Acta Cybernetica **18** (2007) 303–314.

Similarity Based Smoothing In Language Modeling

Zoltán Szamonek* and István Biró†

Abstract

In this paper, we improve our previously proposed Similarity Based Smoothing (SBS) algorithm. The idea of the SBS is to map words or part of sentences to an Euclidean space, and approximate the language model in that space. The bottleneck of the original algorithm was to train a regularized logistic regression model, which was incapable to deal with real world data. We replace the logistic regression by regularized maximum entropy estimation and a Gaussian mixture approach to model the language in the Euclidean space, showing other possibilities to use the main idea of SBS. We show that the regularized maximum entropy model is flexible enough to handle conditional probability density estimation, thus enable parallel computation tasks with significantly decreased iteration steps. The experimental results demonstrate the success of our method, we achieve 14% improvement on a real world corpus.

Keywords: language modeling, word similarity, maximum entropy, SVD

1 Introduction

Data Sparseness in Language Modeling. A central problem in statistical language modeling is data sparseness, *i.e.* we do not have enough data to accurately estimate the probability distribution of words. Even in the context of bigrams, training an efficient model requires huge training corpus, not to mention training higher order distributions. This phenomenon is also known as curse of dimensionality. Probability density estimation thus requires some form of approximation techniques.

Smoothing. Generally, smoothing methods reserve some mass in the probability model for unseen events, and then assign that mass to these events as a function of their marginal frequencies. A great deal of techniques have been proposed for smoothing of n-gram models, including discounting, recursively backing-off to lower order n-grams, linearly interpolating n-grams of different orders. An excellent survey of these and some other techniques can be found in [6].

*Eötvös Loránd University, Pázmány P. sétány 1/c, Budapest 1117, E-mail: zszami@elte.hu

†Computer and Automation Research Institute of the Hungarian Academy of Sciences, Kende u. 13-17, Budapest 1111, E-mail: ibiro@sztaki.hu

Clustering models make use of smoothing ideas, but also attempt to make use of similarities between words. In the class-based n-gram estimation [5], a greedy algorithm searches the space of clustering to find one that increases the data likelihood the most. The method is extended to handle trigram [17], where a heuristic randomization was used to speed-up the clustering process with only a slight loss in performance.

These early works rely on short histories (such as bigram or trigram) and therefore tend to produce syntactically-oriented clusters. Another approach proposed by [2] exploits larger contexts (*i.e.* documents) between words and results in clusters that are semantically oriented. The idea is to exploit that documents can be thought of as a semantically homogeneous set of sentences. The algorithm forms a word-document matrix given the training data, performs singular-value decomposition (SVD) on the resulting matrix, and then clusters over the projections according to a well-chosen measure.

Similarity Information. Clustering is a special case of constructing similarity information over the words (or word-forms) of a language. [8] proposed to use a kernel-based smoothing technique where the kernel function is built using similarity information derived from word co-occurrence distributions. [16] investigated a Markov chain model that exploits *a priori* similarity information, whose stationary distribution was used for solving prepositional phrase attachment problems.

Our contribution. In our previous work [4], we presented a special case of the general Similarity Based Smoothing (SBS) method. We showed, that the similarity based methods improve language models' performance, and tested the approach on bigram estimates. Regularized logistic regression struggled with computational difficulties when estimating the parameters on real world data, although SBS performed well on test cases with lower dimensions, suggesting that the approach may introduce significant improvement to language models.

In this paper, we show that a more general and robust estimation method, the regularized maximum entropy, is able to handle real world problems as well. Our framework is also flexible enough to handle conditional probability density estimation, thus enable parallel computation tasks with significantly decreased iteration steps. The experimental results demonstrate 14% improvement on a real world corpus, which proves the efficiency of the approach.

