Incremental Decoding for Phrase-based Statistical Machine Translation

Baskaran Sankaran, Ajeet Grewal and Anoop Sarkar
Simon Fraser University, Canada
{baskaran, asg10, anoop}@cs.sfu.ca

Extended Abstract

Statistical Machine Translation has advanced tremendously, resulting in the proliferation of several web-based and commercial translation services. In this paper we focus on the notion of *incremental translation* to translate text one word at a time. Such a translation service can be used for language learning, where the user is fluent in the target language and experiments with many different source language sentences interactively, or in real-time translation environments such as speech-speech translation or translation during interactive chats. Google (*http://translate.google.com*) seemingly does incremental decoding, but the underlying algorithms are not public knowledge. They may be simply re-translating the input each time using a fast decoder or re-using prior decoder states as we do here. In this paper we claim that incremental decoding is faster if we start from a partial state covering earlier words and decode just the new words instead of re-decoding the input from scratch each time.

We introduce an **incremental decoder** (ID) algorithm, which is a modification of a beam search decoder for phrase-based MT. When a new word is added to an existing partial sentence, our decoder simply begins from the previous decoder state. We propose *delayed pruning* (DP) a novel pruning strategy to reduce the search errors in incremental decoding.

We are not aware of any other work similar to ours; the closest work is *caitra* (Koehn, 2009) – a web based interactive tool for aided translation. However, that work pre-translates the entire text; also the interactions are limited to dynamic user suggestions and *not* for dynamic translations.

In this work, we address the twin complexities of partial future costs and search errors in incremental decoding so as to allow for faster and efficient incremental decoding. First, the future costs change at every step as words are added to the partial sentence. The decoder has to not only recompute future costs at every step, it should also update the previous partial decoder state to reflect the new future costs. Second, some of the entries that were pruned out in a prior partial decoder state might actually turn out to better candidates more of the input sentence is uncovered, resulting in search errors.

Incremental decoding relies on a *recompute and update* strategy for estimating future costs at every step. We begin with a single bin for the first source word and generate partial hypotheses. For each new source word it recomputes the future costs for all the phrases accounting for the new word and also updates the previous decoder state with the modified future costs. It then proceeds by extending older hypotheses and by generating new hypotheses resulting from the last word added using the recomputed future costs and log-linear combination of other feature scores including the language model (LM) scores.

Algorithm 1 shows the pseudocode for our incremental decoder. We use $P_{type}(l)$ to denote an l-word phrase and H_{type} to denote the set of hypotheses; in both cases *type* correspond to either *old* or *new* indicating if it was not known in the previous decoding state or not. Given a partial sentence S_i (S_i and s_i denote an i-word partial sentence and i^{th} word in a (partial) sentence respectively) the decoder starts with a pre-process step that retrieves the previous decoding state and adds new bins corresponding to the words added. It also extracts the new set of phrases (P_{new}) for the i^{th} word and recomputes the future-costs (f_c) for all the phrases.

In the incremental decoding phase, it iterates through the bins beginning from the first. At each iteration, it updates the future costs and coverage vector of the hypotheses in the partial decoding state. It also adds any newly found phrase to the bin subject to the distortion limit d (line 5).

The inner for-loop (lines 7 to 11) corresponds to the extension of hypotheses sets (grouped by same coverage vector) to generate new hypotheses. While extending hypotheses from a previous bin, this differs from the regular decoder in two ways: i) by extending the existing hypotheses H_{old}

Algorithm 1 Incremental Decoder pseudocode

1: *Input*: (partial) sentence S_p: $s_1 s_2 ... s_{i-1} s_i$ with l_s words where s_i is the new word
2: *PreProcess*(S_p)
3: **for** every bin b_j in $(1 \dots i)$ **do**
4: Update future cost and cover set $\forall$ H_{old}
5: Add any new phrase of length b_j (subject to d)
6: **for** every bin b_k in $(b_{j-4} \dots b_{j-1})$ **do**
7: Generate H_{new} for b_j by extending:
8: H_{old} with every other $P_{new}(b_j - b_k)$
9: H_{new} with every other $P_{any}(b_j - b_k)$
10: Prune bin b_j

of previous decoder state S_{i-1} with new phrases P_{new} (line 9) and ii) by generating new hypotheses that are unknown in the previous state (line 10).

