A Fast Re-scoring Strategy to Capture Long-Distance Dependencies

Anoop Deoras
HLT-COE and CLSP
Johns Hopkins University
Baltimore MD 21218, USA
adeoras@jhu.edu,

Tomáš Mikolov
Brno University of Technology
Speech@FIT
Czech Republic
imikolov@fit.vutbr.cz,

Kenneth Church
HLT-COE and CLSP
Johns Hopkins University
Baltimore MD 21218, USA
kenneth.church@jhu.edu

Abstract

A re-scoring strategy is proposed that makes it feasible to capture more long-distance dependencies in the natural language. Two pass strategies have become popular in a number of recognition tasks such as ASR (automatic speech recognition), MT (machine translation) and OCR (optical character recognition). The first pass typically applies a weak language model (n-grams) to a lattice and the second pass applies a stronger language model to N best lists. The stronger language model is intended to capture more long-distance dependencies. The proposed method uses RNN-LM (recurrent neural network language model), which is a long span LM, to re-score word lattices in the second pass. A hill climbing method (iterative decoding) is proposed to search over *islands of confusability* in the word lattice. An evaluation based on Broadcast News shows speedups of 20 over basic N best re-scoring, and word error rate reduction of 8% (relative) on a highly competitive setup.

1 Introduction

Statistical Language Models (LMs) have received considerable attention in the past few decades. They have proved to be an essential component in many statistical recognition systems such as ASR (automatic speech recognition), MT (machine translation) and OCR (optical character recognition). The task of a language model is to assign probability to any word sequence possible in the language. The probability of the word sequence $W \equiv w_1, \ldots, w_m \equiv w_1^m$ is typically factored using the chain rule:

$$P(w_1^m) = \prod_{i=1}^{m} P(w_i | w_1^{i-1}) \quad (1)$$

In modern statistical recognition systems, an LM tends to be restricted to simple n-gram models, where the distribution of the predicted word depends on the previous $(n-1)$ words i.e. $P(w_i|w_1^{i-1}) \approx P(w_i|w_{i-n+1}^{i-1})$.

Noam Chomsky argued that n-grams cannot learn long-distance dependencies that span over more than n words (Chomsky, 1957, pp.13). While that might seem obvious in retrospect, there was a lot of excitement at the time over the Shannon-McMillan-Breiman Theorem (Shannon, 1948) which was interpreted to say that, in the limit, under just a couple of minor caveats and a little bit of not-very-important fine print, n-gram statistics are sufficient to capture all the information in a string (such as an English sentence). Chomsky realized that while that may be true in the limit, n-grams are far from the most parsimonious representation of many linguistic facts. In a practical system, we will have to truncate n-grams at some (small) fixed n (such as trigrams or perhaps 5-grams). Truncated n-gram systems can capture many agreement facts, but not all.[1]

By long-distance dependencies, we mean facts like agreement and collocations that can span over many words. With increasing order of n-gram models we can, in theory, capture more regularities in the

[1]The discussion in this paragraph is taken as-is from an article (to appear) by Church (2012).

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1116–1127,
Edinburgh, Scotland, UK, July 27–31, 2011.

language. In addition, if we can move to more general models then we could hope to capture more, as well. However, due to data sparsity, it is hard to estimate a robust n-gram distribution for large values of n (say, $n > 10$) using the conventional Maximum Likelihood techniques, unless a more robust technique is employed for modeling which generalizes well on unseen events. Some of these well known long span / complex language models which have shown to perform very well on many speech tasks include: structured language model (Chelba and Jelinek, 2000; Roark, 2001; Wang and Harper, 2002; Filimonov and Harper, 2009), latent semantic analysis language model (Bellegarda, 2000), topic mixture language models (Iyer and Ostendorf, 1999), whole sentence exponential language models (Rosenfeld, 1997; Rosenfeld et al., 2001), feedforward neural networks (Bengio et al., 2001), recurrent neural network language models (Mikolov et al., 2010), among many others.

Although better modeling techniques can now capture longer dependencies in a language, their incorporation in decoders of speech recognition or machine translation systems becomes computationally challenging. Due to the prohibitive increase in the search space of sentence hypotheses (or longer length word sub sequences), it becomes challenging to use a long span language model in the first pass decoding. A word graph (word lattices for speech recognition systems and hypergraphs for machine translation systems), encoding exponential number of hypotheses is hence outputted at the first pass output on which a sophisticated and complex language model is deployed for re-scoring. However, sometimes even re-scoring of this refined search space can be computationally expensive due to explosion of state space.

Previously, we showed in (Deoras et al., 2011) how to tackle the problem of incorporating long span information during decoding in speech recognition systems by variationaly approximating (Bishop, 2006, pp. 462) the long span language model by a tractable substitute such that this substitute model comes closest to the long span model (closest in terms of Kullback Leibler Divergence (Cover and J.A.Thomas, 1991, pp. 20)). The tractable substitute was then used directly in the first pass speech recognition systems. In this paper we propose an approach that keeps the model intact but approximates the search space instead (which can become intractable to handle especially under a long span model), thus enabling the use of full blown model for re-scoring.With this approach, we can achieve full lattice re-scoring with a complex model, at a cost more than 20 times less than of a naive brute force approach that is commonly used today.

The rest of the paper is organized as follows: We discuss a particular form of long span language model in Sec. 2. In Sec. 3 we discuss two standard re-scoring techniques and then describe and demonstrate our proposed technique in Sec. 4. We present experimental results in Sec. 5 followed by conclusions and some remarks in Sec. 6.

