

RESCORING EFFECTIVENESS OF LANGUAGE MODELS USING DIFFERENT LEVELS OF KNOWLEDGE AND THEIR INTEGRATION

Wen Wang, Yang Liu, Mary P. Harper

School of Electrical and Computer Engineering, Purdue University
West Lafayette, IN 47907-1285
{wang28, yangl, harper}@ecn.purdue.edu

ABSTRACT

In this paper, we compare the efficacy of a variety of language models (LMs) for rescoring word graphs and N-best lists generated by a large vocabulary continuous speech recognizer. These LMs differ based on the level of knowledge used (word, lexical features, syntax) and the type of integration of that knowledge (tight or loose). The trigram LM incorporates word level information; our Part-of-Speech (POS) LM uses word and lexical class information in a tightly coupled way; our new SuperARV LM tightly integrates word, a richer set of lexical features than POS, and syntactic dependency information; and the Parser LM integrates some limited word information, POS, and syntactic information. We also investigate LMs created using a linear interpolation of LM pairs. When comparing each LM on the task of rescoring word graphs or N-best lists for the Wall Street Journal (WSJ) 5k- and 20k- vocabulary test sets, the SuperARV LM always achieves the greatest reduction in word error rate (WER) and the greatest increase in sentence accuracy (SAC). On the 5k test sets, the SuperARV LM obtains more than a 10% relative reduction in WER compared to the trigram LM, and on the 20k test set more than 2%. Additionally, the SuperARV LM performs comparably to or better than the interpolated LMs. Hence, we conclude that the tight coupling of knowledge from all three levels is an effective method of constructing high quality LMs.

1. INTRODUCTION

In speech recognition systems, it is generally infeasible to use a one-pass search due to the huge computational effort imposed by a complex LM. Hence, a multi-pass search is often employed, and this is particularly true in the case of a large vocabulary continuous speech recognition system. In such systems, word graphs or N-best lists are chosen as the interface between the acoustic recognizer and a post-processing module using a more complex LM.

Investigations [10, 12] demonstrate that using more advanced LMs can help to eliminate acoustic ambiguity and obtain a better recognition result. In this paper, we investigate the impact of applying a variety of LMs to rescore word graphs and N-best lists. We also investigate the effectiveness of interpolating pairs of LMs to improve the recognition accuracy. The structure of this paper is as follows. In Section 2, we introduce the broad range of LMs that we will use to rescore word graphs and N-best lists. Section 3

describes the experimental setup and results. Conclusions appear in Section 4.

2. LANGUAGE MODEL DESCRIPTION

2.1. Word n-gram LMs

The word n-gram LM for a sequence of N words w_1^N is expressed as follows:

$$Pr(w_1^N) \approx \prod_{i=1}^N Pr(w_i | w_{i-n+1}^{i-1}) \quad (1)$$

We use a bigram LM ($n=2$) and a trigram LM ($n=3$) using the Katz's back-off model combined with Good-Turing discounting in our experiment¹. Other smoothing strategies were evaluated and found to be less effective for this task.

2.2. Our POS LM

Our POS LM is a generalization of word LMs, which allows intrinsic generalization to unseen word sequences and deals with data sparseness by grouping words into classes based on POS and then using these classes in computing n-gram probabilities. The LM uses word and lexical class information in a tightly coupled way and estimates the joint probability of words w_1^N and their tags t_1^N as follows:

$$\begin{aligned} Pr(w_1^N t_1^N) &= \prod_{i=1}^N Pr(w_i t_i | w_{i-1}^{i-1} t_{i-1}^{i-1}) \quad (2) \\ &= \prod_{i=1}^N Pr(t_i | w_{i-1}^{i-1} t_{i-1}^{i-1}) \cdot Pr(w_i | w_{i-1}^{i-1} t_i^i) \\ &\approx \prod_{i=1}^N Pr(t_i | w_{i-2}^{i-1} t_{i-2}^{i-1}) \cdot Pr(w_i | w_{i-2}^{i-1} t_{i-2}^i) \quad (3) \end{aligned}$$

Equation (2) is adopted from the work of Heeman [7], which enables a LM that achieves a significant perplexity reduction compared to a trigram by estimating the joint probability of a sequence of words and their tags. An alternative equation for a POS LM was used by [8]:

$$Pr(w_1^N) = \sum_{t_1, t_2, \dots, t_N} \prod_{i=1}^N Pr(t_i | t_1^{i-1}) Pr(w_i | t_i) \quad (4)$$

A POS LM using Equation (4) is less effective at predicting word candidates than an n-gram word LM [8] because in

¹The bigram and trigram LMs we used in the experiments are provided with WSJ0 and WSJ1 from Linguistic Data Consortium.

this case the use of POS tags depletes the lexical information necessary for predicting the next word.

