# The Latent Words Language Model*

**Koen Deschacht**
Department of Computer Science
K.U.Leuven, Belgium
koen.deschacht@cs.kuleuven.be

**Marie-Francine Moens**
Department of Computer Science
K.U.Leuven, Belgium
sien.moens@cs.kuleuven.be

## 1 Introduction

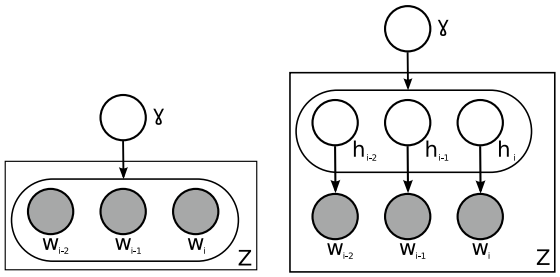
Statistical language models have found many applications in information retrieval since their introduction almost three decades ago. Currently the most popular models are n-gram models, which are known to suffer from serious sparseness issues, which is a result of the large vocabulary size $|V|$ of any given corpus and of the exponential nature of n-grams, where potentially $|V|^n$ n-grams can occur in a corpus. Even when many n-grams in fact never occur due to grammatical and semantic restrictions in natural language, we still observe and exponential growth in unique n-grams with increasing n.

Smoothing methods combine (specific, but sparse and potentially unreliable) higher order n-grams with (less specific but more reliable) lower order n-grams. (Goodman, 2001) found that interpolated Kneser-Ney smoothing (IKN) performed best in a comparison of different smoothing methods in terms of the perplexity on a previously unseen corpus. In this article we describe a novel language model that aims at solving this sparseness problem and in the process learns syntactic and semantic similar words, resulting in an improved language model in terms of perplexity reduction.

## 2 Latent Words Language Model

### 2.1 Definition

We propose a new generative model, termed the Latent Words Language Model (LWLM). LWLM uses a collection of unobserved words $H$ and introduces at every position $i$ of an observed word $w_i$ in the text an unobserved, or hidden variable $h_i$. Fig. 1 shows the structure of the model. For ease of notation, we assume a 3-gram model here, for other values of $n$ the model is defined analogously. In this model, $\gamma$ generates the 3-gram




(a) (b)

Figure 1: Graphical representation of (a) the standard 3-gram generative model and (b) the 3-gram LWLM. Grey circles represent observed variables, white circles hidden variables, the rounded box a trigram variable representing respectively $\mathbf{w}_{i-2}^i$ and $\mathbf{h}_{i-2}^i$ and the square box represents the different positions in a text of length $Z$.

of hidden words $\mathbf{h}_{i-2}^i = h_{i-2}h_{i-1}h_i$. Every hidden variable $h_i$ at position $i$ generates the observed word $w_i$. This model estimates the probability of trigram $\mathbf{w}_{i-2}^i = w_{i-2}w_{i-1}w_i$ as:

$$P_{LW}(\mathbf{w}_{i-2}^i|\gamma) = \sum_{h_{i-2}^i} P(\mathbf{h}_{i-2}^i|\gamma) \prod_{c=i-2}^{i} P(w_c|h_c)$$

It is important to see that, although the model assumes that the words are independent given the hidden variables, the hidden variables are dependent on each other since they belong to the same trigram, and thus introduces an indirect dependency between the words.

### 2.2 Parameter estimation

The LWLM model contains two probabilistic distributions $P(w_i|h_i)$ and $P(\mathbf{h}_{i-2}^i|\gamma)$, that need to be learned from a training text $\mathbf{T}_{train} =< w_0...w_z >$ of length $Z$. We keep a list of estimates of the hidden variables $\mathbf{H}_{train} =< h_0...h_z >$, where $h_i$ generates $w_i$ at every position $i$. Since the hidden variables $h_i$ are not observed in the

* The work reported in this paper was supported by the EU-IST project CLASS (IST-027978).

training set, we need to resort to a procedure to iteratively estimate these variables.

We first set an initial value for $h_i$ at every position $i$ in the training text. We train a standard n-gram language model using interpolated Kneser-Ney smoothing on the training set. We then select, at every position in the training set, a random value for the hidden variable according to the distribution of possible words given by this language model.