2 Recurrent Neural Networks (RNN)

There is a long history of using neural networks to model sequences. Elman (1990) used recurrent neural network for modeling sentences of words generated by an artificial grammar. Work on statistical language modeling of real natural language data, together with an empirical comparison of performance to standard techniques was done by Bengio et al. (2001). His work has been followed by Schwenk (2007), who has shown that neural network language models actually work very well in the state-of-the-art speech recognition systems. Recurrent Neural Network based Language Models (RNN-LMs) (Mikolov et al., 2010) improved the ability of the original model to capture patterns in the language without using any additional features (such as part of speech, morphology etc) i.e. other than lexical ones. The RNN-LM was shown to have superior performance than the original feedforward neural network (Mikolov et al., 2011b). Recently, we also showed that this model outperforms many other advanced language modeling techniques (Mikolov et al., 2011a). We hence decided to work with this model. This model uses whole history to make predictions, thus it lies outside the family of n-gram models. Power of the model comes at a considerable computational cost. Due to the requirement of unlimited history, many optimization tricks for rescoring with feedforward-based NNLMs as presented by Schwenk (2007) cannot be applied during rescoring with RNN LM. Thus, this model is a good candidate

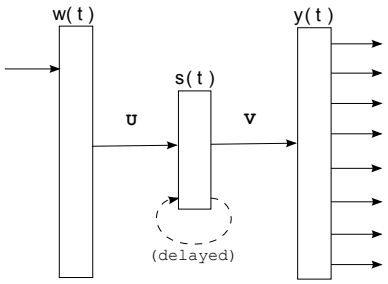

Figure 1: Schematic Representation of Recurrent Neural Network Language Model. The network has an input layer $\mathbf{w}$, a hidden layer $\mathbf{s}$ and an output layer $\mathbf{y}$. Matrices U and V represent synapses.

to show effectiveness and importance of our work.

The basic RNNLM is shown in Fig. 1. The model has an input layer $\mathbf{w}(t)$ that encodes previous word using 1 of N coding (thus, the size of the input layer is equal to the size of the vocabulary, and only the neuron that corresponds to the previous word in a sequence is set to 1). The hidden layer $\mathbf{s}(t)$ has additional recurrent connections that are delayed by one time step. After the network is trained, the output layer $\mathbf{y}(t)$ represents probability distribution for the current word, given the previous word and the state of the hidden layer from the previous time step.

The training is performed by 'backpropagation-through-time' algorithm that is commonly used for training recurrent neural networks (Rumelhart et al., 1986). More details about training, setting initial parameters, choosing size of the hidden layer etc. are presented in (Mikolov et al., 2010). Additional extensions that allow this model to be trained on large corpora are presented in (Mikolov et al., 2011b).

3 Standard Approaches for Rescoring

3.1 Word Lattice Rescoring

A word lattice, $\mathcal{L}$, obtained at the output of the first pass decoding, encodes exponential number (exponential in the number of states (nodes) present in the lattice) of hypotheses in a very compact data structure. It is a directed acyclic graph $G = (\mathcal{V}, \mathcal{E}, n_s, N_e)$, where $\mathcal{V}$ and $\mathcal{E}$ denote set of vertices (nodes / states) and edges (arcs / links), respectively. n_s and N_e denote the unique start state and set of end states.

A path, π, in a lattice is an element of $\mathcal{E}^*$ with consecutive transitions. We will denote the origin / previous state of this path by $p[\pi]$ and destination / next state of this path by $n[\pi]$. A path, π is called a *complete path* if $p[\pi] = n_s$ and $n[\pi] \in N_e$. A path, π, is called a *partial path* if $p[\pi] = n_s$ but $n[\pi]$ may or may not belong to N_e. A path, π, is called a *trailing path* if $p[\pi]$ may or may not be equal to n_s and $n[\pi] \in N_e$. We will also denote the time stamp at the start of the path by $T_s[\pi]$ and the time stamp at the end of the path by $T_e[\pi]$. Since there are nodes attached to the start and end of any path, we will denote the time stamp at any node $u \in \mathcal{V}$ by $T[u]$. Associated with every path, π, is also a word sequence $W[\pi] \in \mathcal{W}^*$, where $\mathcal{W}$ is the vocabulary used during speech recognition. For the sake of simplicity, we will distinguish word sequence of length 1 from the word sequences of length greater than 1 by using lower and upper casing i.e. $w[\cdot]$ and $W[\cdot]$ respectively.

The acoustic likelihood of the path $\pi \in \mathcal{E}^*$ is then given as:

$$A[\pi] = \prod_{j=1}^{|\pi|} P(\mathbf{a}_j | w[\pi_j])$$

where $\forall j \in \{1, 2, \ldots, |\pi|\} \quad \pi_j \in \mathcal{E}$, $\pi = \odot_{j=1}^{|\pi|} \pi_j$ and $P(\mathbf{a}_j | w[\pi_j])$ is the acoustic likelihood of the acoustic substring $\mathbf{a}_j$, spanning between $T_s[\pi_j]$ and $T_e[\pi_j]$, conditioned on the word $w[\pi_j]$ associated with the edge π_j.[2] Similarly, the language model score of the path π is given as:

$$L[\pi] = \prod_{j=1}^{|\pi|} P(w[\pi_j] | w[\pi_{j-1}], \ldots, w[\pi_{j-m+1}])$$

where $P(w[\pi_j] | w[\pi_{j-1}], \ldots, w[\pi_{j-m+1}])$ is the m-th order Markov approximation for estimating the probability of a word given the context upto that point. The speech recognizer, which uses m-th order Markov LM for first pass recognition, imposes a constraint on the word lattice such that at each state there exists an unambiguous context of consecutive $m-1$ words.

A first pass output is then a path π^* having Maximum a Posterior (MAP) probability.[3] Thus π^* is

[2] We will use $\odot$ symbol to denote concatenation of paths or word strings.

[3] Note that asterisk symbol here connotes that the path is *op-*

obtained as:

$$\pi^* = \underset{\substack{\pi: p[\pi]=n_s \\ n[\pi]\in N_e}}{\arg\max} A[\pi]^\gamma L[\pi],$$

where γ is the scaling parameter needed to balance the dynamic variability between the distributions of acoustic and language model (Ogawa et al., 1998). Efficient algorithms such as single source shortest path (Mohri et al., 2000) can be used for finding out the MAP path.

