Sequence Prediction Exploiting Similarity Information

István Bíró and **Zoltán Szamonek** and **Csaba Szepesvári**
Computer and Automation Research Institute of the Hungarian Academy of Sciences
Kende u. 13-17, Budapest 1111, Hungary
Eötvös Loránd University
Pázmány P. sétány 1/c, Budapest 1117, Hungary
ibiro@sztaki.hu, zszami@elte.hu, szcsaba@sztaki.hu

Abstract

When data is scarce or the alphabet is large, smoothing the probability estimates becomes inescapable when estimating n-gram models. In this paper we propose a method that implements a form of smoothing by exploiting similarity information of the alphabet elements. The idea is to view the log-conditional probability function as a smooth function defined over the similarity graph. The algorithm that we propose uses the eigenvectors of the similarity graph as the basis of the expansion of the log conditional probability function whose coefficients are found by solving a regularized logistic regression problem. The experimental results demonstrate the superiority of the method when the similarity graph contains relevant information, whilst the method still remains competitive with state-of-the-art smoothing methods even in the lack of such information.

1 Introduction

Statistical language models (SLM) attempt to capture the regularities of a natural language using statistical methods to construct estimates of distributions of various linguistic units, such as morphemes, words, phrases, sentences, or documents. Because of the categorical nature and the large vocabulary of natural languages, statistical techniques must estimate a large number of parameters, and thus depend critically on the availability of a large amount of training data. Given the complexity of languages, such training data is rarely available. In the lack of a sufficiently large and rich dataset, simple approaches like straightforward maximum likelihood (ML) estimation tend to produce poor quality models.

Smoothing in this context is typically used to refer to techniques that combine various ML probability estimates to produce more accurate models. An enormous number of techniques have been proposed for the 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 [Chen and Goodman, 1996].

Clustering models make use of smoothing ideas, but also attempt to make use of similarities between words. In the class-based n-gram estimation [Brown *et al.*, 1992], 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 by [Ueberla, 1994], 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 span relationships (such as bigram or trigram) and therefore tend to produce syntactically-oriented clusters. Another approach proposed by [Bellegarda *et al.*, 1996] exploits large span relationships between words and results in clusters that are semantically oriented. The idea is to exploit that documents can be thought of as a semantically homogenous set of sentences. Hence, if a set of documents is available then this knowledge-source could be exploited to create semantically meaningful word-clusters. 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.

Clustering is a special case of constructing similarity information over the alphabet elements (most often over the words or word-forms of a language). [Dagan *et al.*, 1999] proposed to use a kernel-based smoothing technique where the kernel function is built using similarity information derived from word-cooccurrence distributions. They have shown up to 20% reduction in perplexity for unseen bigrams as compared to standard back-off schemes. [Toutanova *et al.*, 2004] 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 this paper we propose a method that implements a form of smoothing by exploiting similarity information of the alphabet elements. The idea is to view the log-conditional probability function as a smooth function defined over the similarity graph. The algorithm that we propose uses the eigenvectors of the similarity graph as the basis of the expansion of the log conditional probability function. The coefficients of the expansion are found by solving a regularized logistic regression problem. The experimental results demonstrate the superiority of the method when the similarity graph contains relevant information, whilst the method still remains competitive with state-of-the-art methods even in the lack of such information.

2 Sequence prediction based on similarity information

Consider a word prediction task over some vocabulary $\mathcal{V}$. Let $p(w|h)$ denote the conditional probability of word $w \in \mathcal{V}$, where $h \in \mathcal{V}^*$ is some history. If we constrain the size of histories to $n-1$ words we get the well known n-gram model. For the sake of simplicity in this paper we will only deal with bigrams (the methods proposed can be readily 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-probabilities of the bigrams can be written in the form

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

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 will 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 [Chung, 1997]. 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. It is left for future work to determine if the results presented here can be improved by using other techniques, such as diffusion wavelets by [Coifman and Maggioni, 2006].

In the particular method 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 loglikelihood 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 [Ding *et al.*, 2002] 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 [Dhillon, 2001; Ding *et al.*, 2002]: $U_k' = U_k S_k^{1/2}$.

