EFFICIENT NEARLY ERROR-LESS LVCSR DECODING BASED ON INCREMENTAL FORWARD AND BACKWARD PASSES

David Nolden, Ralf Schlüter, Hermann Ney

Human Language Technology and Pattern Recognition Group
RWTH Aachen University, Aachen, Germany
{nolden, schlueter, ney}@cs.rwth-aachen.de

ABSTRACT

We show that most search errors can be identified by aligning the results of a symmetric forward and backward decoding pass. Based on this knowledge, we introduce an efficient high-level decoding architecture which yields virtually no search errors, and requires virtually no manual tuning. We perform an initial forward- and backward decoding with tight initial beams, then we identify search errors, and then we recursively increment the beam sizes and perform new forward and backward decodings for erroneous intervals until no more search errors are detected. Consequently, each utterance and even each single word is decoded with the smallest beam size required to decode it correctly. On all tested systems we achieve an error rate equal or very close to classical decoding with ideally tuned beam size, but unsupervisedly without specific tuning, and at around 2 times faster runtime. An additional speedup by factor 2 can be achieved by decoding the forward and backward pass in separate threads.

Index Terms— search, error detection, error-less, decoding, LVCSR, pruning

1. INTRODUCTION

Search space pruning is essential [1, 2] for efficient large vocabulary continuous speech recognition (LVCSR). The most important pruning method is the global beam pruning, which limits the hypothesis space based on a specified beam size. The beam size is typically chosen manually to achieve a specific tradeoff between accuracy and efficiency.

While it is simple to run several recognitions with different beam sizes on a development corpus and choose a beam size with the desired tradeoff, it is never clear how the same beam would perform on unseen data, because the optimal beam size not only depends on the models, but also on the mismatch between the models and the data which is being recognized.

Experience has shown that there usually is a specific beam size at which the precision reaches a limit, but that exact beam size tends to vary widely on different speech recognition systems and corpora. The precision and efficiency of speech recognition systems is very sensitive regarding the beam size, and even a slight mis-tuning of the beam size can either lead to an enormous waste of resources, or a significant loss in precision.

In research, where precision usually is the main factor measured, it is common practice to choose an exaggerated beam size to be on the safe side when dealing with unknown input data, which results in suboptimal efficiency.

In this work we introduce an algorithm which unsupervisedly recognizes each utterance and even each word with (nearly) exactly the beam size required to recognize it correctly, without manual tuning, and without a transcription of the recognized text to tune with. We achieve this by incrementally detecting search errors through symmetric forward- and backward decoding passes, and then re-recognizing the erroneous portions of the signal with incremented beam sizes, until no more errors are detected.

Previous uses of forward / backward decoding in LVCSR can be split into two general categories: Firstly as a speedup technique, where the results of the forward decoding pass are used to guide the backward decoding pass [3, 4], basically creating a very advanced look-ahead similar to [5], but with much more complex involved models, and with a much longer look-ahead range. Secondly as a technique to generally improve the modelling precision by building separate forward- and backward models and combining the results using ROVER or similar techniques [6, 7, 8]. In this work, we build a backward pass which is exactly equivalent to the forward pass regarding the involved models like the authors of [4], but we combine the passes in the opposite way: Instead of making one pass depend on the other pass, we rather make both passes as independent as possible regarding the search space, in order to detect the occured search errors based on the independent forward and backward decoding results.

While the primary goal of this work is to deterministically and unsupervisedly achieve the best possible precision at reasonable runtime efficiency, we will also show that our incremental decoding is considerably faster than classical decoding with a static beam size tuned to achieve the same accuracy, because for most utterances a much smaller beam suffices to find the correct word sequence.