Under a new n-gram Language Model, rescoring involves replacing the existing language model scores of all paths π. If we denote the new language model by L_{new} and correspondingly the score of the path π by $L_{new}[\pi]$, then it is simply obtained as:

$$L_{new}[\pi] = \prod_{j=1}^{|\pi|} P(w[\pi_j]|w[\pi_{j-1}], \ldots, w[\pi_{j-n+1}])$$

where $P(w[\pi_j]|w[\pi_{j-1}], \ldots, w[\pi_{j-n+1}])$ is the n-th order Markov approximation for estimating the probability of a word given the unambiguous context of $n-1$ words under the new rescoring LM. If the Markov rescoring n-gram LM needs a bigger context for the task of prediction (i.e. $n > m$, where $m-1$ is the size of the unambiguous context maintained at every state of the word lattice), then each state of the lattice has to be split until an unambiguous context of length as large as that required by the new re-scoring language model is not maintained. The best path, π^* is then obtained as:

$$\pi^* = \underset{\substack{\pi: p[\pi]=n_s \\ n[\pi]\in N_e}}{\arg\max} A[\pi]^\eta L_{new}[\pi],$$

where η acts as the new scaling parameter which may or may not be equal to the old scaling parameter γ.

It should be noted that if the rescoring LM needs a context of the entire past in order to predict the next word, then the lattice has to be expanded by splitting the states many more times. This usually blows up the search space even for a reasonably small number of state splitting iterations. When the task is to do rescoring under a long span LM, such as RNN-LM, then *exact* lattice re-scoring option is not feasible. In order to tackle this problem, a suboptimal approach via N best list rescoring is utilized. The details of this method are presented next.

3.2 N best List Rescoring

N best list re-scoring is a popular way to capture some long-distance dependencies, though the method can be slow and it can be biased toward the weaker language model that was used in the first pass.

Given a word lattice, $\mathcal{L}$, top N paths $\{\pi_1, \ldots, \pi_N\}$ are extracted such that their joint likelihood under the baseline acoustic and language models are in descending order i.e. that:

$$A[\pi_1]^\gamma L[\pi_1] \geq A[\pi_2]^\gamma L[\pi_2] \geq \ldots \geq A[\pi_N]^\gamma L[\pi_N]$$

Efficient algorithms exist for extracting N best paths from word lattices (Chow and Schwartz, 1989; Mohri and Riley, 2002). If a new language model, L_{new}, is provided, which now need not be restricted to finite state machine family, then that can be deployed to get the score of the entire path π. If we denote the new LM scores by $L_{new}[\cdot]$, then under N best list paradigm, optimal path $\tilde{\pi}$ is found out such that:

$$\tilde{\pi} = \underset{\pi\in\{\pi_1,\ldots,\pi_N\}}{\arg\max} A[\pi]^\eta L_{new}[\pi], \qquad (2)$$

where η acts as the new scaling parameter which may or may not be equal to γ. If $N \ll |\mathcal{L}|$ (where $|\mathcal{L}|$ is the total number of complete paths in word lattice, which are exponentially many), then the path obtained using (2) is not guaranteed to be optimal (under the rescoring model). The short list of hypotheses so used for re-scoring would yield suboptimal output if the best path π^* (according to the new model) is not present among the top N candidates extracted from the lattice. This search space is thus said to be *biased* towards a weaker model mainly because the N best lists are representative of the model generating them. To illustrate the idea, we demonstrate below a simple analysis on a relatively easy task of speech transcription on WSJ data.[4] In this setup, the recognizer made use of a bi-

timal under some model. This should not be confused with the Kleene stars appearing as superscripts for $\mathcal{E}$ and $\mathcal{W}$, which serve the purpose of regular expressions implying 0 or many occurances of the element of $\mathcal{E}$ and $\mathcal{V}$ respectively.

[4]Full details about the setup can be found in (Deoras et al., 2010)

gram LM to produce lattices and hence N best lists. Each hypothesis in this set got a rank with the top most and highest scoring hypothesis getting a rank of 1, while the bottom most hypothesis getting a rank of N. We then re-scored these hypotheses with a better language model (either with a higher order Markov LM i.e. a trigram LM (*tg*) or the log linear combination of n-gram models and syntactic models (*n-gram+syntactic*) and re-ranked the hypotheses to obtain their new ranks. We then used Spearman's rank correlation factor, ρ, which takes values in $[-1, +1]$, with -1 meaning that the two ranked lists are negatively correlated (one list is in a reverse order with respect to the other list) and $+1$ meaning that the two ranked lists are positively correlated (the two lists are exactly the same). Spearman's rank correlation factor is given as:

$$\rho = 1 - \frac{6 \sum_{n=1}^{N} d_n^2}{N(N^2 - 1)}, \quad (3)$$

where d_n is the difference between the old and new rank of the n^{th} entry (in our case, difference between $n (\in \{1, 2, \ldots, N\})$ and the new rank which the n^{th} hypothesis got under the rescoring model).

Table 1 shows how the correlation factor drops dramatically when a better and a complementary LM is used for re-scoring, suggesting that the N best lists are heavily biased towards the starting models. Huge re-rankings suggests there is an opportunity to improve and also a need to explore more hypotheses, i.e. beyond N best lists.

Model	(ρ)	**WER (%)**
bg	1.00	18.2%
tg	0.41	17.4%
n-gram+syntactic	0.33	15.8%

Table 1: Spearman Rank Correlation on the N best list extracted from a bi-gram language model (*bg*) and re-scored with relatively better language models including, trigram LM (*tg*), and the log linear combination of n-gram models, and syntactic models (*n-gram+syntactic*). With a bigger and a better LM, the WER decreases at the expense of huge re-rankings of N best lists, only suggesting the fact that N best lists generated under a weaker model, are not reflective enough of a relatively better model.