Inspired by the superior performance of a full second-order HMM POS tagger developed by Thede et al. [13], we employ Equation (3) which uses richer history information in probability estimation than a traditional trigram. We compared a variety of smoothing algorithms [14] and chose the modified Kneser-Ney smoothing algorithm [4] using cutoffs for rare n-grams and a heldout data set to optimize parameters. The LM hypothesizes categories for out-of-vocabulary words using the leave-one-out technique [9]. The combination of rich history and an effective smoothing strategy produces a high quality POS LM that improves upon Heeman’s POS LM. To avoid penalizing the LM for incorrect tag assignments, the word perplexity evaluation of our POS LM is based on the formula in [7].

2.3. Our SuperARV LM

Investigations of LMs employing word, syntactic structural information, and lexical features have been made [2, 3, 11]. However, none of these LMs integrates all three knowledge sources in a tightly coupled way. In this paper, we develop a new dependency-based almost-parsing LM, *SuperARV LM*, which uses enriched tags (*SuperARVs*) combining syntactic dependencies and lexical features of words within a uniform representation.

The SuperARV LM is a highly lexicalized probabilistic LM based on the Constraint Dependency Grammar (CDG) [5]. CDG represents a parse as assignments of dependency relations to functional variables (denoted *roles*) associated with each word in a sentence. The set of dependency relations from a corpus can be used to develop a grammar [6]. A SuperARV is an abstraction of the joint assignment of dependencies for a word, which provides a mechanism for lexicalizing CDG parse information. Words typically have more than one SuperARV to indicate different types of word usage. SuperARVs encode lexical information as well as syntactic and semantic constraints in a uniform representation that is much more fine-grained than POS. A SuperARV is defined as the four-tuple for a word, $\langle C, F, (R, L, UC)^+, DC \rangle$, where C is the lexical category of the word, $F = \{Fname_1 = Fvalue_1, \dots, Fname_f = Fvalue_f\}$ is a feature vector (where $Fname_i$ is the name of a feature and $Fvalue_i$ is its corresponding value), $(R, L, UC)^+$ is a list of one or more three-tuples, each representing an abstraction of a role value assignment, where R is a role variable, L is a functionality label, and UC represents the relative positional relation of a word and its dependent. The following features are used in our SuperARV LM, namely, *agr*, *case*, *vtype* (e.g., progressive), *mood*, *gap*, *inverted*, *voice*, *behavior* (e.g., mass, count), *type* (e.g., interrogative, relative) [14]. DC represents the relative ordering of the positions of a word and all of its modifyees. Besides the governor role defined in the traditional dependency grammar, we add need roles to ensure the grammatical requirements of a word are met. Including need roles also provides a mechanism for using non-headword dependencies, which Bod [1] has shown contributes to improved parsing accuracy. Figure 1 gives an example of a SuperARV. Notice that this schema provides an explicit way to organize information concerning one consistent set of dependency links

for a word.

Category: Verb, Features: {VerbType=past, Voice=active, Inverted=yes, Gapp=gap, Mood=wh, Agr=all}	
Role=G, Label=VP, (PX > MX)	<i>(Governed by a word on its left)</i>
Role=Need1, Label=S, (PX < MX)	<i>(Need a modifyee on its right)</i>
Role=Need2, Label=S, (PX < MX)	<i>(Need a modifyee on its right)</i>
Role=Need3, Label=S, (PX = MX)	<i>(No modifyee)</i>
Dependent Positional Constraints: MX[G] < PX = MX[Need3] < MX[Need1] < MX[Need2]	

Figure 1: The SuperARV of the word *did* in the sentence *what did you learn*. Note: G represents the governor role; the $Need1$, $Need2$, and $Need3$ roles are used to ensure the constraints that the requirements of the word are met. PX and MX represent the position of a word and its modifyee, respectively.

