To Appear In Proceedings of ICSLP-98, Sydney

TOWARDS BETTER INTEGRATION OF SEMANTIC PREDICTORS IN STATISTICAL LANGUAGE MODELING

Noah Coccaro and Daniel Jurafsky†**

Department of Computer Science* Department of Linguistics†
University of Colorado at Boulder
{noah,jurafsky}@colorado.edu

ABSTRACT

We introduce a number of techniques designed to help integrate semantic knowledge with N-gram language models for automatic speech recognition. Our techniques allow us to integrate Latent Semantic Analysis (LSA), a word-similarity algorithm based on word co-occurrence information, with N-gram models. While LSA is good at predicting content words which are coherent with the rest of a text, it is a bad predictor of frequent words, has a low dynamic range, and is inaccurate when combined linearly with N-grams. We show that modifying the dynamic range, applying a per-word confidence metric, and using geometric rather than linear combinations with N-grams produces a more robust language model which has a lower perplexity on a Wall Street Journal testset than a baseline N-gram model.

1. INTRODUCTION

There has been a lot of recent work on augmenting n-gram language models with other information sources such as longer distance syntactic, and semantic constraints (e.g. [8], [6]). In previous work, we [1] and others [2], [5] have suggested the use of Latent Semantic Analysis (LSA) [3] as a model of semantic knowledge to be applied to ASR. LSA is a model of word semantic similarity based on word co-occurrence tendencies, and has been successful in IR and NLP applications, such as spelling correction [7]. LSA is good at predicting the presence of words in the domain of the text, but not good at predicting their exact location. The N-gram model complements the LSA model by filling in the missing information – where exactly the content words should go.

We have, however, also discovered that LSA is often a bad predictor of the next word for several reasons. First, LSA is only good at predicting words that are closely tied to a semantic domain. Second, LSA produces cosines which tend to have a narrow dynamic range – it is not trivial to map these cosine distances into a probability estimation. Furthermore, optimal combination of LSA and N-gram probabilities is a difficult and unsolved problem.

Our solution to these problems involves three techniques to combine the 2 estimators: a confidence metric for LSA, a modification of the dynamic range of the LSA probabilities, and a more conservative evidence combination function. Our algorithm achieves a significant reduction in perplexity on the Wall Street Journal corpus.

2. LSA AS A LANGUAGE MODEL

LSA is a vector based model model of semantics based on word co-occurrences. Words that tend to occur together, or with similar words are considered to be semantically similar. The LSA algorithm is trained on a corpus of documents. Documents here are any semantically cohesive set of words, such as paragraphs, articles from newspapers, newsgroup articles, etc. Since structure within the documents is not maintained, this is referred to as a bag-of-words model. To build the LSA model for the experiments in this paper, we used 81,553 articles from the Wall Street Journal, years 1987–1989, containing of a total 35,126,006 word tokens. We used the 20K vocabulary set from the WSJ0 distribution as our vocabulary set. A term by document matrix is then created, with rows corresponding to words in the vocabulary, and columns to documents. Each entry in the matrix is a weighted frequency of the corresponding term in the corresponding document. This weighting is chosen to reduce influence of frequently occurring terms, such as the function words and is described in more detail in Section 4.

The next step is to reduce this very large sparse matrix into a compressed matrix based on singular value decomposition, SVD.

$$M = T \times S \times D'$$

The original $t \times d$ matrix M is decomposed into a reduced rank $t \times k$ term matrix T, a diagonal matrix of singular values, S, and a $d \times k$ document matrix D.