2 Language modeling based on similarity information

The goal of language modeling is to provide an accurate estimation for $P(w|h)$, where $w \in \mathcal{V}$ is a word from the vocabulary, and $h \in \mathcal{V}^*$ is some history. Restricting the history to one word, we arrive at the well known bigram model (first order Markov model). For the sake of simplicity in this paper we will deal only with bigrams (the methods proposed can be easily extended to higher order n-grams). In what follows the notation $p(y|x)$ will be used to denote the bigram probabilities.

Given appropriate basis functions, $\{\varphi_i\}$, the log-conditional probabilities of the bigrams can be written in the form

$$\log p(y|x) \propto \sum_{i \in \mathcal{I}} \alpha_i \varphi_i(x, y). \tag{1}$$

Clearly, the simplest choice is to let $\mathcal{I} = \mathcal{V} \times \mathcal{V}$ and use the so-called Euclidean basis functions, $\varphi_i(x, y) = \varphi_{(i_1,i_2)}(x, y) = \delta(i_1, x)\delta(i_2, y)$, where $\delta(\cdot, \cdot)$ is the Kronecker's delta function. With this basis function set, fitting this model to some dataset is equivalent to maximum likelihood training. The main idea in the paper is that given some similarity information over $\mathcal{V}$ it might be possible to construct another basis function set that facilitates generalization to unseen cases. Since the basis functions that we will construct may form an overcomplete representation, we will resort to a form of regularization to prevent overfitting.

2.1 Incorporating similarity information

Let $\mathcal{V} = \{w_1, \ldots, w_{|\mathcal{V}|}\}$ be the words in the vocabulary. In order to estimate the $p(y|x)$ conditional probability distribution, where $(x, y) \in \mathcal{W}$, we used the similarity information hidden in the context. Assume that the similarity information between words is represented as an undirected weighted graph $G = (\mathcal{V}, E, W)$, where $E \subset \mathcal{V} \times \mathcal{V}$ are the edges and $W : E \to \mathbb{R}_0^+$ assigns non-negative weights to word-pairs. For mathematical convenience, we use W to denote the $|\mathcal{V}| \times |\mathcal{V}|$ weight matrix, where the $(i, j)^{\text{th}}$ entry of W is the weight of the edge $i \to j$. The idea is to construct basis functions that respect the similarity information in W. One way to accomplish this is to use spectral graph clustering which is known to yield basis functions that can be used to represent any square integrable function over G [7]. In fact, spectral graph clustering methods construct basis functions that are natural for constructing geodesically smooth global approximations of the target function. In other words, smoothness respects the edges of the graph. In our case this means that similar words (or word-pairs) will be assigned similar probabilities.

In the particular methods we have chosen, the basis functions are computed using the singular value decomposition of the matrix $P = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$. Here D is the diagonal *valency* matrix; its diagonal elements are defined by $d_i = \sum_j w_{ij}$. This operator is spectrally similar to the normalized graph Laplacian operator, $L = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$. In fact, elementary algebra yields that $I - L = D^{-\frac{1}{2}} W D^{-\frac{1}{2}} = P$. The spectrum of the graph Laplacian is known to have a close relationship to global properties of the graph, such as "dimension", bottlenecks, clusters, mixing times of random walks, etc. The spectral decomposition method employed is motivated as follows: The smoothness of a function, $f : \mathcal{V} \to \mathbb{R}$ on the graph G can be measured by the Sobolev-norm $\|f\|_G = \sum_{v \in \mathcal{V}} |f(v)|^2 d_v + \sum_{(u,v) \in E} (f(u) - f(v))^2 w_{uv}$. The first term here controls the size of the function, whilst the second controls the gradient. Using this norm, the objective to exploit similarity information can be expressed as the desire to find a log-likelihood function whose G-norm is small. Now, the projection of a function f on the linear function space spanned by the top k eigenvectors of the Laplacian is the smoothest approximation to f with k basis

functions, in the sense of the G-norm. Hence, influenced by [10] we decomposed P using singular value decomposition: $P = USV^T$. For practical purposes, we truncated the SVD of P in such a way that the $\|\mathcal{P}_k\|_F = 0.9\,\|\mathcal{P}\|_F$, where $\|\cdot\|_F$ is the Frobenius norm, and $\mathcal{P}_k = U_k S_k V_k^T$ is the truncated version of P where only the k column vectors of U and k row vectors of V corresponding to the k largest singular values of S are kept. For proper normalization, the singular vectors in U_k are multiplied by the square root of the respective singular values [9, 10]: $U'_k = U_k S_k^{1/2}$.