2. SEARCH ERROR DETECTION

Viterbi search tries to find the best scoring path through the HMM search network. Beam pruning restricts how far the decoder considers hypotheses which temporarily have a bad score relative to the best one. As a simple abstraction, the search space can be considered a 3-dimensional surface, where the x and y axis represent a combination of an HMM state and a timeframe index, and the z axis is the local score (emission and language model score combined). The goal of the decoder then is to find a valid path over the surface which minimizes the accumulated scores. A search error occurs whenever the globally best path crosses a valley of bad scores, but there is an alternative path which seems more promising at the given time, and the global beam is too tight to let the decoder hypotheses cross the valley. Figure 1 illus-

trates such a search error based on a decoder which can only recognize two alternative single-words. The actual search space in LVCSR is cyclic and contains tens of thousands of words, however the same principles apply. In the illustration, the decoder follows the path of lowest resistance based on its tight beam, which leads through the wrong upper word, while the globally best path leads through the correct lower word, thus a search error occurs.

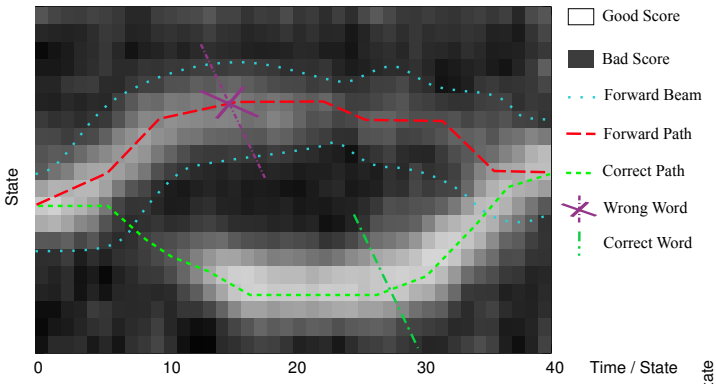

Fig. 1. *Forward search with tight beam finds the wrong word.*

Figure 2 illustrates a backward search based on the same models, which detects the correct lower path. If forward and backward search yield different paths then we know that at least one is wrong.

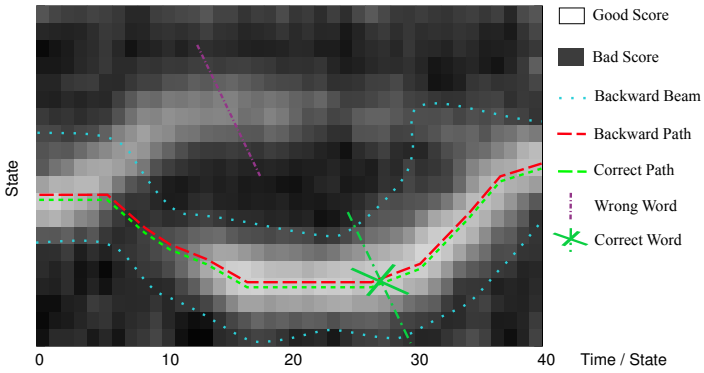

Fig. 2. *The backward search (coincidentally) finds the correct word even with a tight beam.*

In this example, if we widen the beam size for the forward search which produced the worse scoring path, then the forward search finds the correct lower path leading throught the correct word too (see Figure 3).

Since the global optimum of the forward and backward pass are the same, it is clear that a mismatch in the forward and backward pass proves the occurence of a search error. However, does an equality of the forward and backward pass result also prove the *absence* of a search error? The answer is clearly no: Consider Figure 4. If the correct path is isolated by a score valley on both sides, and the path of lowest resistance for both the forward and backward search lead through the same word, then the same search error occurs in both the forward and the backward pass.