Decreasing k, the number of dimensions retained, reduces the accuracy with which M can be recreated from its component matrixes, but importantly, it reduces the noise from the original matrix. We chose 300 as a value for k in our experiments, as empirical results in IR find that to be a good value. In this task, we are only interested in the term matrix, T. Each row of T is a vector representation of the semantics of a particular word, in a k dimensional space. We can now compare the semantic distance between any two words by looking at the cosine of the angle (normalized dot product) of the two corresponding rows (vectors) in the matrix T. Figure 1 shows the closest and farthest words to the word *fishing*, according to LSA. Furthermore, we can compare a single word to a set of words, such as a newspaper article, or some arbitrary context of a speech recognition system. The vectors for each word in the set are combined to form a vector that represents the centroid of that set, a point in the k dimensional semantic

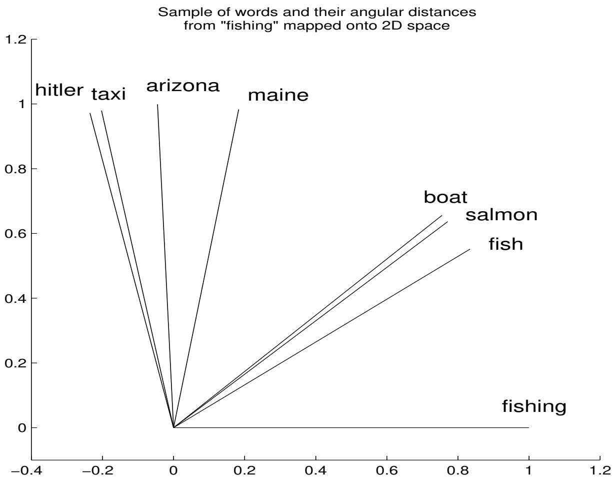

Figure 1: Plotted are angular distances of several words from the word "fishing", mapped into 2D. Some of the closest words ("fishing", "fish") and the farthest words, ("hitler", "taxi") are shown. The states are roughly ordered in terms of their importance to the fishing industry.

space. Then, the distance between the vector for any one word and the centroid can be computed, yielding the semantic distance between that word and the document. We use this comparison to get a measure of semantic distance between a hypothesis of what word comes next, and every word previous in the article currently being recognized.

3. DERIVING LSA PROBABILITIES

In the previous section we showed how LSA has predictive powers for subsequent words in a semantically cohesive text, such as a newspaper article. Now we show how to convert this into a probability estimate with relatively little computation. The simplest method is to use the distance directly, and normalize such that the least likely word has a probability of 0, and all probabilities sum to 1 (see Equation 4.) However, this approach is too simplistic and performs poorly, because the probabilities generated have a very limited dynamic range — much smaller than that of N-grams. The probabilities derived hover around 1/VocabSize, not differing enough have much impact. Our next technique overcomes this problem.

We increase the dynamic range of the LSA probabilities by raising the cosines to some power first, then normalizing. We selected an appropriate power, γ, by empirical testing, and found 7 to work best on a held out dev-test set.

The context for our implementation is defined as all the words, C in the WSJ article before the current word. This allows us to draw on the natural semantic coherence of a newspaper article while still allowing on-line predictability of the next word. First, the centroid, $\vec{C}$ of the vectors corresponding to the words C in the context is computed

$$\vec{C} = \sum_{t=1}^{i} \vec{w}_t \tag{1}$$

The cosine is computed between the LSA vector for the ith word in the document, w_i and the centroid of the vectors of all the words in the context, C.

$$\cos(\vec{w}_i, \vec{C}) = \frac{(\vec{w}_i \cdot \vec{C})}{\|\vec{w}_i\|\|\vec{C}\|} \tag{2}$$

We find the smallest cosine between the context and any word W_j which ranges over the N vocabulary items. This is computed for normalization purposes.

$$\text{MinCos}(C) = \min_{j=1}^{N} \cos(\vec{W}_j, \vec{C}) \tag{3}$$

Then a first estimate of the LSA probability is computed by taking the cosine between this word and the context, subtracting out the MinCos to calibrate the lowest value at zero, and then normalizing by the sum of the cosines of all words in the vocabulary with the context

$$\hat{P^L}(w_i|C) = \frac{\cos(\vec{w}_i, \vec{C}) - \text{MinCos}(C)}{\sum_{j=1}^{N} \cos(\vec{W}_j, \vec{C}) - \text{MinCos}(C)} \tag{4}$$