Now, the basis functions are obtained using the columns of U_k: Denoting these columns by $\psi_1, \ldots, \psi_k$, $\varphi_{(i_1,i_2)}(x, y) = \delta(i_1, y)\psi_{i_2}(x)$, where $i_1, x, y \in \mathcal{V}$, $i_2 \in \{1, \ldots, k\}$. Since the similarity information might be unreliable we added the Euclidean basis functions, $\varphi_i'(x, y) = \phi_{(i_1,i_2)}(x, y) = \delta(i_1, x)\delta(i_2, y)$, 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 (e.g. [Baker, 1977]): In order to extend a singular vector ψ_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, $\sum_{x \in \mathcal{V}} \frac{w_{xz}}{\sqrt{d_x d_z}} \psi_i(x)$ 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. Just like in regression, we may use all the basis functions, as well as all the products of two basis functions:

$$\log p(y|x) \propto \sum_a \sum_{i \in \mathcal{I}} \alpha_i^a \varphi_i^a(x, y) + \sum_{a,b} \sum_{i,j \in \mathcal{I}} \alpha_{ij}^{ab} \varphi_i^a(x, y) \varphi_j^b(x, y) \quad (2)$$

In fact, this is a second order ANOVA model [Lin, 2000]. Higher order models (products of more than two basis functions) may also be used. In practice, however, they are rarely used since second order models typically produce sufficiently good results. When more than one model sets are combined, the number of parameters increases rapidly. Hence, the use of a regularized fitting method becomes of high importance.

2.2 Solving the regression problem

Since by assumption all of our basis functions depend only through the Kronecker delta on y, equation (1) when raised to the exponent takes the form:

$$p(y|\beta(x),\alpha) = \frac{1}{Z_\alpha} e^{\alpha_y^T \beta(x)}, \text{ where } Z_\alpha = \sum_{i=1}^{|\mathcal{V}|} e^{\alpha_i^T \beta(x)} \quad (3)$$

is the normalizing factor, $\alpha = \left[\alpha_1, \ldots, \alpha_{|\mathcal{V}|}\right]^T$ is the matrix of the weight parameters to be learned , α_y contains the weight parameters of word y, and $\beta(x)$ is the vector formed from the basis functions evaluated at x. We shall call $\beta(x)$ the feature vector associated with x.

Let the training data be the sequence $\mathcal{D} = (v_1, \ldots, v_N)$, $v_i \in \mathcal{V}$. Assuming that the data is generated by a first order Markov model where the transition probabilities can be modeled by a multinomial model (3), the data log-likelihood takes the following form:

$$\begin{aligned} l_{\mathrm{ML}}(\alpha|\mathcal{D}) &= \sum_{j=2}^{N} \Big[\alpha_{v_j}^T \beta(v_{j-1}) - \quad (4) \\ &- \log \sum_{i=1}^{|\mathcal{V}|} \exp\left(\alpha_i^T \beta(v_{j-1})\right) \Big] + \text{const} \end{aligned}$$

The maximum likelihood (ML) estimate of the parameter vector 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, if the feature vectors, β, are high dimensional). One way to enforce smoothness is to introduce a prior distribution over the parameters α_{ij}. Several choices are possible for such priors. In this work we studied the behaviour of the method when it was used either with the Laplacian-prior, $p(\alpha_{ij}) \propto \lambda_{ij} \exp(-\lambda_{ij} |\alpha_{ij}|)$, or with the Gaussian-prior, $p(\alpha) \propto \frac{1}{\lambda_{ij}} \exp(-\frac{\alpha_{ij}^2}{2\lambda_{ij}^2})$.

Assuming one of the above priors, the training problem becomes the maximization of the following objective function:

$$\begin{aligned} l_{\mathrm{MAP}}(\alpha|\mathcal{D}) = \sum_{j=2}^{N} \Bigg[\alpha_{v_j}^T \beta(v_{j-1}) - \quad (5) \\ - \log \sum_{i=1}^{|\mathcal{V}|} \exp\left(\alpha_i^T \beta(v_{j-1})\right) \Bigg] - \sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{k} \lambda_{ij} |\alpha_{ij}|^p \end{aligned}$$

i.e. an ℓ^p penalty term is added to the ML objective function. Here $p = 1$ or 2 depending on which prior one wishes to use: $p = 1$ corresponds to the Laplacian prior, whilst $p = 2$ corresponds to the Gaussian prior.

We used the SMLR java implementation of [Krishnapuram *et al.*, 2005] which implemented both the Laplacian (l^1) and Gaussian priors (l^2), to find the solution for equation (5). The sketch of the algorithm is shown on Algorithm 1.

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 Logistic Regression Model weight parameters using the training data

3 Experimental Results

We compared the performance of similarity based smoothing, SBS, on synthetic and real-world data to both plain bigram estimates (BI) and interpolated Kneser-Ney smoothing (IKN). IKN is considered as one of the best smoothing techniques [Goodman, 2001].

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

$$H_N(p, \hat{p}) = -\frac{1}{N} \sum_{i=1}^{N} \log_2 \hat{p}\,(w_i|w_{i-1})\,. \quad (6)$$

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. The cross-entropy of $\hat{p}$ and p is the sum of the entropy of p and the Kullback-Leibler divergence of $\hat{p}$ and p, a non-negative quantity. The smaller the cross-entropy, the closer the estimated model is to the true model.