However, in LVCSR it is very unlikely that forward and backward search yield exactly the same search error, for the following reasons:

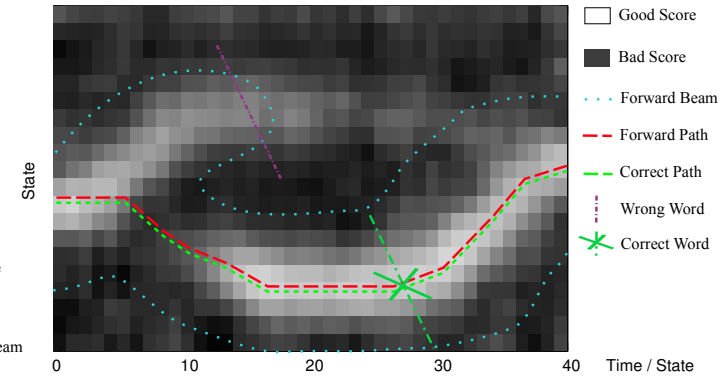

Fig. 3. *With a wider beam, the forward search finds the correct word too.*

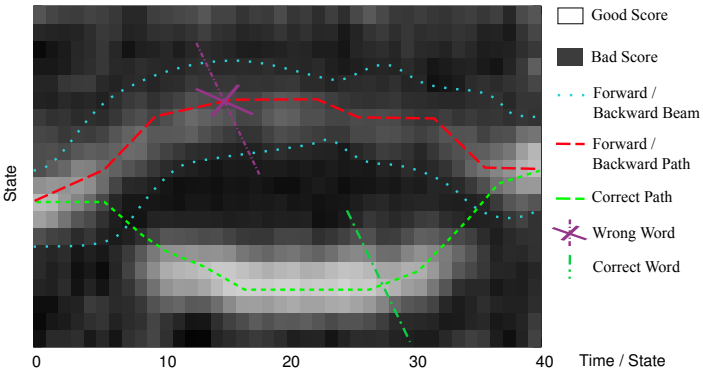

Fig. 4. *Problem: Forward and backward search both find the wrong word based on a tight beam.*

1. When the globally best path leads through a valley of bad scores, then that valley *must* be followed by much better scores, otherwise the path would not be the globally best path. Therefore, if the globally best word starts with bad scores, it is very likely that it ends with good scores, and thus, the more likely it is that the forward search yields a specific search error, the less likely it is that the backward search yields the same error, and vice versa. An obvious problem here are long words, because long words can start and end with bad scores, but have plently of time in between to outweight the mismatched start and end with good scores (like in Figure 4). However, in LVCSR, the vocabulary consists of thousands up to millions of different words, all based on the same underlying acoustic models. Most probably, the phoneme sequence which forms the well-scoring central part of the correct word in Figure 4 would also be covered by combinations of other smaller words, and the forward and backward search would recognize different sequences of those words, rather than a common long unlikely alternative word.

2. In general, as long as the models are asymmetric (eg. at least a bigram language model), each search direction does its local pruning decisions purely based on knowledge which is yet completely hidden to the other direction (eg. forward decoder based purely on past, backward decoder based purely on future). Thus decoder errors of both directions can be considered independent, and therefore the a priori likelihood of both directions

doing the same search error is $1/N$ where N is the size of the vocabulary (ignoring the reasoning mentioned in 1.). The caveat here are some components of the models which are inherently symmetric (see Section 4).

The following experiments will show that we indeed can detect the majority of all search errors by aligning the result of a symmetric forward and backward decoding pass.

3. BACKWARD DECODING

In order to detect search errors as described in Section 2 we must be able to perform exactly symmetric decoding for the forward and backward pass, eg. the globally best path and its score (and thus also the best word sequence) must be exactly the same for the forward and backward search. We perform our experiments based on a dynamic network decoder following the word conditioned tree search approach with a minimized search network, full-order sparse language model (LM) look-ahead, and acoustic look-ahead, as described in [9]. Reversing the search network of such a decoder is relatively straightforward: During construction of the single-word search network, we simply reverse the phoneme sequences of words and their context-dependency. Additionally an exactly reversed n-gram LM is required, which we reverse using the mechanism described in [4] using a script which was kindly published by the authors. Furthermore, as our decoder allows no skip transitions from the previous-to-last HMM state of a word into the first HMM state of the successor word, we need to forbid skip transitions starting at virtual root states, because those are the equivalent transitions in the reversed search network.