SuperARVs can be accumulated from a corpus annotated with CDG relations and stored directly in the CDG lexicon. A SuperARV can be selected from the lexicon and used to selectively generate role values that meet their constraints. Since there are no benchmark corpora annotated with CDG information, we developed a methodology [14] adapted from [3] to transform constituent bracketing into CDG annotations. In addition to generating dependency structures by headword percolation [3], our transformer also utilizes a rule-based method to generate lexical features and need role values for words [14].

The SuperARV LM is implemented similarly to the POS LM using Equation (3). A smoothing strategy and perplexity evaluation method similar to those used for the POS LM is employed.

The work of the SuperARV LM is most closely related to the almost-parsing based LM developed by Srinivas [11] that is based on a notion of a *supertag*, the elementary structure of the Lexicalized Tree-Adjoining Grammar. This structure localizes dependencies, including long distance dependencies, by requiring that all and only the dependent elements be present within the same structure. On the WSJ corpus, he obtained a perplexity reduction compared to a POS n-gram LM [9]. The supertag n-gram LM [11] uses a deficient conditional probability model based on Equation (4) and a back-off smoothing algorithm. By comparison, our SuperARV LM incorporates syntactic dependency information with more lexical feature information, uses Equation (3), and applies a more effective smoothing algorithm for this model.

2.4. Parser LM

Chelba et al. [3] built a Parser LM which develops hidden hierarchical structures incrementally to model long distance dependencies. The LM, when interpolated with a trigram LM, reduces perplexity and WER compared to a trigram LM. Charniak [2] developed a Parser LM based upon an “immediate-head” parser. On the WSJ Penn Treebank corpus, his LM reduces the perplexity by 22% compared to

the trigram model baseline. In this paper, Charniak’s LM is used for rescore N-best lists. Because this Parser LM parses a sentence in a top-down manner, it can be used to rescore N-best lists but not lattices.

3. EXPERIMENTS

3.1. Experimental Setup

In the experiment, we use the WSJ 1992 (denoted 92-5k) and 1993 5K (93-5k) closed vocabulary and 1993 20K (93-20k) open vocabulary tasks. This task was chosen because it is a large vocabulary continuous speech recognition task for which we have access to the needed annotations for training a wide variety of LMs. The provided word n-gram LMs were trained on the WSJ 1987-1989 text files. The POS tagging and constituent bracketing (translated to CDG annotations) corresponding to this training set were extracted from the Penn and BLLIP WSJ Treebanks to train the POS and SuperARV LMs, respectively. The Parser LM was trained [2] on sections 00-20 of the WSJ Penn Treebank with sections 21-22 used for development.

The acoustic recognizer, built using HTK [15], is an HMM triphone system with an embedded bigram LM. The gender independent acoustic model is trained based on the combination of WSJ0’s and WSJ1’s short-term speaker independent training data. Cross-word triphone models are tied at the state level using decision trees. The acoustic feature vector contains 12 Mel frequency cepstral coefficients and log energy, plus the first and second differentials of these coefficients. The cepstral features are normalized for each sentence by subtracting the cepstral mean of the sentence. The decoding is a token-passing Viterbi search process. For each test sentence, we generate a word graph, as well as 100-best lists based on the interpolation of the acoustic score and bigram LM score, with the grammar scale chosen to minimize the WER for the corresponding development set.

3.2. Rescoring Results and Discussion

One advantage of the N-best list representation as an interface is that it meets the input requirements of most subsequent LMs. Any LM that applies to a written sentence can also be applied to N-best lists. We will use the bigram and trigram, POS, SuperARV, and the Parser LM described in Section 2 to rescore the N-best lists. All LMs except the Parser LM will also be used to rescore word graphs using the Viterbi algorithm. During rescoring, the best hypothesis is picked based on the combination of the acoustic score and the new LM score weighted by a grammar scale that is chosen to minimize WER based on the development set.

The test set perplexity of each LM is presented in Table 1 for comparison with the recognition results. The lowest perplexity for each test set is shown in bold face. The SuperARV LM achieves a relative perplexity reduction on the 92-5k, 93-5k, and 93-20k test sets of 38.76%, 47.30%, and 19.32% compared to the trigram LM, and 36.82%, 12.03%, and 8.82% compared to the POS LM, respectively.