Now, the basis functions are obtained using the columns of U_k: Denoting these columns by $\varphi_1, \ldots, \varphi_k$, $\varphi_{(i_1,i_2)}(x,y) = \delta(i_1,y)\varphi_{i_2}(x)$, where $i_1, x, y \in \mathcal{V}$, $i_2 \in \{1,\ldots,k\}$. When the similarity information is unreliable we may add the Euclidean basis functions, $\varphi'_i(x,y) = \varphi_{(i_1,i_2)}(x,y) = \delta(i_1,x)\delta(i_2,y)$, to the set obtained. This way even when the automatically constructed basis functions are useless, the method still has a chance to recover. To handle unknown words, one may resort to Nyström's method [1]: In order to extend a singular vector φ_i to a new word, z, it suffices to know w_{xz} for all $x \in \mathcal{V}$ in the set of known words. In fact, $\sum_{x\in\mathcal{V}} \frac{w_{xz}}{\sqrt{d_x d_z}}\varphi_i(x)$ can be shown to be a good approximation.

In the experiments below we always used a single similarity graph, though the algorithm has the natural ability to use more graphs by simply merging the set of resulting basis functions. We may use all the basis functions, as well as all the products of two basis functions [15].

Algorithm 1 Similarity Based Smoothing (SBS)

Input: similarity information between words, training data
Output: estimated probability distribution

1. Build a similarity graph between words $G = (V, E, W)$
2. Calculate the normalized matrix of G: $P = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$
3. Determine the SVD decomposition of $P = USV^T$
4. Calculate the mapping operator from the singular vectors of top singular values: $\Phi = U_k S_k^{\frac{1}{2}}$
5. Train the Estimation Model's weight parameters using the training data (Maximum entropy estimation, Logistic regression, Gauss estimation)

2.2 Logistic regression based learning

In our previous work [4], we implemented SBS using logistic regression with Laplacian and Gauss priors [14]. Logistic regression (LR) was already able to achieve good results, but unfortunately it was infeasible on real world data with large vocabularies.

Int this paper, we investigate joint maximum entropy models with regularization constraints which is a generalization of LR models, and provide better scaling properties. Our regularized maximum entropy implementation significantly outperformed the previous LR estimation in terms of computational time, which made real world experiments possible. We did the synthetic experiments for both cases, and it turned out, that using regularized maximum entropy estimation, in most of the cases yielded better model quality as well.

2.3 Maximum entropy model

Incorporating the similarity information, the basis function takes the form of $\varphi_{(i_1,i_2)}(x,y) = \delta(i_1,y)\varphi_{i_2}(x)$, substituting into (1) and rose to the exponent:

$$p_\lambda(x,y) = \frac{1}{Z_\lambda} e^{\left(\sum_{i_1,i_2} \lambda_{i_1,i_2}\delta(i_1,y)\varphi_{i_2}(x)\right)} = \frac{1}{Z_\lambda} e^{\left(\sum_{i_2} \lambda_{y,i_2}\varphi_{i_2}(x)\right)} = \frac{1}{Z_\lambda} e^{\left(\lambda_y^T \varphi(x)\right)} \quad (2)$$

where Z_λ is the normalizing factor, $\lambda = \left[\lambda_1, \ldots, \lambda_{|\mathcal{V}|}\right]^T$ is the matrix of weight parameters to be learned, λ_y contains the weight parameters of word y, and $\varphi(x)$ is the vector formed from the basis functions evaluated at x. We shall call $\varphi(x)$ the feature vector associated with x. From $p(x,y)$, it is straightforward to calculate the conditional probability $p(y|x)$.