4. DEGENERATED SEARCH

There are certain situations in which the independence between forward and backward decoding as described in Section 2 is not given.

Firstly, the LM is in parts symmetric: Both the forward and the backward LM have a shared unigram back-off level in which words are biased the same way from both directions. This is specifically problematic when the LM training data is limited, because then the unigram component plays a larger role regarding the overall scores.

Secondly, the non-word noise models (silence etc.) usually are not modelled by the LM and thus have an LM score of zero. If the beams are much too tight, then the search hypotheses of both the forward and the backward pass may focus only on the noise models, and thus produce the same search error in forward and backward search.

To somewhat prevent such degenerated decoding, we enforce a specific minimum search space size. However the size of the search space is very system-specific and also depends on the confidence of a specific segment, thus we keep these limits very low, and rather rely on automatic beam adaptation to steer the beam into reasonable areas (the search error rate should stay below 50%, see Section 7).

5. REPETITIVE DECODING

Based on the forward / backward search error detection introduced in the previous sections, we can perform virtually error-less decoding by applying Algorithm 1 to each utterance.

However this algorithm wastes resources if the utterances are large, because the whole utterance is re-recognized with a larger beam even if only a small portion of the utterance was recognized wrongly.

Loop:
- Perform forward and backward decoding, align results.
- If an **error** (eg. mismatch) was detected:
  - Increment the beam size, go back to **Loop**.
- If **no error** was detected:
  - We're ready, the utterance was (probably) recognized without search errors.

Algorithm 1: *Repetitive forward-backward search.*

6. INCREMENTAL DECODING

The effect of search errors is local. We know from the word pair approximation [10] that the start time of one word mainly depends on the predecessor word, but that the effect of earlier words regarding the acoustic alignment fades out very quickly. Thus, if we want to correct a detected search error, it may be sufficient regarding the acoustic alignment to re-recognize the erroneous portion plus *one* correctly recognized context word. The boundary time between the correct context word and the wrongly recognized words would probably change, but the boundary time between the correct context word and the its outer context is static following the word pair approximation, and thus would be left untouched.

Based on this idea, we can only updates those portions of the utterance which were recognized erroneously, plus one surrounding correctly recognized context word on each side, following Algorithm 2.

Start: **Refine(**0**,** T**,** $()$**)**.

Refine(S**,** T**,** C**):**
- Perform forward and backward decoding on timeframes S ... T, with decoder context C.
- Align forward and backward results:
  - Align overlapping words with equal identity.
  - Select closed ranges of aligned words.
  - For each range, discard boundary words until their inner boundary time matches.
  - Discard ranges which are shorter than $n-1$.
- For each interval S' ... T' not covered by the aligned ranges:
  - Extend the interval by one aligned context word, and build the new decoder context C' based on the surrounding words and the acoustic word boundaries.
  - **Refine(**S'**,** T'**,** C'**)**.
- Combine refined results with results that were already recognized correctly.

Algorithm 2: *Incremental forward-backward search.*

Figure 5 illustrates the recursive search algorithm based on a simple made up english example. Overall, when accumulating the runtime of all passes, the recursive algorithm is still faster (with a real time factor of $2 \cdot 2.4 = 4.8$) than when recognizing the whole sequence with the beam size which would be required to recognize all words correctly in the first forward-pass alone (eg. 5.4).