In the next section, we propose an algorithm which keeps the representation of search space as simple as that of N best list, but does not restrict itself to top N best paths alone and hence does not get *biased* towards the starting weaker model.

4 Proposed Approach for Rescoring

A high level idea of our proposed approach is to identify *islands of confusability* in the word lattice and replace the problem of global search over word lattice by series of local search problems over these islands in an iterative manner. The motivation behind this strategy is the observation that the recognizer produces bursts of errors such that they have a temporal scope. The recognizer output (sentence hypotheses) when aligned together typically shows a pattern of confusions both at the word level and at the phrase level. Regions where there are singleton words competing with one another (reminiscent of a confusion bin of a Confusion Network (CN) (Mangu, 2000)), choice of 1 word edit distance works well for the formation of local neighborhood. Regions where there are phrases competing with other phrases, choice of variable length neighborhood works well. Previously, Richardson et al. (1995) demonstrated a hill climbing framework by exploring 1 word edit distance neighborhood, while in our own previous work (Deoras and Jelinek, 2009), we demonstrated working of iterative decoding algorithm, a hill climbing framework, for CNs, in which the neighborhood was formed by all words competing with each other in any given time slot, as defined by a confusion bin.

In this work, we propose a technique which generalizes very well on word lattices and overcomes the limitations posed by a CN or by the limited nature of local neighborhood. The size of the neighborhood in our approach is a *variable* factor which depends upon the confusability in any particular region of the word lattice. Thus the local neighborhood are in some sense a function of the confusability present in the lattice rather than some predetermined factor. Below we describe the process, virtue of which, we can cut the lattice to form many self contained smaller sized sub lattices. Once these sub lattices are formed, we follow a similar hill climbing procedure as proposed in our previous work (Deoras and Jelinek, 2009).

4.1 Islands of Confusability

We will continue to follow the notation introduced in section 3.1. Before we define the procedure for cutting the lattice into many small self contained lattices, we will define some more terms necessary for the ease of understandability of the algorithm.[5] For any node $v \in \mathcal{V}$, we define forward probability, $\alpha(v)$, as the probability of any partial path $\pi \in \mathcal{E}^*$, s.t. $p[\pi] = n_s, n[\pi] = v$ and it is given as:

$$\alpha(v) = \sum_{\substack{\pi \in \mathcal{E}^* \\ s.t. p[\pi]=n_s, n[\pi]=v}} A[\pi]^\gamma L[\pi] \qquad (4)$$

Similarly, for any node $v \in \mathcal{V}$, we define the backward probability, $\beta(v)$, as the probability of any trailing path $\pi \in \mathcal{E}^*$, s.t. $p[\pi] = v, n[\pi] \in N_e$ and it is given as:

$$\beta(v) = \sum_{\substack{\pi \in \mathcal{E}^* \\ s.t. p[\pi]=v, n[\pi] \in N_e}} A[\pi]^\gamma L[\pi] \qquad (5)$$

If we define the sum of joint likelihood under the baseline acoustic and language models of all paths in the lattice by Z, then it can simply be obtained as: $Z = \sum_{u \in N_e} \alpha(u) = \beta(n_s)$

In order to cut the lattice, we want to identify sets of nodes, $S_1, S_2, \ldots, S_{|\mathcal{S}|}$ such that for any set $S_i \in \mathcal{S}$ following conditions are satisfied:

1. For any two nodes $u, v \in S_i$ we have that: $T[u] = T[v]$. We will define this common time stamp of the nodes in the set by $T[S_i]$.
2. $\nexists\, \pi \in \mathcal{E}$ such that $T_s[\pi] < T[S_i] < T_e[\pi]$.

The first property can be easily checked by first pushing states into a linked list associated with each time marker (this can be done by iterating over all the states of the graph) then iterating over the unique time markers and retrieving back the nodes associated with it.The second property can be checked by first iterating over the unique time markers and for each of the marker, iterating over the arcs and terminating the loop as soon as some arc is found out violating property 2 for the specific time marker. Thus the time complexity for checking property 1 is $O(|\mathcal{V}|)$ and that for property 2 is $O(|\mathcal{T}| \times |\mathcal{E}|)$, where $|\mathcal{T}|$ is the total number of unique time markers. Usually $|\mathcal{T}| \ll |\mathcal{E}|$ and hence the time complexity for checking property 2 is almost linear in the number of edges. Thus effectively, the time complexity for cutting the lattice is $O(|\mathcal{V}| + |\mathcal{E}|)$.

Having formed such sets, we can now cut the lattice at time stamps associated with these sets i.e. that: $T[S_1], \ldots, T[S_{|\mathcal{S}|}]$. It can be easily seen that the number of sub lattices, C, will be equal to $|\mathcal{S}| - 1$.We will identify these sub lattices as $\mathcal{L}_1, \mathcal{L}_1, \ldots, \mathcal{L}_C$. At this point, we have not formed self contained lattices yet by simply cutting the parent lattice at the cut points.

Once we cut the lattice at these cut points, we implicitly introduce many new starting nodes and ending nodes for any sub lattice. We will refer to these nodes as exposed starting nodes and exposed ending nodes. Thus for some j^{th} sub lattice, $\mathcal{L}_j$, there will be as many new exposed starting nodes as there are nodes in the set S_j and as many exposed ending nodes as there are nodes in the set S_{j+1}. In order to make these sub lattices consistent with the definition of a word lattice (see Sec. 3.1), we unify all the exposed starting nodes and exposed ending nodes. To unify the exposed starting nodes, we introduce as many *new* edges as there are nodes in the set S_j such that they have a common starting node, $n_s[\mathcal{L}_j]$, (newly created) and distinct ending nodes present in S_j. To unify the exposed ending nodes of $\mathcal{L}_j$, we introduce as many *new* edges as there are nodes in the set S_{j+1} such that they have distinct starting nodes present in S_{j+1} and a common ending node $n_e[\mathcal{L}_j]$ (newly created). From the totality of these new edges and nodes along with the ones already present in $\mathcal{L}_j$ forms an induced directed acyclic subgraph $G[\mathcal{L}_j] = (\mathcal{V}[\mathcal{L}_j], \mathcal{E}[\mathcal{L}_j], n_s[\mathcal{L}_j], n_e[\mathcal{L}_j])$.