Finally, the probability is reestimated by raising it to a power, γ and renormalizing.

$$P^L(w_i|C) = \frac{\hat{P^L}(w_i|C)^{\gamma}}{\sum_{j=1}^{N} \hat{P}(w_j|C)^{\gamma}} \tag{5}$$

Introducing γ greatly improved results. The baseline perplexity computed solely by the bigram model on a held out development test set was 147.8. Adding in the LSA model without the power factor, (that is, $\gamma = 1$) we showed a small decrease in performance, to 148.5. However, using a γ of 7.0 decreased the perplexity to 130.4, an 11% improvement. (See Table 2 for more details of the experiment.)

4. LSA CONFIDENCE METRIC

One problem with LSA is that compared to the N-gram it is a poor predictor of function words ('the', 'of'), and other common words with uniform distribution over contexts. But LSA does well at predicting the presence of content words which are specific to a context, even if they have not occurred yet in the document (e.g. 'soft' (as in 'soft drink') in the context of a WSJ story about Coca-Cola). We introduce a confidence metric associated with each word that helps determine to what degree the LSA model is effective at predicting that word. Our confidence metric is a 'global term weighting' found to be useful in IR applications: the entropy of the frequency of a word over all documents in the training corpus [4]. Thus, the LSA confidence for term i is calculated by

$$\text{LSA Confidence}_i = 1 + \frac{\sum_{j=1}^{n\text{docs}} P(j|i)\log(P(j|i))}{\log(n\text{docs})}$$

where $n$docs is the number of documents in the corpus, and $P(j|i)$ is the likelihood of document j given that term i occurs in it j:

$$P(j|i) = \frac{\text{Count of term } i \text{ in document } j}{\text{Count of term } i \text{ in whole corpus}}$$

Word	N-grm	LSA	LSA Conf.	Word	N-grm	LSA	LSA Conf
coca	-	-	-	investigation	272	35947	.32
cola	-	-	-	into	13	38902	.14
enterprises	9	210	.39	alleged	588	19267	.33
incorporated	3	13889	.10	antitrust	402	11220	.41
said	7	43672	.05	violations	31	12065	.38
its	36	23377	.06	in	20	14420	.47
atlanta	7255	**1154**	.37	the	4	12192	.04
coca	6392	**60**	.50	soft	8406	**93**	.39
cola	1	70	.49	drink	3	111	.45
bottling	18	69	.57	industry	33	26139	.19
company	5	7825	.07	by	239	10678	.07
unit	93	19348	.15	a	15	12813	.04
is	41	28504	.07	federal	210	13387	.17
a	11	19433	.04	grand	67	15935	.37
target	2228	18611	.31	jury	2	10745	.38
of	4	12521	.04	in	7	10098	.05
an	147	23809	.07	atlanta	1110	**910**	.37

Table 1: Sample perplexities assigned by LSA and bigram models and LSA confidences, for a sentence in WSJ.Bold-faced values indicate places where LSA was a better predictor of the correct word than the N-gram model.

The LSA confidence ranges from a low of 0.04 for *of* to 0.91 for *kevlar*. Words that are very promiscuous, occurring in many documents without regard to their content (high entropy, uniform distribution) will get a very low LSA confidence value, while words that are less promiscuous, usually occurring with the same family of words, will get a higher confidence value.

In Table 1, we see that LSA often incorrectly assigns a low probability (high perplexity) to words, often much lower than an N-gram model does. However, where LSA outperforms the bigram model, we see that the term being predicted has a relatively high LSA confidence.

In the next section we will show how the LSA confidence measure can be used to discount the LSA probability when combining it with the N-gram predictor. For these words, such as *of*, the context that LSA uses to base its judgment on is not nearly as discriminative as the previous word, which the N-gram model uses to predict these words. As we will see, the LSA confidence model gave a lower perplexity on our WSJ development set than the baseline bigram model. Table 2 shows that the perplexity computed with the variable confidence was 130.4, an improvement over the bigram-only baseline of 147.8.