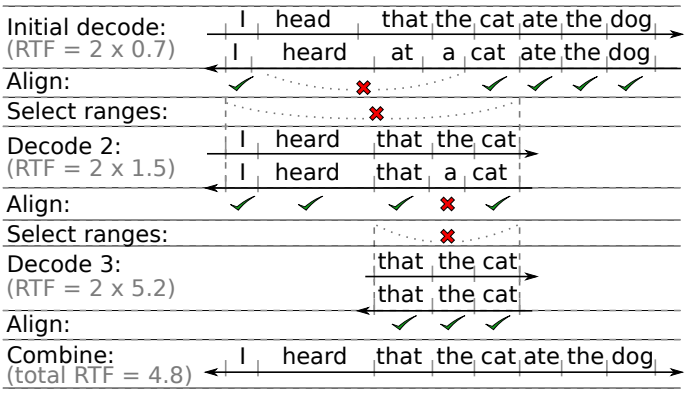

Fig. 5. *Illustration of incremental forward-backward search.*

Special care has to be taken regarding the n-gram LM, because it depends on $n-1$ predecessor words. We can only update multiple erroneous sub-ranges if there are at least $n-1$ correctly recognized words between them, because otherwise the updated ranges are not independent. Therefore we merge all update ranges which are not divided by at least $n-1$ correctly recognized words.

Furthermore, our error detection algorithm is not 100% reliable, and the word pair approximation is just an approximation which may not hold in all cases, thus it may happen that words which were earlier marked as correctly recognized change when they are re-recognized as context words with a larger beam. If this happens, then we go back one step upwards, mark the corresponding word as mismatch, thereby extending the corresponding update range (or merging it with other ranges), and then we re-recognize the grown update range(s). Fortunately this happens so rarely that it has a minor effect on the overall efficiency.

It may happen, either due to numerical instabilities, or because finding the correct path is simply not feasible under realistic pruning constraints (eg. histogram pruning), that we fail to ever find the same word sequence in forward and backward decoding, and thus might end in an endless loop of ever growing beam sizes. For this case, we have integrated two abortion points: If the histogram pruning is fully saturated for at least 50% of all frames, or if the beam pruning has crossed a specific absolute value (eg. typically 50 times the LM scaling factor), then we give up. Fortunately, this also happens rarely, typically in less than 0.3% of all utterances. Additionally we define a tolerance parameter l: If the difference between the score found in the forward pass and the backward pass is lower than or equal to this value, then we count it as a match, no matter how the alignment looks. This parameter is specifically important in cases where the models can produce different paths with exactly the same scores.

In order to correctly update subranges of an utterance with across-word acoustic modelling and an n-gram LM, we need to correctly re-initialize the decoder to start with the corresponding left LM context and with the corresponding left acoustic context, and to choose the final state with the correct corresponding right acoustic context and considering the right LM context in the final language model score. We achieve this by encoding the search network state used to cross the word boundaries in the lattice, and while updating we enforce those corresponding boundary states from the previous decoding pass to be used. Correctly initializing and finalizing the LM context is trivial in a dynamic network decoder.

Computing emission scores usually takes a significant portion of the total runtime, thus we share computed emission scores between the forward- and backward pass, and between initial and later passes.

7. INITIAL BEAM ADAPTATION

The beam size used for the initial decoding is important, because we may miss search errors if the initial decoding is degenerated. On the other hand, if the initial beam size is too large, then we may lose the efficiency advantage which we get from the incremental decoding. Therefore we start with an exaggerated initial beam size, but then we adapt the initial beam size on-the-fly to match a specific initial search error.

We compute the initial search error rate for an utterance as follows:

$$R = \frac{(F + B - 2 \cdot C)}{F + B} \tag{1}$$

Where F is the number of tokens produced by the forward search, B is the number of tokens produced by the backward search, and C is the number of tokens that were recognized correctly. After each processed utterance, we permanently tighten the initial beam size used for the next utterances if the initial search error rate was below our target error rate, otherwise we widen the beam. Our target search error rate used for the following experiments is 20%, which corresponds to a degradation of 20% absolute in precision.

Since we always randomize the order in which utterances are processed by the decoder, we can be sure that we don't over-adapt the beam size regarding a specific speaker or condition. Otherwise special care would have to be taken to prevent the initial beam from becoming problematically tight (see Section 4).