For any path $\pi \in \mathcal{E}[\mathcal{L}_j]$ such that $p[\pi] = n_s[\mathcal{L}_j]$ and $n[\pi] \in S_j$, we assign the value of $\alpha(n[\pi])$ to denote the joint likelihood $A[\pi]^\gamma L[\pi]$ and assign $epsilon$ for word associated with these edges i.e. $w[\pi]$. We assign $T[S_j] - \delta T$ to denote $T_s[\pi]$ and $T[S_j]$ to denote $T_e[\pi]$. Similarly, for any path $\pi \in \mathcal{E}[\mathcal{L}_j]$ such that $p[\pi] \in S_{j+1}$ and $n[\pi] = n_e[\mathcal{L}_j]$,

[5] Goel and Byrne (2000) previously demonstrated the lattice segmentation procedure to solve the intractable problem of MBR decoding. The cutting procedure in our work is different from theirs in the sense that we rely on time information for collating competing phrases, while they do not.

we assign the value of $\beta(p[\pi])$[6] to denote the joint likelihood $A[\pi]^{\gamma}L[\pi]$ and assign $epsilon$ for word associated with these edges i.e. $w[\pi]$. We assign $T[S_{j+1}]$ to denote $T_s[\pi]$ and $T[S_{j+1}] + \delta T$ to denote $T_e[\pi]$. This completes the process and we obtain self contained lattices, which if need be, can be independently decoded and/or analyzed.

4.2 Iterative Decoding on Word Lattices

Once we have formed the self contained lattices, $\mathcal{L}_1, \mathcal{L}_1, \ldots, \mathcal{L}_C$, where C is the total number of sub lattices formed, then the idea is to divide the re-scoring problem into many small re-scoring problems carried over the sub lattices one at a time by fixing single best paths from all the remaining sub lattices.

The inputs to the algorithm are the sub lattices (produced by cutting the parent lattice generated under some Markov n-gram LM) and a new rescoring LM, which now need not be restricted to finite state machine family. The output of the algorithm is a word string, $\mathbf{W}^*$, such that it is the concatenation of final decoded word strings from each sub lattice. Thus if we denote the final decoded path (under some decoding scheme, which will become apparent next) in the j^{th} sub lattice by π_j^* and the concatenation symbol by '$\cdot$', then $\mathbf{W}^* = W[\pi_1^*] \cdot W[\pi_2^*] \cdot \ldots \cdot W[\pi_C^*] = \odot_{j=1}^{C} W[\pi_j^*]$.

Algorithm 1 Iterative Decoding on word lattices.

Require: $\{\mathcal{L}_1, \mathcal{L}_1, \ldots, \mathcal{L}_C\}$, L_{new}
  PrevHyp $\leftarrow$ null
  CurrentHyp $\leftarrow \odot_{j=1}^{C} W[\hat{\pi}_j]$
  while PrevHyp $\neq$ CurrentHyp **do**
    for $i \leftarrow 1 \ldots C$ **do**

$$\hat{\pi}_i \leftarrow \underset{\substack{\pi_i \in \mathcal{E}_i^*: \\ p[\pi_i]=n_s[\mathcal{L}_i] \\ n[\pi_i]=n_e[\mathcal{L}_i]}}{\operatorname{argmax}} \Big(L_{new}[\hat{\pi_1} \cdot \ldots \cdot \pi_i \cdot \ldots \cdot \hat{\pi_k}] \times A[\pi_i]^{\eta} \prod_{\substack{j=1 \\ j \neq i}}^{k} A[\hat{\pi_j}]^{\eta} \Big)$$

    end for
    PrevHyp $\leftarrow$ CurrentHyp
    CurrentHyp $\leftarrow \odot_{j=1}^{C} W[\hat{\pi}_j]$
  end while
  $\forall j \in \{1, 2, \ldots, C\} \quad \pi_j^* \leftarrow \hat{\pi}_j$

[6]The values of $\alpha(\cdot)$ and $\beta(\cdot)$ are computed under parent lattice structure.

The algorithm is initialized by setting **PrevHypo** to null and **CurrHypo** to the concatenation of 1-best output from each sub lattice. During the initialization step, each sub lattice is analyzed independent of any other sub lattice and under the baseline acoustic scores and baseline n-gram LM scores, 1-*best* path is found out. Thus if we define the best path under baseline model in some j^{th} sub-lattice by $\hat{\pi}_j$, **CurrHypo** is then initialized to: $W[\hat{\pi}_1] \cdot W[\hat{\pi}_2] \cdot \ldots \cdot W[\hat{\pi}_C]$. The algorithm then runs as long as CurrHypo is not equal to PrevHypo. In each iteration, the algorithm sequentially re-scores each sub-lattice by keeping the surrounding context fixed. Once all the sub lattices are re-scored, that constitutes one iteration. At the end of each iteration, CurrHypo is set to the concatenation of 1 best paths from each sub lattice while PrevHypo is set to the old value of CurrHypo. Thus if we are analyzing some i^{th} sub-lattice in some iteration, then 1-best paths from all but this sub-lattice is kept fixed and a *new* 1-*best* path under the re-scoring LM is found out. It is not hard to see that the likelihood of the output under the new re-scoring model is guaranteed to increase monotonically after every decoding step.

Since the cutting of parent lattices produce many small lattices with considerably lesser number of nodes, in practice, an exhaustive search for the 1-best hypothesis can be carried out via N best list. Algorithm 1 outlines the steps for iterative decoding on word lattices.