Table 2 shows the WER and SAC after rescoring using each LM alone, with the lowest WER and highest SAC for each test set presented in bold face. As can be seen from Table 2, when using each LM alone, the SuperARV LM performs the best with POS the second best. When

rescoring N-best lists on the 92-5k, 93-5k, and 93-20k test sets, the SuperARV LM yields a relative WER reduction of 15.98%, 9.62%, and 2.18% compared to the trigram, and 1.77%, 2.74%, and 1.64% compared to the POS LM, respectively. On the other hand, the SuperARV LM achieves an **absolute** increase on SAC of more than 5% on the 5k test sets and around 2% on the 20k test set, compared to the trigram, and 0.6%-2.33% on the test sets compared to the POS LM. Similar results were observed when rescoring lattices.

LM	92-5k	93-5k	93-20k
2gram	85.51	83.67	164.48
3gram	45.61	50.51	109.22
POS	44.21	30.26	96.64
SARV	27.93	26.62	88.12
Parser	214.53	196.88	250.55

Table 1: Perplexity results for each language model.

LM	92-5k	93-5k	93-20k
	WER(SAC)	WER(SAC)	WER(SAC)
rescoring 100-best lists			
2gram	6.18(48.79)	9.02(33.02)	16.80(23.47)
3gram	4.63(59.39)	7.07(43.26)	14.71(30.99)
POS	3.96(64.55)	6.57(46.51)	14.63(31.92)
SARV	3.89(65.15)	6.39(48.84)	14.39(32.86)
Parser	8.18(31.21)	9.87(32.56)	20.14(15.02)
rescoring word graphs			
2gram	6.18(48.79)	9.02(33.02)	16.80(23.47)
3gram	4.43(61.52)	6.91(43.26)	14.74(30.52)
POS	3.92(64.85)	6.55(47.91)	14.54(32.39)
SARV	3.87(65.76)	6.37(49.77)	14.35(33.80)

Table 2: Rescoring results using each LM alone.

LM	92-5k	93-5k	93-20k
	WER(SAC)	WER(SAC)	WER(SAC)
2gram+Parser	5.31(53.33) ↑	8.34(39.53) ↑	16.77(24.88) ↑
3gram+Parser	3.94(65.76) ↑	6.68(46.51) ↑	14.86(31.46) ↓
POS+Parser	3.92(64.85) ↑	6.29 (48.37) ↑	14.63(31.92)
SARV+Parser	3.89 (65.15)	6.39(48.84)	14.39(32.86)
2gram+3gram	4.39(61.52) ↑	6.96(44.19) ↑	14.80(30.05) ↓
3gram+POS	3.96(64.55)	6.57(46.51)	14.74(31.46) ↓
POS+SARV	3.90(64.85) ↓	6.37(48.84) ↑	14.63(31.92) ↓
3gram+SARV	3.89 (65.15)	6.39(48.84)	14.34(33.33) ↑

Table 3: Rescoring 100-best lists using interpolated LMs.

In order to investigate the effectiveness of loosely integrating LMs that use different levels of knowledge, we linearly interpolated pairs of LMs to rescore 100-best lists. Recognition results are shown in Table 3 (with bold face used as in Table 2). The interpolation weights were chosen to minimize the WER on the corresponding development set. Note that when the combination of two LMs is worse than the result of the stronger LM in this pair by itself on the development set, the weight of the weaker LM was set to 0; hence, for some combinations we obtain the same result on the test set as that of the stronger LM in Table 2, and there are no arrows shown in these cells in Table 3. The

uparrows and downarrows in Table 3 depict whether the WER and SAC of the interpolated LM is better or worse than that of the stronger LM²; an up-down arrow marks a mixed result on WER and SAC.

As can be seen from Table 3, sometimes the interpolation of a LM pair yields a more accurate result than the stronger LM by itself and sometimes not. However, the interpolation of word n-gram and Parser LMs produces a consistent improvement on WER and SAC compared to the stronger LM³. This suggests that the Parser LM provides additional information, namely syntactic structure, that is not contained in a traditional n-gram LM, which captures information about adjacent words. A similar result has also been found in [3], where a traditional trigram and a structured LM were combined, and a better result was obtained than using either LM alone. On the other hand, SuperARV overlaps significantly with the knowledge represented in the Parser LM; hence, it does not gain from interpolation with that LM. The POS LM falls between the n-gram LMs and SuperARV LM in the degree of overlap with the Parser LM. POS+SuperARV and 3gram+SuperARV occasionally obtain a slight improvement compared to the stronger LM, but typically not, and 3gram+POS leads to no improvement or a degradation. This is likely due to the fact that the two LMs for interpolation are not independent; much of the information from the weaker LM is already included in the stronger one.