8. STRICTNESS

But what exactly do we consider a search error? In the most strict interpretation, a search error occurs whenever both passes don't produce exactly the same token sequence (including silence and noise tokens) with equal boundary times and equal scores. However, such a strict interpretation may be exaggerated, as all we usually really care about is finding the correct word sequence. Since we use the outcome of the initial decoding to adapt the beam size, we use a more strict interpretation during the initial decoding and the later incremental decodings, to prevent mis-tuning of the initial beam. During initial decoding, we enforce equal boundary times and equal scores for all tokens (including noise tokens). During the incremental decodings, we only compare the identities of word tokens, but ignore noise tokens, boundary times, and scores.

9. EXPERIMENTAL RESULTS

The primary advantage of our proposed framework is the robustness and convenience: No specific tuning is required, and we hope the system to blindly achieve a very good precision at reasonable runtime, independently of the data which is being recognized. To validate this robustness, we test the system on a variety of different languages.

We perform experiments on our Cantonese, Vietnamese, Turkish, Pashto, and Tagalog systems which we trained for the IARPA Babel program as part of the LORELEI consortium [11] under the BabelLR condition (eg. only resources supplied by the program were used for training). All those

systems were constructed with a very similar anatomy: Deep hierarchical bottleneck MLP features [12] are concatenated with MFCC or Gammatone features, a voicedness feature, and a tone feature. Based on those featuers, 2-pass acoustic models with a tied covariance matrix comprising around 4500 Gaussian mixtures based on a triphone CART with across-word acoustic modelling, and overall around 75 Gaussian densities per mixture, are trained. Confidence-weighted CMLLR adaptation and MPE training is used for the second pass. A 4-gram Kneser-Ney LM is trained on the transcription of the acoustic training data (except for Vietnamese where only a 2-gram LM is used). We perform all experiments on the second pass only. The vocabulary is supplied by the Babel program and typically contains around 20k to 40k pronunciations. The training data typically comprises around 70 hours of real speech (not counting silence) with 1 to 2 million running words. The test corpus used for our experiments consists of around 7 hours of real speech. The used segmentation is supplied by IBM, and typical utterances have a length of 3 to 7 seconds. Most of the data are challenging low quality recordings of telephone conversations. The performance of the models is competitive both within the LORELEI consortium and across, as confirmed during the 2013 Babel evaluation campaign.

Where not stated otherwise, real time factors (RTF) were measured directly on our computing cluster consisting primarily of 16-core AMD Opteron machines with around 2.6Ghz, under full load. Each run was split up into 120 sub-jobs which were scheduled separately, to reach a good randomized distribution among the different cluster nodes. The runtime on a dedicated Intel Core2 Duo 2.8Ghz desktop machine is typically 2 to 3 times faster than the reported numbers. We have verified that the reported numbers are representative based on a large number of experiments.

For all experiments, the word end pruning beam is kept in a linear relationship to the primary global beam, typically by the factor 0.5. The histogram pruning is configured to cap the size of the search space at 200k state hypotheses.

Figure 6 shows the results achieved on the Babel Turkish task. The best achievable WER with a statically tuned beam is 49.6% at an RTF of 9.5, and 49.7% is achieved at RTF 4.75. We achieve a WER of 49.7% using incremental decoding at an overall accumulated RTF of 1.72, which is nearly three times faster than the statically tuned beam for the same WER. By performing the forward and backward search in separate threads, we can achieve an additional speedup nearly by factor 2.

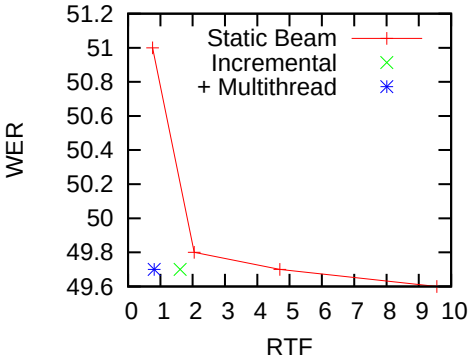

Fig. 6. *Babel Turkish.*

Figure 7 shows the results achieved on Cantonese (the character error rate is used for evaluation rather than the word error rate). The achieved CER is again 0.1% above the best error rate achieved using a statically tuned beam, but at an RTF which is more than 2 times faster than the static beam at equal precision.