We use Gibbs sampling to improve this initial estimate. Gibbs sampling is a Markov Chain Monte Carlo method that generates a number of estimates $H^0_{train}, ...., H^Q_{traing}$ for the hidden variables. In every iteration $\tau$, Gibbs sampling generates a new estimate $H^{\tau+1}_{train}$ according to the previous estimate $H^{\tau}_{train}$ by selecting a random position $j$ and updating the value of the hidden variable at that position. The probability distributions $P^\tau(w_j|h_j)$ and $P^\tau(\mathbf{h}^j_{j-2}|\gamma)$ are constructed by collecting the counts from all positions $i \neq j$. These distributions are then used to compute the probability distribution of the unobserved variable $h_j$ given the word $w_j$ and the sequences of the 3-grams $\mathbf{h}^{j-1}_{j-2}$, $\mathbf{h}^j_{j-1}$ and $\mathbf{h}^{j+1}_j$ .

$$P^\tau(h_j|w_j, \mathbf{h}^{j-1}_{j-2}, \mathbf{h}^{j+2}_{j+1}, \gamma) = \frac{P^\tau(w_j|h_j)\sum_{c=j-2}^{j} P^\tau(\mathbf{h}^{c+2}_c|\gamma)}{\sum_{h_j} P^\tau(w_j|h_j)\sum_{c=j-2}^{j} P^\tau(\mathbf{h}^{c+2}_c|\gamma)}$$

We select a new value for the hidden variable according to this distribution and place it at position $j$ in $H^{\tau+1}_{train}$. The current estimate for all other unobserved words remains the same. We perform a large number of sampling iterations and save the values of the unobserved variables at specified intervals. The collection of saved samples is then used to construct the final model.

### 2.3 Evaluation

We perform experiments on three different corpora: the *Reuters*[1] and *APNews*[2] corpora consist of short news texts distributed by respectively the Reuters and the Associated Press news agencies. The *EnWiki* corpus consists of the first 500 articles from the English Wikipedia. For every corpus we used $5M$ words for training, $100K$ words as held-out data for the optimization of various parameters and $100K$ for testing.

[1] see http://www.daviddlewis.com/resources

[2] Identical to the corpus used in (Bengio et al., 2003). We thank the authors for making this corpus available.

| Method | Reuters | APNews | EnWiki |
|---|---|---|---|
| IKN 3-gram | 113.15 | 132.99 | 160.83 |
| IBM 3-gram | 108.38 | 125.65 | 149.21 |
| LWLM 3-gram | 99.12 | 116.65 | 148.12 |
| IKN 4-gram | 102.08 | 117.78 | 143.20 |
| IBM 4-gram | 102.91 | 112.15 | 142.09 |
| LWLM 4-gram | 93.65 | 103.62 | 134.68 |

Table 1: Results in terms of perplexity of different language models. See text for details.

Table 1 gives the perplexity results of the LWLM on three different corpora. We compared our method with interpolated Kneser-Ney smoothing (IKN) and with fullibmpredict (IBM)[3], the best class based language model described in (Goodman, 2001). The different parameters for every model (discount factors, interpolation weight, number of classes, ...) were optimized on the held-out set for every data set separately.

We see that the LWLM performs significantly better than both other models, with a maximum improvement of $12.40\%$ (on *Reuters*, using 3-gram counts) on Interpolated Kneser-Ney smoothing. The smallest improvement on this model was $5.87\%$, on *EnWiki* using 4-grams.

## 3 Conclusions and future work

We have presented the Latent Words Language Model and shown that this model partially solves the sparseness problem posed by traditional n-gram models, resulting in a maximum improvement on state-of-the-art language models of $12.40\%$. Furthermore, informal evaluation revealed that the learned similarities represent to a high degree the meaning of the word, retrieving synonyms and closely related words. In the future we would like to perform a formal analysis of these possibilities, employing the model for word sense disambiguation, semantic role annotation or named entity recognition.

## References

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

J. Goodman. 2001. A bit of progress in language modeling. *Technical report, Microsoft Research.*

[3] Both models are implemented by the authors, based on (Goodman, 2001)