4.3 Entropy Pruning

In this section, we will discuss a speed up technique based on entropy of the lattice. Entropy of a lattice reflects the confidence of the recognizer in recognizing the acoustics. Based on the observation that if the N best list / lattice generated under some model has a very low entropy, then the Spearman's rank correlation factor, ρ (Eqn. 3), tends to be higher even when the N best lists / lattice is re-ranked with a bigger and a better model. A low entropy under the baseline model only reflects the confidence of the recognizer in recognizing the acoustic. Table 2 shows the rank correlation values between two sets of N best lists. Both sets are produced by a bigram LM (bg). The entropy of N best lists in the first set is 0.05 nats or less. The N best lists in the second set have an entropy greater than 0.05 nats.

Both these sets are re-ranked with bigger and better models (see Table 1 for model definitions). We can see from Table 2 that the rank correlation values tend to be higher (indicating little re-rankings) when the entropy of the N best list, under the baseline model, is lower. Similarly, the rank-correlation values tend to be lower (indicating more re-rankings) whenever the entropy of the N best list is higher. Note that these entropy values are computed with respect to the starting model (in this case, bigram LM). Of course, if the starting LM is much weaker than the rescoring model, then the entropy values need not be reflective of the difficulty of the overall task. This observation then suggests that it is safe to rescore only those N best lists whose entropy under the starting model is higher than some threshold.

Rescoring Model	$\rho_{(H \leq 0.05)}$	$\rho_{(H > 0.05)}$
bg	1.00	1.00
tg	0.58	0.38
n-gram+syntactic	0.54	0.31

Table 2: Spearman Rank Correlation on the N best list extracted from a bi-gram language model (*bg*) and rescored with relatively better language models (see Table 1 for model definitions). Entropy under the baseline model correlates well with the rank correlation factor, suggesting that exhaustive search need not be necessary for utterances yielding lower entropy.

While computation of entropy for N best list is tractable, for a word lattice, the computation of entropy is intractable if one were to enumerate all the hypotheses. Even if we were able to enumerate all hypotheses, this method tends to be slower. Using efficient semiring techniques introduced by Li and Eisner (2009) or using posterior probabilities on the edges leading to end states, we can compute the entropy of a lattice in one single forward pass using dynamic programming. It should, however, be noted that, for dynamic programming technique to work, only n-gram LMs can be used. One has to resort to approximate entropy computation via N best list, if entropy under long span LM is desired.

4.3.1 Speed Up for Iterative Decoding

Our speed up technique is simple. Once we have formed self contained sub lattices, we want to prune all but the top few best complete paths (obtained under baseline / starting model) of those sub lattices whose entropy is below some threshold. Thus, believing in the original model's confidence, we want to focus only on those sub lattices which the recognizer found difficult to decode in the first pass. All other part of the parent lattice will be not be analyzed. The thresholds for pruning is very application and corpus specific and needs to be tuned on some held out data.

5 Experiments and Results

We performed recognition on the Broadcast News (BN) dev04f, rt03 and rt04 task using the state-of-the-art acoustic models trained on the English Broadcast News (BN) corpus (430 hours of audio) provided to us by IBM (Chen et al., 2009). IBM also provided us its state-of-the-art speech recognizer, Attila (Soltau et al., 2010) and two Kneser-Ney smoothed backoff n-gram LMs containing 4.7M n-grams ($n \leq 4$) and 54M n-grams ($n \leq 4$), both trained on 400M word tokens. We will refer to them as KN:BN-Small and KN:BN-Big respectively. We refer readers to (Chen et al., 2009) for more details about the recognizer and corpora used for training the models.

We trained two RNN based language models - the first one, denoted further as RNN-limited, was trained on a subset of the training data (58M tokens). It used 400 neurons in the hidden layer. The second model, denoted as RNN-all, was trained on all of the training data (400M tokens), but due to the computational complexity issues, we had to restrict its hidden layer size to 320 neurons.

We followed IBM's multi-pass decoding recipe using KN:BN-Small in the first pass followed by either N best list re-scoring or word lattice re-scoring using bigger and better models.[7] For the purpose of re-scoring, we combined all the relevant statistical models in one unified log linear framework reminiscent of work by Beyerlein (1998). We, however, trained the model weights by optimizing expected WER rather than 1-*best* loss as described in (Deoras et al., 2010). Training was done on N best lists of size 2K. We will refer to the log linear com-

[7]The choice of the order and size of LM to be used in the first pass decoding was determined by taking into consideration the capabilities of the decoder.

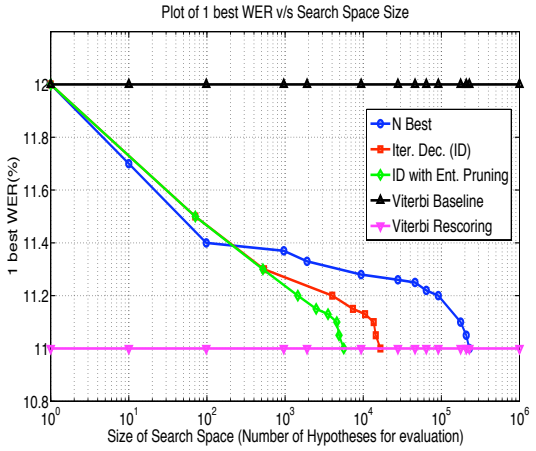

Figure 2: Plot of WER (y axis) on rt03+dev04f set versus the size of the search space (x axis). The baseline WER obtained using KN:BN-Small is 12% which then drops to 11% when KN:BN-Big is used for re-scoring. N best list search method obtains the same reduction in WER by evaluating as many as 228K sentence hypotheses on an average. The proposed method obtains the same reduction by evaluating 14 times smaller search space. The search effort reduces further to 40 times if entropy based pruning is employed during re-scoring.

bination of KN:BN-Big and RNN-limited by KN-RNN-lim; KN:BN-Big and RNN-all by KN-RNN-all and KN:BN-Big, RNN-limited and RNN-all by KN-RNN-lim-all.