We suspected that part of the improvement from the LSA confidence model was due to a general discounting of the LSA probability rather than the per-word effect. We thus ran another experiment to test these two factors. We computed the average LSA confidence for all terms in the development test set, and used that constant factor (0.09) as a weight. The γ factor was held the same, 7.0. Perplexity in this test was 139.7, demonstrating the variable LSA confidence did improve results.

5. COMBINING LSA & N-GRAMS

Since the N-gram model is still a good predictor of words, we want to guarantee that it contributes at least half the probability mass to predicting the next word. Therefore, we divide the LSA confidence in half, such that it ranges from 0 to 0.5. Thus, for words that LSA is very confident about predicting, like *kevlar*, with an LSA confidence of 0.91, the LSA and N-gram models will be about equally considered.

$$\lambda_i = \frac{\text{LSA Confidence}_i}{2}$$

We now address the problem of combining our modified LSA estimator with the N-gram probability. We found simple linear combination to be inadequate (Equation 6), partly because the LSA estimator often predicts words that are syntactically disallowed. We need a non-linear combination function that gives a much higher probability when the two models agree — that is, when the predicted word is both syntactically and semantically likely — and gives a low probability if either estimator believes a word unlikely. We chose the geometric mean as our non-linear combination function (Equation 7). When the LSA confidence is high, this function forces the N-gram and LSA model to agree a word is likely in order to get a high resulting probability. When the LSA confidence is low, the need for agreement is reduced.

Consider the case of the bigram *"coca atlanta"* which is a very unlikely bigram. We do not want LSA to increase the likelihood of this bigram just because the term *atlanta* is semantically related to the rest of the context. The bigram model assigns a very low probability to this word sequence. The geometric mean in such a case would yield a very low probability, but the arithmetic mean would incorrectly give a significant probability to this bigram due to the LSA influence. Equation 6 shows this sub-optimal linear equation we tested:

$$P(w_i|w_1, w_2 ... w_{i-1}) = \frac{(P^L(w_i|w_1,w_2,...w_{i-2})*\lambda_i)+(P^B(w_i|w_{i-1})*(1-\lambda_i))}{\sum_{j=1}^{N}(P^L(W_j|w_1,w_2,...w_{i-2})*\lambda_j)+(P^B(W_j|w_{i-1})*(1-\lambda_j))} \quad (6)$$

The final equation that we used, which includes the LSA dynamic range adjustment, the LSA confidence, and the geometric combination of the LSA and N-gram probabilities is shown below:

$$P(w_i|w_1, w_2 ... w_{i-1}) = \frac{P^L(w_i|w_1,w_2,...w_{i-2})^{\lambda_i} P^B(w_i|w_{i-1})^{1-\lambda_i}}{\sum_{j=1}^{N} P^L(W_j|w_1,w_2,...w_{i-2})^{\lambda_j} P^B(W_j|w_{i-1})^{1-\lambda_j}} \quad (7)$$

As in earlier equations, w_i is the ith word to be recognized, W_j is the jth term in the vocabulary, N is the number of words in the vocabulary, P^L is the word probability according to LSA, and P^B is the word probability according to the bigram model.

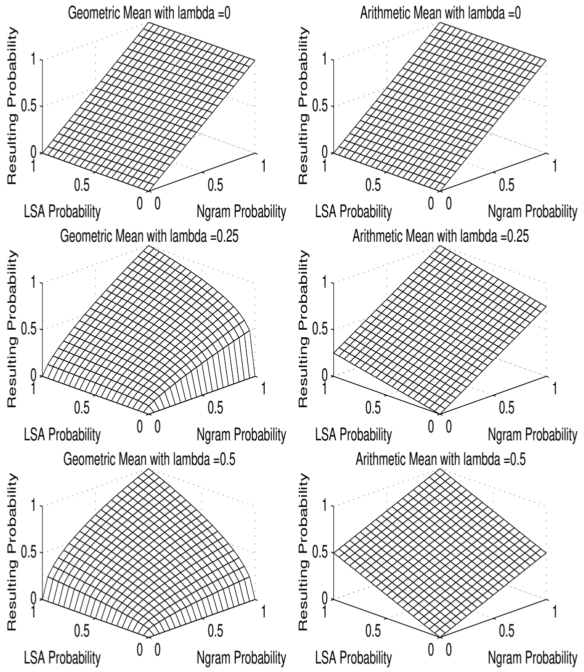

Figure 2: Above are shown the probability spaces for combining LSA and N-gram probabilities, with three different values of the LSA confidence metric