The last term in (2) is a Gibbs distribution with parameter vector λ_y. According to [3], the maximum likelihood Gibbs distribution is equivalent to the maximum entropy distribution estimated from the training data. Assuming that the data sequence $\mathcal{D} = (v_1, \ldots, v_N)$, $v_i \in \mathcal{V}$ is generated by a first order Markov model where the transition probabilities can be modeled by (2), the data log-likelihood takes the following form:

$$L_{\mathrm{ML}}(\lambda|\mathcal{D}) \quad = \quad \sum_{j=2}^{N} \left[\lambda_{v_j}^T \varphi(v_{j-1}) - \log Z_\lambda\right] \quad (3)$$

The maximum likelihood (ML) estimate of the parameter vectors is the vector that maximizes this function. In order to prevent overtraining it is advisable to prefer smooth solutions, especially if the number of basis functions is big (in other words, feature vectors are high dimensional). One way to enforce smoothness is to introduce a prior distribution over the parameters λ_{ij}. Several choices are possible for such priors. In this work we studied the behavior of the method using the Laplacian-prior, $p(\lambda_{ij}) \propto \beta_{ij} \exp(-\beta_{ij}\,|\lambda_{ij}|)$. In this case, the training problem becomes the maximization of the following objective function:

$$L_{\mathrm{MAP}}(\lambda|\mathcal{D}) = \sum_{j=2}^{N} \left[\lambda_{v_j}^T \varphi(v_{j-1}) - \log Z_\lambda\right] + \sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{k} \beta_{ij}\,|\lambda_{ij}| \quad (4)$$

i.e. an ℓ^1 penalty term is added to the ML objective function.

Applying priors to avoid overtraining is called as regularization in maximum entropy modeling. We implemented a state-of-the-art regularized maximum entropy [11] algorithm to find the solution for equation (4).

Computational speed-up. Instead of estimating $p(x, y)$ joint probability, we estimated $p(x|y)$. The reasons were two-fold, firstly in this case we were able to train these conditional models parallel, secondly because the number of parameters to be estimated reduced to k (dimension of feature vectors) from $|V| \cdot k$, the number of iterations reduced adequately in all $p(x|y)$ instance resulting in an overall decrease in running time.

2.4 Gauss based estimation

As another approach, it is obvious to try some simple models, when the words are mapped to an Euclidean space. One such model can simply fit a Gaussian mixture model on the training data, and use it to estimate (maybe only the missing) statistics.

$$P(y|x) = \frac{1}{Z_x} \sum_{x' \in \mathcal{V}} P'(y|x') \cdot e^{\left(-\frac{||\varphi(x') - \varphi(x)||}{\sigma^2}\right)}$$

We used the maximum likelihood estimate for P', and Z_x was a normalization factor such that $\sum_y P(y|x) = 1$. If the distance of words in the Euclidean space reflects the similarity between words (as in our case), this may lead to a good estimate.

3 Experimental Results

We compared the performance of our approach, SBS, on synthetic and real-world data to interpolated Kneser-Ney smoothing (IKN). IKN is considered as one of the best smoothing techniques [13]. As special cases of SBS, we present the results for the maximum entropy model (Maxent), the logistic regression estimation (Logreg) and the Gauss mixture approach (only Maxent is calculated for real world data).

In order to evaluate the methods we calculated the empirical cross-entropy of the trained model on held-out data, $w_1, \ldots, w_N$:

$$H_N(p, \hat{p}) = -\frac{1}{N} \sum_{i=1}^{N} \log_2 \hat{p}\,(w_i|w_{i-1})\,. \tag{5}$$

Here $\hat{p}(w_i|w_{i-1})$ is the probability assigned to the transition from w_{i-1} to w_i by the model and p denotes the true distribution. (For the synthetic datasets we have $w_i \sim p(\cdot|w_{i-1})$.) Since by assumption, the held-out data has the same distribution as the training data, by the Shannon-McMillan-Breiman theorem we know that (under mild assumptions) H_N is close to the cross-entropy of $\hat{p}$ with respect to the true distribution p of the data.