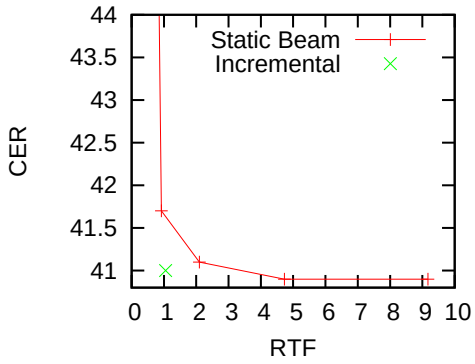

Fig. 7. *Babel Cantonese.*

Figure 8 shows the results achieved on Vietnamese (the evaluated value is the token error rate here, because the system uses morphemes rather than full words). The incremental decoding achieves the best possible TER at an RTF which is more than 3 times faster than the statically tuned beam.

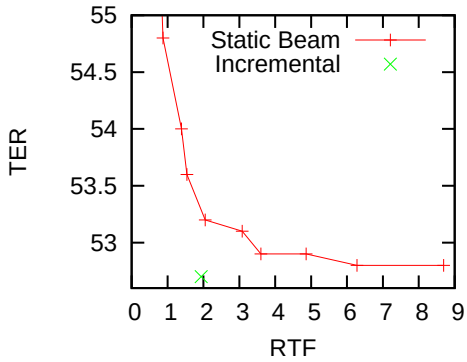

Fig. 8. *Babel Vietnamese.*

Figure 9 shows the results achieved on Pashto. The achieved WER is 0.1% above the optimum, however at a considerably faster RTF than the static beam at equal precision.

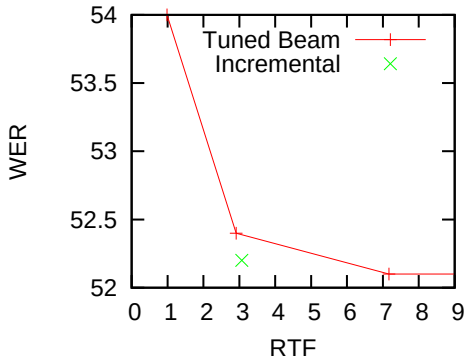

Fig. 9. *Babel Pashto.*

Figure 10 shows the results achieved on Tagalog. The

achieved WER is 0.1% above the optimum, however at an RTF which is 2 to 3 times faster than the static beam at equal precision.

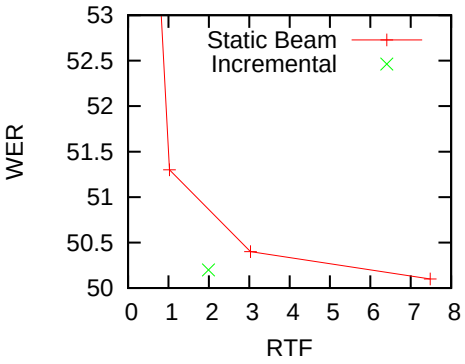

Fig. 10. *Babel Tagalog.*

We see that we can deterministically achieve a precision very close to the optimum, without knowing the reference, and at an RTF which faster than the static beam tuned for the same precision. This may be surprising, because we decode everything at least twice, once forward and once backward. However, two fast decodings with a tight beam can be faster than one slow decoding with a large beam, and since most speech is already recognized correctly with a relatively tight beam, we can save a lot of effort by only applying a large beam to those portions of speech which really need it.