The graphs in Figure 2 demonstrate the resulting probability space from using geometric and arithmetic means. Consider the point on these graphs where the LSA probability is high, and the N-gram probability is nearly zero. Note that the geometric mean assigns this case a low probability, while the arithmetic mean gives it a relatively high probability. On the left is geometric mean, on the right, arithmetic mean, with increasing LSA confidence as one goes down.

Testing with the arithmetic mean showed significantly worse results than when testing with the geometric mean, yielding perplexity of 143.1 vs. 130.4 on the development test.

We now tested our final model on a separately held out test set. We again trained on 35 million words from WSJ 1987–1989, but now tested on 234,000 words from WSJ years 1995 and 1996 found in the NAB news corpus. We showed a 12% decrease in test set perplexity when using the LSA model, going from a perplexity of 191 to a perplexity of 168. Table 2 shows this result and intermediate dev test results.

6. CONCLUSIONS AND FUTURE WORK

We have shown how three methods which require relatively little computation can significantly increase the performance of a language model that incorporates semantic information. Our semantic confidence measure improved performance by accurately predicting when a semantic model would be beneficial. Increasing the influence of a metric with low dynamic range proved to be important. Finally, a geometric combination of evidence favors situations where the two orthogonal models agree.

Dev Test Method	PPL
Bigram	147.8
Bigram + LSA + Confidence + Geometric Mean	148.5
Bigram + LSA + Confidence + γ + Arithmetic Mean	142.1
Bigram + LSA + γ + Geometric Mean	139.7
Bigram + LSA + Confidence + γ + Geometric Mean	130.4
Held out Test Set	
Method	PPL
Bigram	191
Bigram + LSA + Confidence + γ + Geometric Mean	168

Table 2: Resulting Perplexities on development test set, and on a separately held out test set.

One further modification of the system would be to add an N-gram confidence metric. This would help in cases where the LSA prediction is wrong despite a high LSA confidence, e.g. for the word *cola* in Table 1. We are currently investigating a factor based on the entropy of the N-gram distributions for a given prefix.

7. ACKNOWLEDGMENTS

Thanks to Tom Landauer for inspiring conversations that led to these ideas, Apple Computer for funding an internship that started the implementation of this work, Andreas Stolcke for comments and providing the SRI LM toolkit we used, Jim Martin and Wayne Ward for providng useful feedback, and the NSF via NSF IRI-9704046 and NSF IIS-9733067 to the second author.

8. REFERENCES

1. J. Bellegarda, J.W. Butzburger, Y. Chow, N. Coccaro, and D. Naik. A novel word clustering algorithm based on latent semantic analysis. In *Proceedings of ICASSP-96*, 1996.
2. Jerome Bellegarda. A latent semantic analysis framework for large-span language modeling. In *Eurospeech*, 1997.
3. S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.
4. Susan. T. Dumais. Improving the retrieval of information from external sources. *Behavior Resarch Methods, Instruments and Computers*, 23(2):229–236, 1991.
5. Y. Gotoh and S. Renals. Document space models using latent semantic analysis. In *Eurospeech*, 1997.
6. R. M. Iyer. *Improving And Predicting Performance Of Statistical Language Models In Sparse Domains*. PhD thesis, Boston University, 1998.
7. M. P. Jones and J. H. Martin. Contextual spelling correction using latent semantic analysis. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, 1997.
8. R. Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language*, 1, 1996.