For each test, we calculated the cross entropies for the IKN estimate and the proposed similarity based smoothing techniques. In the case of synthetic datasets

we also estimated the entropy of the models by using the true transition probabilities in (5), the corresponding points on the graphs are labeled as 'Model'.

3.1 Tests on Synthetic Data

The model generating the synthetic data. In order to develop a good understanding of the possible advantages and limitations of the proposed method, we tested it in a controlled environment where data was generated using some designated bigram model. In order to make the test interesting, we decided to use "clustered" Markov models of the following form: the probability of observing y given x is computed by

$$P(y|x) = P(\, y \,|\, c(y)\,)\, P(\, c(y) \,|\, c(x)\,), \tag{6}$$

where $c(x)$ is the cluster of symbol (word) x ($c : \mathcal{V} \to \mathcal{C}$ with $|\mathcal{C}| < |\mathcal{V}|$). The idea is that the next observation depends on the past observation x only through its class, $c(x)$ and hence two past observations, x and x' yield to identical future distributions whenever $c(x) = c(x')$. Note that when $P(y|c) = \delta(c(y), c)$ then the model can be thought of as a Hidden Markov Model (HMM) with state transitions defined over $\mathcal{C}$, whilst when the observation probabilities are unconstrained then in general no HMM with $|\mathcal{C}|$ *states* is equivalent to this model. In any way, the model can be specified by two matrices, (A, B), with $A_{c_1,c_2} = P(c_2|c_1)$ ($A \in \mathbb{R}^{|\mathcal{C}|\times|\mathcal{C}|}$) and $B_{y,c} = P(y|c)$ ($B \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{C}|}$).

For computing the matrix A, we start with a permutation matrix of size $|\mathcal{C}| \times |\mathcal{C}|$ and perturb it with random noise so that *(i)* all transition probabilities are nonzero and *(ii)* for each state the number of "significant" transitions lies between 1 and $\frac{|\mathcal{C}|}{4}$.

For computing B, we start from the idealized block-structured matrix with $B'_{yc} \propto \delta(c(y), c)$ and then perturb it according to

$$B_{yc} \propto B'_{yc} + \delta(1 + \gamma z'_{yc}),$$

where the elements of z'_{yc} are independent random variables uniformly distributed in $[-0.5, 0.5]$. If $\delta = 0$ then the block structure of the source is clearly identifiable based on the outputs: Given an observation y and knowing $\mathcal{C}$ we can infer with probability one that the hidden state is $c(y)$ and as noted before the model collapses to a hidden Markov model with $\mathcal{C}$ states. When δ is non-zero this structure is blurred. In the experiments we used $\delta = 0.1$ whilst we let γ change in $[0, 1]$. Note that $\gamma = 1$ roughly corresponds to a noise level of 5% in the observations.

Similarity information provided for SBS. We controlled how well the similarity information provided for SBS reflects the actual block structure of the data. The perfect similarity information assigns 0 to observations (words) in different clusters, whilst it assigns the same positive value, say 1, to observations in the same cluster. The corresponding matrix is denoted by S ($S \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$): $S_{xy} = \delta(c(x), c(y))$.

We then disturbed S by adding noise to it. For this a random $|\mathcal{V}| \times |\mathcal{V}|$ matrix (Z) was created whose entries were independent, uniformly distributed random variables taking values in $[0, 1]$. Given a noise parameter, $\epsilon \geq 0$, the perturbed matrix is computed by

$$S_\epsilon = S + \epsilon\left((-S \odot Z) + ((\mathbf{1} - S) \odot Z)\right).$$

Here $U \odot V$ denotes the component wise product of matrices U and V, $\mathbf{1}$ denotes the matrix with all of its entries being 1. In words, increasing ϵ reduces the inter-cluster similarity and increases between-cluster similarity. In the extreme case when $\epsilon = 1$ all similarity information is lost.