10. CONCLUSIONS

- We have shown that most search errors can be detected by aligning a symmetric forward and backward decoding pass. There are plenty of applications for unsupervised search error detection, for example for unsupervised tuning of decoder beam sizes without using a specific development corpus.
- We have introduced a decoding architecture which exploits automatic search error detection to efficiently achieve a precision very close to the global optimum, but without requiring a development corpus for tuning, without any human intervention, and at an RTF around two times faster than when trying to achieve the same precision using statically tuned beam sizes (four times faster when using separate threads for the forward and backward search). Each segment, and even each individual word, is recognized with just the right beam size required for correct recognition.
- Such an architecture is especially useful when rapidly developing new ASR systems, when time constraints do not allow exhaustive tuning of decoder parameters, when no development corpus is available, if a significant mismatch between the development and the evaluation corpus can be expected which would lead to a mis-tuning of the decoder parameters, or simply if the person operating the decoder is not able or not willing to properly tune it.
- On a higher level we have shown that search with virtually no search errors is possible under common time and memory constraints, even when following the classical decoding approach with fixed beams. On most systems the incremental decoding could successfully eliminate all detectable search errors on more than 99.7% of all segments under the given histogram pruning constraints.

11. ACKNOWLEDGEMENTS

This work was partly realized under the Quaero Programme, funded by OSEO, French State agency for innovation.

Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD / ARL) contract number W911NF-12-C-0012.[1]

12. REFERENCES

[1] D. Nolden, R. Schlüter, and H. Ney, "Extended Search Space Pruning in LVCSR," 2012, ICASSP.

[2] D. Nolden, R. Schlüter, and H. Ney, "Search Space Pruning Based on Anticipated Path Recombination in LVCSR," in *Interspeech*, Portland, OR, USA, September 2012.

[3] L. Nguyen, R. Schwartz, and P. Placeway, "Search algorithms for software-only real-time recognition with very large vocabularies," in *Proceedings of the Workshop on Human Language Technology*, 1993, pp. 91–95.

[4] M. Hannemann, D. Povey, and G. Zweig, "Combining forward and backward search in decoding," in *ICASSP*, Vancouver, Canada, May 2013, pp. 6739–6743.

[5] D. Nolden, R. Schlüter, and H. Ney, "Acoustic Look-Ahead for More Efficient Decoding in LVCSR," in *Interspeech*, Florence, Italy, August 2011.

[6] Wafi Abo-Gannemhy, Itshak Lapidot, and H. Guterman, "Speech recognition using combined forward and backward Viterbi search," in *IEEE Convention of the Electrical and Electronic Engineers in Israel*, 2010.

[7] M. Tang and P. Di Cristo, "Backward viterbi beam search for utilizing dynamic task complexity information," in *Interspeech*, Brisbane, Australia, September 2008.

[8] Ta Li, Changchun Bao, Weiqun Xu, Jielin Pan, and Yonghong Yan, "Improving voice search using forward-backward lvcsr system combination," in *The Sixth International Symposium on Neural Networks (ISNN 2009)*, vol. 56 of *Advances in Intelligent and Soft Computing*, pp. 769–777. Springer Berlin Heidelberg, 2009.

[9] D. Nolden, D. Rybach, R. Schlüter, and H. Ney, "Joining Advantages of Word-Conditioned and Token-Passing Decoding," 2012, ICASSP.

[10] S. Ortmanns, H. Ney, and X. Aubert, "A Word Graph Algorithm for Large Vocabulary Continuous Speech Recognition," January 1997, vol. 11, pp. 43 – 72, Computer, Speech and Language.

[11] B. Kingsbury, J. Cui, X. Cui, M. Gales, K. Knill, J. Mamou, L. Mangu, D. Nolden, M. Picheny, B. Ramabhadran, R. Schlüter, A. Sethy, and P. Woodland, "A high-performance cantonese keyword search system," in *ICASSP*, Vancouver, Canada, May 2013, pp. 8277–8281.

[12] Zoltán Tüske, Ralf Schlüter, and Hermann Ney, "Deep hierarchical bottleneck MRASTA features for LVCSR," in *ICASSP*, Vancouver, Canada, May 2013, pp. 6970–6974.

[1]The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.