.
- * Sec. 5.3: how do you decide which phrases are the "most common" ones?
- * Tab. 7: the #threads in the last row should probably be 32
- * Further relevant reading: (Wuebker et al, Fast and Scalable Decoding with Language Model Look-Ahead for Phrase-based Statistical Machine Translation, ACL 2012)
- * Many typos. Proofread!

Review #2

APPROPRIATENESS: 5
CLARITY: 2
ORIGINALITY: 3
EMPIRICAL SOUNDNESS / CORRECTNESS: 3
THEORETICAL SOUNDNESS / CORRECTNESS: 3
MEANINGFUL COMPARISON: 2
SUBSTANCE: 2
IMPACT OF IDEAS OR RESULTS: 4
IMPACT OF ACCOMPANYING SOFTWARE: 2
IMPACT OF ACCOMPANYING DATASET: 1
RECOMMENDATION: 2
MENTORING: YES
AUTHOR RESPONSE: NO

Comments

- Strengths:

This paper presents several improvements to decoder speed for Statistical Machine Translation. This issue is relevant in many environments where machine translation is applied in commercial

environments, and decoder speed is often critical when users are waiting actively for the translations to be produced. Personally, I feel the results are interesting, and are worth being published. Nevertheless, while the results seem quite promising, there are several shortcomings that prevent this reviewer from recommending this paper for acceptance.

- Weaknesses:

Even though the paper should be fairly easy to read, there are many places where the reader needs to struggle heavily to understand what is actually being said. Numerous English mistakes (only in the first page I was able to spot 12 of them) render the paper very difficult to read. In addition, there are many explanations that are missing, but are crucial towards understanding what the paper is actually about. These would typically be minor issues, but the high amount of them means that I wouldn't count on many people actually understanding the paper, let alone replicate the results.

The improvements introduced are heavily under-explained. Section 3 attempts to list them, but none of them are thoroughly enough explained so as to be able to understand what is going on. For instance:

* What are these memory pools? How do they provide better memory management than the OS? After reading the paper several times, I have some intuition about it, but no true certainty at all.

* The different stack configurations need further explanation. Why are some better than the others? What's the algorithmic difference that leads to better decoding time?

* How are "the most common source phrases" (page 6) selected? According to a test corpus? To the counts in training?

* The discussion about the ministacks is way too preliminary. I haven't been able to understand what they are.

- General Discussion:

All in all, I feel the work presented here is worth publishing. I see some potential, and the results look great. But the paper needs heavy proofreading, restructuring and clarification. I would recommend e.g. trying to group up the explanation about the improvements in Section 3, since right now they are spread about in several sections. Also, some small introduction is required in Section 1, since the reader is left puzzled in Section 2, related work, when a "closest in intent" work is reviewed... closest to what?

Review #3

APPROPRIATENESS: 3

CLARITY: 4

ORIGINALITY: 3

EMPIRICAL SOUNDNESS / CORRECTNESS: 4

THEORETICAL SOUNDNESS / CORRECTNESS: 2

MEANINGFUL COMPARISON: 4

SUBSTANCE: 3

IMPACT OF IDEAS OR RESULTS: 3

IMPACT OF ACCOMPANYING SOFTWARE: 1

IMPACT OF ACCOMPANYING DATASET: 1

RECOMMENDATION: 2

MENTORING: NO

AUTHOR RESPONSE: N/A

Comments

This paper presents an efficient implementation of Moses-like phrase-based SMT decoder. Its improvements are summarized in four aspects: (1) using custom memory management, (2) employing a different stack configuration suitable for applying distortion limits, (3) employing a different phrase table caching strategy for keeping commonly-used phrases, and (4) merging a lexicalized reordering model with a phrase table.

- Strengths: These aspects sound reasonable for speeding up the phrase-based decoder, and actually achieved large improvement in its decoding speed especially with multi-core situations.

- Weaknesses: This paper mostly sounds like "some tricks in software engineering have improved decoding speed." (2) and (3) are come from SMT-related issues but the speed improvement by these aspects is not so large. The large portion of the presented improvements comes from (1) --- basically not related to computational linguistics.

- General Discussion: All of the presented techniques are interesting and effective, but most of them are not come from linguistic or NLP- or SMT-related insights and findings. Although this kind of work is very valuable for practical use, I do not recommend this paper as "research paper" in ACL.

Some questions:

* Why the speed testing was done by a "closed" condition using a subset of the training data? Isn't it realistic situation?

* Moses's multi-core issue seems to be come from Hyper-threading from Figure 8

--- Isn't it an algorithmic problem but an engineering problem? How does Moses work with "bigger" servers?