We used two sets for decoding: rt03+dev04f set was used as a development set while rt04 was used as a blind set for the purpose of evaluating the performance of long span RNN models using the proposed approach. We made use of OpenFst C++ libraries (Allauzen et al., 2007) for manipulating lattice graphs and generating N best lists. Due to the presence of hesitation tokens in reference transcripts and the need to access the silence/pause tokens for penalizing short sentences, we treated these tokens as regular words before extracting sentence hypotheses. This, and poorly segmented nature of the test corpora, led to huge enumeration of sentence hypotheses.

5.1 n-gram LM for re-scoring

In this setup, we used KN:BN-Small as the baseline starting LM which yielded the WER of 12% on rt03+dev04f set. Using KN:BN-Big as the re-scoring LM, the WER dropped to 11%. Since the re-scoring LM belonged to the n-gram family, it was possible to compute the optimal word string by re-scoring the whole lattice (see Sec. 3.1). We now compare the performance of N best list approach (Sec. 3.2) with our proposed approach (Sec. 4). N best list achieved the best possible reduction by evaluating as many as 228K sentence hypotheses on an average. As against that, our proposed approach achieved the same performance by evaluating 16.6K sentence hypotheses, thus reducing the search efforts by $\mathbf{13.75}$ times. By carrying out entropy pruning (see Sec. 4.3) on sub lattices, our proposed approach required as little as 5.6K sentence hypotheses evaluations to obtain the same optimal performance, reducing the search effort by as much as $\mathbf{40.46}$ times. For the purpose of this experiment, entropy based pruning was carried out when the entropy of the sub lattice was below 5 nats. Table 3 compares the two search methods for this setup and Fig. 2 shows a plot of WER versus the size of the search space (in terms of number of sentence hypotheses evaluated by an n-gram language model).

On rt04, the KN:BN-Small LM gave a WER of 14.1% which then dropped to 13.1% after re-scoring with KN:BN-Big. Since the re-scoring model was an n-gram LM, it was possible to obtain the optimal performance via lattice update technique (see Sec. 3.1). We then carried out the re-scoring of the word lattices under KN:BN-Big using our proposed technique and found it to give the same performance yielding the WER of 13.1%.

5.2 Long Span LM for re-scoring

In this setup, we used the strongest n-gram LM as our baseline. We thus used KN:BN-Big as the baseline LM which yielded the WER of 11% on rt03+dev04f. We then used KN-RNN-lim-all for re-scoring. Due to long span nature of the re-scoring LM, it was not possible to obtain the optimal WER performance. Hence we have compared the performance of our proposed method with N best list approach. N best list achieved the lowest possible WER after evaluating as many as 33.8K sentence hypotheses on an average. As against that, our proposed approach in conjunction with entropy pruning obtained the same performance by evaluating just 1.6K sentence hypotheses, thus reducing the search by a factor of $\mathbf{21}$. Fig 3 shows a plot of WER versus

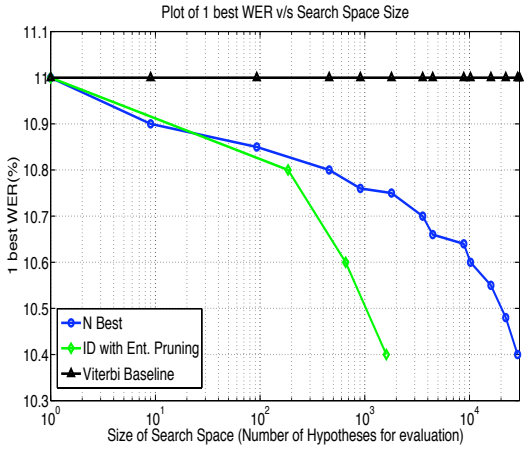

Figure 3: Plot of WER (y axis) on rt03+dev04f set versus the size of the search space (x axis). The baseline WER obtained using KN:BN-Big is 11% which then drops to 10.4% when KN-RNN-lim-all is used for re-scoring. N best list search method obtains this reduction in WER by evaluating as many as 33.8K sentence hypotheses on an average, while the proposed method (with entropy pruning) obtains the same reduction by evaluating 21 times smaller search space.

the size of the search space (in terms of number of sentence hypotheses evaluated by a long span language model).

In-spite of starting off with a very strong n-gram LM, the N best lists so extracted were still not representative enough of the long span rescoring models. Had we started off with KN:BN-Small, the N best list re-scoring method would have had no chance of finding the optimal hypothesis in reasonable size of hypotheses search space. Table 4 compares the two search methods for this setup when many other long span LMs were also used for re-scoring.

On rt04, the KN:BN-Big LM gave a WER of 13.1% which then dropped to **12.15**% after re-scoring with KN-RNN-lim-all using our proposed technique.[8] Since the re-scoring model was not an n-gram LM, it was not possible to obtain the optimal performance but we could enumerate huge N best list to approximate this value. Our proposed method is much faster than huge N best lists and no worse in terms of WER. As far as we know, the result obtained on these sets is the best performance ever reported on the Broadcast News corpus for speech recognition.

[8]The WER obtained using KN-RNN-lim and KN-RNN-all were 12.5% and 12.3% respectively.

Models	WER	NBest	ID	Saving
KN:BN-Small	12.0	-	-	-
KN:BN-Big	11.0	228K	5.6K	40

Table 3: The starting LM is a weak n-gram LM (KN:BN-Small) and the re-scoring LM is a much stronger but n-gram LM (KN:BN-Big). The baseline WER in this case is 12% and the optimal performance by the re-scoring LM is 11.0%. The proposed method outperforms N best list approach, in terms of search efforts, obtaining optimal WER.