The entries in the bin are then pruned according to the specified beam size and/or threshold. However, as noted above, some of the pruned entries could potentially become better candidates as the sentence is completed later. Thus using the conventional pruning methods could result in search error lowering the BLEU score.

We overcome the search errors issue by introducing *delayed pruning* (DP). The idea is to delay pruning of hypotheses earlier on. A hypothesis having a poor score at a particular decoding state, we call a *tortoise* having a slow start as opposed to a high scoring *hare* hypothesis in the same state. We hypothesize that, given enough chance a tortoise might improve its score and move ahead of a hare in terms of log-linear score[1].

We achieve this by relaxing the cube pruning to generate a small, fixed number of hypotheses for hyper-edges that are not represented in the priority queue of the bin and place them in the bin. These hypotheses are distinct from the regular top-k hare derivations. Though this reduces the search error, it leads to increasing number of possibilities to be explored at later stages with vast majority of them being worse.

We use a two level strategy to avoid this proliferation of worse hypotheses and at the same time to reduce search error. At the first level, we choose few hypotheses by comparing their normalized language model scores against a threshold and flag them as *tortoises*. These are extended minimally at each subsequent decoder state subject to their normalized LM score.

At the second level, poor scoring tortoises are pruned out beyond a threshold number of bins called *race-course limit*. When a tortoise improves in score and breaks into the beam through the cube pruning step, it is removed from the tortoise set and placed in the regular decoding stream.

The evaluation of our decoder was performed using our implementation of a beam-search decoder which is similar to Moses (Koehn et al., 2007). The decoder was written in Java and included *cube pruning* (Huang and Chiang, 2007) and *lazier cube pruning* (Pust and Knight, 2009) as part of the decoder. We used Giza++ and Moses respectively for aligning the sentences and training the system and used 7 standard features identical to Moses. The decoder objects were serialized in the memory and retrieved as required by the ID.

We experimented in three language pairs using WMT07 and WMT10 datasets for French-English, English-French and in English-Spanish. Our experiments focused on comparing incremental decoding with regular decoding in terms of search errors (as measured by BLEU) and translation speed. We also studied the effects of different race course limits on search error.

Due to space constraint, we provide below brief results only for French-English translation in Table1. We would like to note that our in-house decoder compares favourably with Moses: the slightly better bleu score for our decoder could be due to the minor differences in the corresponding config settings.

Decoder	**Bleu**	**TER**
Moses	26.98	55.11
Regular decoding	27.53	54.18
ID w/o delay pruning	27.01	54.74
ID w delay pruning	27.62	54.55

Table 1: French-English using WMT07 dataset

Secondly, the incremental decoding without the delay pruning resulted in poor candidates either in terms of translation score or its quality. However, the incremental decoding showed noticeable improvement when the delayed pruning was turned on as can be seen in the bleu; the TER scores also show a similar pattern. Furthermore, our results showed the incremental decoding with DP to be faster than the regular decoding by about 50%.

[1]Here we are concerned not with the speed but with avoiding search errors and we consider log-linear score as a goodness measure for a partial hypothesis

References

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *In Proceedings of ACL-07*, Prague, Czech Republic, June.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *In Proceedings of ACL-07: Demo and Poster Sessions*, Prague, Czech Republic, June.

Philipp Koehn. 2009. A web-based interactive computer aided translation tool. In *In Proceedings of ACL-IJCNLP 2009: Software Demonstrations*, Suntec, Singapore, August.

Michael Pust and Kevin Knight. 2009. Faster mt decoding through pervasive laziness. In *In Proceedings of NAACL-HLT*, Boulder, Colorado, June.