Training and test datasets. A training dataset normally contains $N_{\text{tr}} = 300$ observation sequences (except when we experiment by changing this parameter), each having a random length that was generated by drawing a random number L from a normal distribution with mean 11 and variance 6 and then setting the length to $\max(2, L)$. The test dataset (separate from the training dataset) had 5000 sequences which was generated using the same procedure. All measurements were repeated 100 times and the average values are presented.

3.2 Results

We performed a sensitivity analysis to test the effect of how the various parameters influence the results. In particular, we studied the performance of SBS as a function of the observation noise, γ, that masks the identifiability of the block structure, the noise in the similarity matrix (ϵ) that gradually decreases the quality of the similarity information available for SBS, the number of training sentences (N_{tr}) and the cluster structure. These parameters were changed one by one, whilst the others are kept at their default values which were $\gamma = 0.2$, $\epsilon = 0.1$, $N_{\text{tr}} = 300$. The default cluster structure was to have 6 clusters, each having $30, 20, 10, 5, 5$ and 5 words respectively (so that some clusters were bigger, some were smaller). Thus $|\mathcal{V}|$, was kept at 75.

Sensitivity to noise masking the block-structure. When $\gamma = 0$, the observations can be used directly to infer the underlying classes and the estimation problem is easier. When the block-structure masking noise is increased the problem becomes harder.

Figure 1 shows the results of the measurements. It can be observed that the proposed methods perform significantly better over the considered spectrum of γ than IKN. On the other hand, SBS Logreg and SBS Maxent were able to maintain its estimation accuracy for the whole range of γ, while the Gauss estimate had a slight loss of quality.

Robustness against the degradation of the similarity information. In the next set of experiments we investigated the sensitivity of the methods to the

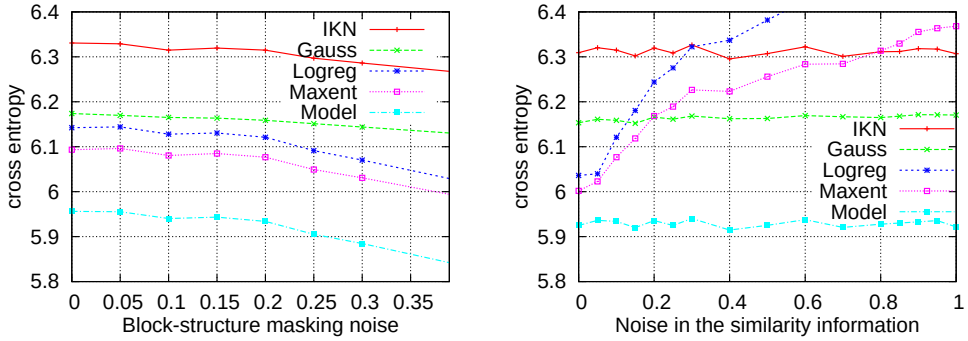

Figure 1: Left: The cross-entropy of models built with various algorithms as a function of the block-structure masking noise parameter (γ). Right: Cross-entropy of the noise parameter, ϵ, governing the quality of the similarity information.

quality of the similarity information. For this we gradually increased the similarity-information noise parameter, ϵ, from 0 to 1. As we have discussed, when $\epsilon = 0$ the similarity information is perfect, whilst when $\epsilon = 1$ all similarity information is lost.

As expected, when the similarity information gets weaker, the performance of SBS methods degrade and converge to that of the ML-estimated bigram model (cross entropy 9.8 due to lack of all bigrams in training data). It is notable that even when $\epsilon = 0.7$ (when the similarity information is already very poor) SBS Maxent performs as well as IKN. The reason is that although in these experiments we did not add the Euclidean basis functions to the set of basis functions, we can expect the spectrum of a high-noise similar matrix to be uniform, hence covering 90% of the spectrum will add almost all singular vectors to the basis and thus the model automatically retains its power. This effect may be the reason for the Gauss estimation, which seems to maintain it's performance during this experiment. Namely, as the noise gets bigger, the more features will be kept, and the Gauss estimate seems to handle this very well (the training of the Gauss estimate does not depend on the number of features as long as it contains the necessary information, while LR and Maxent had a linear growth in the number of parameters).