Models	WER	NBest	ID	Saving
KN:BN-Big	11.0	-	-	-
KN-RNN-lim	10.5	42K	1.1K	38
KN-RNN-all	10.5	26K	1.3K	20
KN-RNN-lim-all	10.4	34K	1.6K	21

Table 4: The starting LM is a strong n-gram LM (KN:BN-Big) and the re-scoring model is a long span LM (KN-RNN-*). The baseline WER is 11.0%. Due to long span nature of the LM, optimal WER could not be estimated. The proposed method outperfoms N best list approach on every re-scoring task.

6 Conclusion

We proposed and demonstrated a new re-scoring technique for general word graph structures such as word lattices. We showed its efficacy by demonstrating huge reductions in the search effort to obtain a new state-of-the-art performance on a very competitive speech task of Broadcast news. As part of the future work, we plan to extend this technique for hypergraphs and lattices in re-scoring MT outputs with complex and long span language models.

Acknowledgement

This work was partly funded by Human Language Technology, Center of Excellence and by Technology Agency of the Czech Republic grant No. TA01011328, and Grant Agency of Czech Republic project No. 102/08/0707. We would also like to acknowledge the contribution of Frederick Jelinek towards this work. He would be a co-author if he were available and willing to give his consent.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A General and Efficient Weighted Finite-State Transducer Library. In *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23. Springer.

J. R. Bellegarda. 2000. Exploiting latent semantic information in statistical language modeling. *Proceedings of IEEE*, 88(8):1279–1296.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2001. A Neural Probabilistic Language Model. In *Proceedings of Advances in Neural Information Processing Systems*.

Peter Beyerlein. 1998. Discriminative Model Combination. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

Ciprian Chelba and Frederick Jelinek. 2000. Structured Language Modeling. *Computer Speech and Language*, 14(4):283–332.

S. F. Chen, L. Mangu, B. Ramabhadran, R. Sarikaya, and A. Sethy. 2009. Scaling shrinkage-based language models. In *Proc. of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 299–304.

Noam Chomsky. 1957. *Syntactic Structures*. The Hague: Mouton.

Yen-Lu Chow and Richard Schwartz. 1989. The N-Best algorithm: an efficient procedure for finding top N sentence hypotheses. In *Proceedings of the workshop on Speech and Natural Language*, HLT '89, pages 199–202, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kenneth Church. 2012. A Pendulum Swung Too Far. *Linguistic Issues in Language Technology - LiLT*. to appear.

T.M. Cover and J.A.Thomas. 1991. *Elements of Information Theory*. John Wiley and Sons, Inc. N.Y.

Anoop Deoras and Frederick Jelinek. 2009. Iterative Decoding: A Novel Re-Scoring Framework for Confusion Networks. In *Proc. of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 282 –286.

Anoop Deoras, Denis Filimonov, Mary Harper, and Fred Jelinek. 2010. Model Combination for Speech Recognition using Empirical Bayes Risk Minimization. In *Proc. of IEEE Workshop on Spoken Language Technology (SLT)*.

Anoop Deoras, Tomáš Mikolov, Stefan Kombrink, Martin Karafiát, and Sanjeev Khudanpur. 2011. Variational Approximation of Long-Span Language Models for LVCSR. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Jeffery Elman. 1990. Finding Structure in Time. In *Cognitive Science*, volume 14, pages 179–211.

Denis Filimonov and Mary Harper. 2009. A Joint Language Model with Fine-grain Syntactic Tags. In *Proc. of 2009 Conference on Empirical Methods in Natural Language Processing*.

V. Goel and W. Byrne. 2000. Minimum Bayes Risk Automatic Speech Recognition. *Computer, Speech and Language*.

Rukmini Iyer and Mari Ostendorf. 1999. Modeling Long Distance Dependence in Language: Topic Mixtures Versus Dynamic Cache Models. *IEEE Transactions on Speech and Audio Processing*, 7(1):30–39.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 40–51, Singapore, August.

Lidia Luminita Mangu. 2000. *Finding consensus in speech recognition*. Ph.D. thesis, The Johns Hopkins University. Adviser-Brill, Eric.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan "Honza" Černocký, and Sanjeev Khudanpur. 2010. Recurrent Neural Network Based Language Model. In *Proc. of the ICSLP-Interspeech*.

Tomáš Mikolov, Anoop Deoras, Stefan Kombrink, Lukáš Burget, and Jan "Honza" Černocký. 2011a. Empirical Evaluation and Combination of Advanced Language Modeling Techniques. In *Proc. of Interspeech*.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan "Honza" Černocký, and Sanjeev Khudanpur. 2011b. Extensions of Recurrent Neural Network Language Model. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Mehryar Mohri and Michael Riley. 2002. An Efficient Algorithm for the N-Best-Strings Problem. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*.

M. Mohri, F.C.N. Pereira, and M. Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science, 231:17-32*.

A. Ogawa, K. Takeda, and F. Itakura. 1998. Balancing Acoustic and Linguistic Probabilities. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

F. Richardson, M. Ostendorf, and J.R. Rohlicek. 1995. Lattice-based search strategies for large vocabulary speech recognition. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Roni Rosenfeld, Stanley F. Chen, and Xiaojin Zhu. 2001. Whole-Sentence Exponential Language Models: a Vehicle for Linguistic-Statistical Integration. *Computer Speech and Language*, 15(1).

Roni Rosenfeld. 1997. A Whole Sentence Maximum Entropy Language Model. In *Proc. of IEEE workshop on Automatic Speech Recognition and Understanding (ASRU)*, Santa Barbara, California, December.

D.E. Rumelhart, G. E. Hinton, and R.J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323:533–536.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech and Language*, 21(3):492–518.

C. E. Shannon. 1948. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27:379–423, 623–656.

H. Soltau, G. Saon, and B. Kingsbury. 2010. The IBM Attila speech recognition toolkit. In *Proc. of IEEE Workshop on Spoken Language Technology (SLT)*.

Wen Wang and Mary Harper. 2002. The SuperARV language model: investigating the effectiveness of tightly integrating multiple knowledge sources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.