Comments:

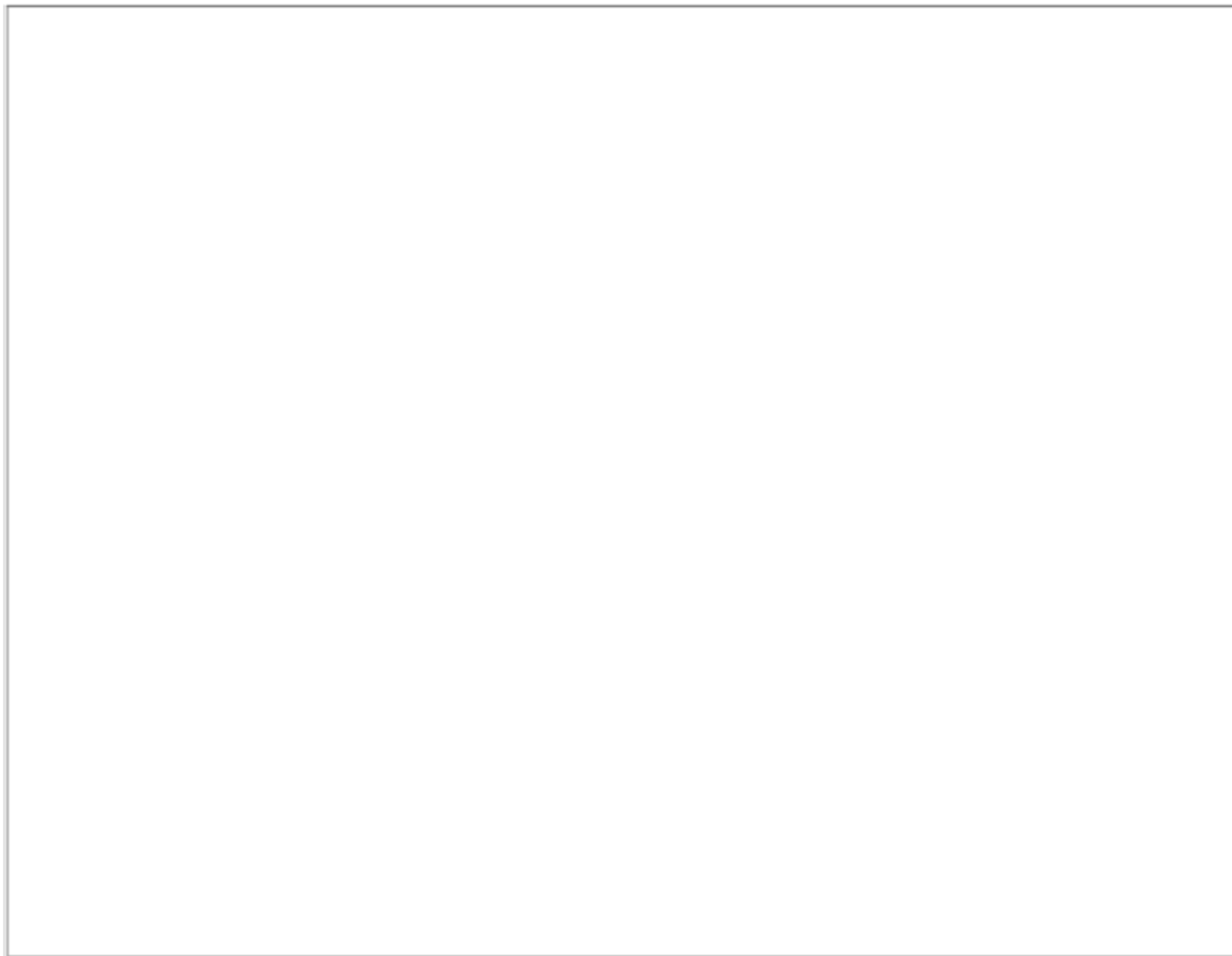
* Many typos. The manuscript needs proofreading before the submission. p.1 efficiently use multiple (Fernandez et al.) -> efficiently use multiple cores(?) (Fernandez et al.) p.1 There have be speculation -> There have been speculation p.1 with a single parameter -> with a single parameter p.3 All models files -> All model file p.4 formats were choosen for -> formats were chosen for p.4 Datastructures -> Data structures p.5 ie. -> i.e. p.6 optimatization -> optimization p.6 when mmore threads -> when more threads

* Wrong reference (Tanaka and Yamamoto, 2013)

* "latest version of the Moses decoder" does not specify the one the author(s) used. Use unique revision ID.

Submit Response

Use the following box to enter your response to the reviews. Please limit your comments to 600 words (longer responses will not be accepted by the system).



Submit

START Conference Manager (V2.61.0 - Rev. 4057)