The investigation of LM combinations suggests that some metrics could be used to predict whether LMs can be effectively combined. Intuitively, we expect to combine two independent LMs. One way to measure the degree of independence of two LMs is to determine the correlation of the probabilities assigned to the test sentences. We found that this measure, to some extent, is predictive of whether two LMs can be combined. For example, the bigram and trigram LMs have lower correlations with the Parser LM than the POS and SuperARV LMs. Also, we observed the correlation between the n-gram and Parser LMs on the 5k test sets was lower than on 20k test set, which is consistent with the fact that the combination of n-gram and Parser on the 5k sets yields a greater, more consistent improvement than on the 20k set. Other measurements were investigated that were less predictive (e.g., the Kullback-Leibler divergence, the N-best re-ranking) than the correlations. We will investigate other measurements in future work.

4. CONCLUSIONS

We have compared the quality of five LMs, as well as linear interpolations of these models. The SuperARV LM is the most effective model for rescoring both N-best lists and word graphs generated for the WSJ evaluation sets. The SuperARV LM or an interpolation with the SuperARV LM also outperforms all of the interpolated LMs on WER and

²The linear interpolation of two LMs can be worse than the result provided by one LM in the pair alone since the interpolation weights were selected based on the development set and might not be the best for the test set.

³The 3gram+Parser has a degraded WER on 93-20k because the 20k development set is fairly different from the test set. Improvement is obtained on both WER and SAC if weights are set based on the test set.

SAC, except for the SAC on the WSJ 92-5k set and the WER on the WSJ 93-5k set. This suggests that the tight coupling of knowledge involving word, lexical features, and syntactic structure helps us to build a more powerful LM. Investigations on the interpolation of LMs suggest that two LMs that incorporate much of the same type of knowledge are poor candidates for a consistently effective interpolated LM.

5. ACKNOWLEDGMENTS

This research was supported by Intel, Purdue Research Foundation, and National Science Foundation under Grant No. IRI 97-04358, CDA 96-17388, and BCS-9980054.

6. REFERENCES

- [1] R. Bod. What is the minimal set of fragments that achieves maximal parse accuracy? In *Proceedings of the ACL'2001*. Association for Computational Linguistics, 2001.
- [2] E. Charniak. Immediate-head parsing for language models. In *Proceedings of the ACL'2001*, 2001.
- [3] C. Chelba. *Exploiting Syntactic Structure for Natural Language Modeling*. PhD thesis, Johns Hopkins University, 2000.
- [4] S. F. Chen and J. T. Goodman. An empirical study of smoothing techniques for language modeling. Technical report, Harvard University, Computer Science Group, 1998.
- [5] M. P. Harper and R. A. Helzerman. Extensions to constraint dependency parsing for spoken language processing. *Computer Speech and Language*, 9:187–234, 1995.
- [6] M. P. Harper, C. M. White, W. Wang, M. T. Johnson, and R. A. Helzerman. Effectiveness of corpus-induced dependency grammars for post-processing speech. In *Proceedings of the NAACL'2000*, 2000.
- [7] P. A. Heeman. POS tagging versus classes in language modeling. In *Proceedings of the 6th Workshop on Very Large Corpora, Montreal*, 1998.
- [8] F. Jelinek. Self-organized language modeling for speech recognition. *Readings in Speech Recognition*, 1990.
- [9] T. R. Niesler and P. C. Woodland. A variable-length category-based n-gram language model. In *Proceedings of ICASSP 1996*, pages 164–167, 1996.
- [10] M. Rayner, D. Carter, V. Digalakis, and P. Price. Combining knowledge sources to reorder N-best speech hypothesis lists. In *ARPA Human Language Technology Workshop*, pages 212–217, 1994.
- [11] B. Srinivas. ‘Almost parsing’ technique for language modeling. In *Proceedings of the ICSLP'96*, volume 2, pages 1173–1176, 1996.
- [12] A. Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, pages 165–201, 1995.
- [13] S. M. Thede and M. P. Harper. A second-order hidden Markov model for part-of-speech tagging. In *Proceedings of the ACL'1999*, pages 175–182. Association for Computational Linguistics, June 20–26, 1999.
- [14] W. Wang. Investigating probabilistic constraint dependency grammars in language modeling. Technical report, Purdue University, School of Electrical Engineering, 2001.
- [15] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. HTK. World Wide Web, <http://www.htk.eng.cam.ac.uk>, 2000.