Investigation of the $\epsilon = 1$ case: Gauss estimation seems to result in a nearly unigram distribution: $P(y|x) = P_{Gauss}(y)$, which seems to be a very good model for this data. Due to the small corpus size (about 3000 bigrams for 5625 word pairs) and lack of similarity information, other models (including IKN) were trying to capture the small differences in counts, while Gauss tried to fit normal distributions randomly over the space resulting in nearly unigram distribution. However, we must note, that there is still a significant difference between the original model and the estimations.

Sparsity of training data. The main promise of SBS is that by using the similarity information it is capable of achieving better performance even when the available training data is limited. In order to validate this, we performed a set of experiments when the number of sentences in the training data was varied. The results are shown in Figure 2. All SBS models were able to outperform IKN for a wide range of values of N_{tr}, although for very small N_{tr} IKN seems to be better than Maxent and Logreg. We see, that Gauss performs very well at the beginning, but it does not converge quickly to the theoretical optimum 'Model'.

Cluster structure. We tested SBS with large variety of cluster setups, ranging from 1 to 15 clusters, and with vocabulary sizes between 7 and 110. The results are summarized on Figure 2.

It is clear that if the number of clusters is small or when all the words are in different clusters, then there is no significant structural information in the similarity. It is notable that in such cases SBS Logreg was still able to perform as well as IKN. On the other hand, if there is a significant hidden structure in the model, all SBS methods greatly outperform IKN. In this experiment, $H(p, \hat{p}_{\mathrm{ikn}})$ ranges 2.5 to 6.5.

Gauss seems to be able to perform well only with large vocabularies. Maybe this is due to the observation made at sparsity of training data experiments, that it does not really converge to the optimal model when more data is available.

clusters $\hat{p}_* =$	$H(p, \hat{p}_{\mathrm{ikn}}) - H(p, \hat{p}_*)$		
	$\hat{p}_{\mathrm{Gauss}}$	$\hat{p}_{\mathrm{Logreg}}$	$\hat{p}_{\mathrm{Maxent}}$
1,1,1,1,1,1,1	-0.126	0.000	-1.933
10	-1.713	0.017	0.017
10x2	-0.044	0.089	0.073
10x3	-0.016	0.178	0.160
10x5	0.097	0.254	0.266
10,5	-0.050	0.044	0.015
10,7,5	-0.100	0.102	0.087
10,7,5,3	-0.012	0.128	0.128
30,20,10	0.161	0.284	0.309
30,20,10,5x3	0.156	0.194	0.238
30,20,10,5x3,1x3	0.161	0.151	0.219
30,20,10,5x3,1x6	0.169	0.113	0.209
30,20,10,5x3,1x9	0.166	0.073	0.194

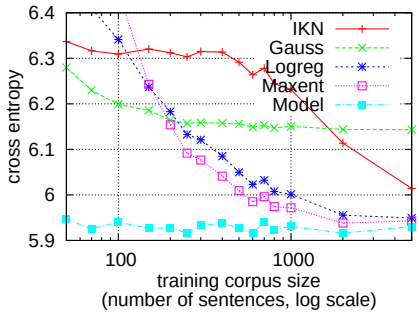

Figure 2: Left: The cross-entropy decrease of SBS as compared with IKN for a number of different cluster structures. Right: The cross-entropy of models built with various algorithms as a function of the number of sentences in the training data (N_{tr}).

3.3 Tests on Real Data

The SBS Maxent model was built using 55,000 sentences from the Wall Street Journal (WSJ) corpus. We tested the model on the Brown corpus, which contained 47,000 sentences.

We reduced the feature space by using only the most relevant features from Φ and we did not add the Euclidean basis functions to the features. With these constraints, the model is not in its most general form, but this relaxation reduced the computational requirements significantly. To compensate the lack of information, we use IKN to estimate high frequency bigrams, and leave the low frequency case ($C(x) < 2$ when estimating $P(y|x)$) for SBS where IKN doesn't perform well.

The similarity graph was constructed according to the number of word occurrences in the different senses of each word (WordNet [12] overview property). Weights for the similarity graph were constructed according to weight information in the overview.

Combined vocabulary size from WSJ and Brown contained nearly 70,000 words. We reduced this to 18,000 words by merging the infrequent ones, which were mainly numerical values and named entities.

The cross entropy using IKN trained on WSJ and tested on Brown corpus was 9.47, while using our approach for the sparse part of the IKN estimate, the cross entropy was only 9.27. This is a 14% improvement in terms of Perplexity.

This result shows, that using general similarity graph from WordNet we were able to improve the estimation even on a different corpus.

4 Conclusions

The improved Similarity Based Smoothing algorithm is now capable to work on real world data. We have found that regularized maximum entropy models provide a more flexible way of learning the parameters of the probability distribution to be estimated. We have also investigated that a simple Gaussian mixture model over the words might perform well, if relevant similarity information is present in the underlying data. We achieved 14% perplexity improvement over Interpolated Kneser-Ney smoothing on real world corpus using the maximum entropy SBS method.

Using WordNet, we created a similarity graph over words. This graph is an external information source, we built it independently from the training corpus, thus we expect this graph to be domain independent. Real world experiments showed that maximum entropy version of SBS was able to utilize this large graph and improved the bigram estimation.

The current work shows that the expectations with SBS are true. It is able to handle large dimension real world problems as well as synthetic and low dimension problems (*e.g.* POS tag bigrams).

We also showed that this kind of word representation allows us using robust mathematical models (Logistic Regression, Maxent, Gaussian Mixtures) to conserve the topology (*i.e.* distance of points in the Euclidean space maps well to distance of words in similarity), yielding promising results.

References

[1] Baker, C. T. H. *The Numerical Treatment of Integral Equations*. Clarendon Press, Oxford, 1977.

[2] Bellegarda, Butzberger, Chow, Zen-Lu, Coccardo, and Naik. A novel word clustering algorithm based on latent semantic analysis. In *ICASSP*, 1996.

[3] Berger, A. L., Pietra, S. A. Della, and Pietra, V. J. Della. A maximum entropy approach to natural language processing. In *Computational Linguistics*, 1996.

[4] Bíró, István, Szamonek, Zoltán, and Szepesvári, Csaba. Sequence prediction exploiting similary information. In *IJCAI*, 2007.

[5] Brown, P., de Souza, P., Mercer, R., Pietra, V.J, and Lai, J. Class based n-gram models of natural language. In *Computational linguistics*, 1992.

[6] Chen, Stanley F. and Goodman, Joshua. An empirical study of smoothing techniques for language modeling. In *ACL*, 1996.

[7] Chung, Fan. *Spectral Graph Theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, 1997.

[8] Dagan, Ido, Lee, Lillian, and Pereira, Fernando C. N. Similarity-based models of word cooccurrence probabilities. *Machine Learning*, 34:43, 1999.

[9] Dhillon, Inderjit S. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*, pages 269–274, 2001.

[10] Ding, He, Zha, and Simon. Unsupervised learning: Self-aggregation in scaled principal component space. *LNCS*, 2002.

[11] Dudik, Phillips, and Schapire. Performance guarantees for regularized maximum entropy density estimation. In *COLT*, 2004.

[12] Fellbaum, Christaine, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, Massachusetts, 1989.

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

[14] Krishnapuram, Carin, Figueiredo, and Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE*, 2005.

[15] Lin, Y. Tensor product space anova models.

[16] Toutanova, Manning, and Ng. Learning random walk models for inducing word dependency distributions. In *ICML*, 2004.

[17] Ueberla, J.P. An extended clustering algorithm for statistical language models. Technical report, Simon Fraser University, 1994.