For each test, we calculated the cross entropies for the ML-trained bigram, IKN and the proposed similarity based smoothing technique. In the case of synthetic datasets we also estimated the entropy of the models by using the true transition probabilities in (6). The corresponding points in 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

$$P(y|x) = P(\, y \,|\, c(y)\,)\, P(\, c(y) \,|\, c(x)\,), \quad (7)$$

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 $B_{y,c} = P(y|c)$ ($B \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{C}|}$). Let us now describe how these matrices were generated.

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 for 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 $N_{\mathrm{tr}} = 300$ 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 9 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 $\gamma = 0.2$, $\epsilon = 0.1$, $N_{\mathrm{tr}} = 300$. The default cluster structure was to have 6 clusters, having observations $30, 20, 10, 5, 5$ and 5, respectively (so that some clusters were bigger, some were smaller). Thus $|\mathcal{V}|$, was kept at 75. We present the results for both the Laplacian and Gaussian priors, the corresponding results are labeled as 'SBS Gauss' and 'SBS Lapl' in the graphs.

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.

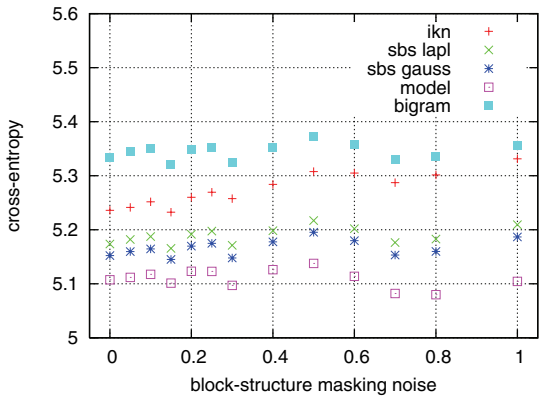

Figure 1: The cross-entropy of models built with various algorithms as a function of the block-structure masking noise parameter (γ)

Figure 1 shows the results of the measurements. It can be observed that the proposed method performs significantly better over the considered spectrum of γ than its peers: In fact, as the masking noise gets bigger, IKN's performance converges to that of the bigram model. On the other hand, SBS was able to maintain its estimation accuracy for the whole range of γ. In all of these experiments SBS with the Gaussian parameter-prior performed slightly better than SBS with the Laplace prior. We believe that this is because the Laplacian prior prefers sparse solutions and the number of basis functions is not high enough for making sparse solutions preferable.

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

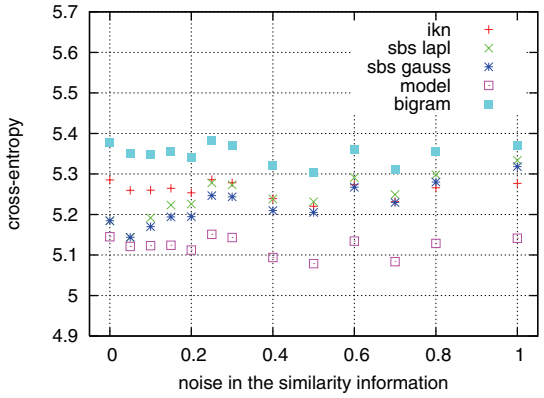

Figure 2: The cross-entropy of models built with various algorithms as a function of the noise parameter, ϵ, governing the quality of the similarity information

As expected, when the similarity information gets weaker, the performance of SBS degrades and converges to that of the ML-estimated bigram model. It is notable that even when $\epsilon = 0.7$ (when the similarity information is already very poor) SBS 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.

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 3. Both with the Gaussian and the Laplacian priors, SBS outperformed IKN for a wide range of values of N_{tr}. It is interesting to note that the Gaussian prior performed much better than any of the other methods when the number of sentences was very small.

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 Table 1.

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 was still able to perform as well as IKN. On the other hand, if there is a significant hidden structure in the model, SBS greatly outperforms IKN.

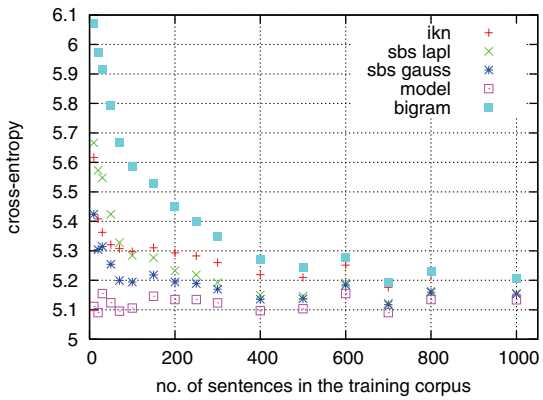

Figure 3: The cross-entropy of models built with various algorithms as a function of the number of sentences in the training data (N_{tr}).

clusters	$H(p,\hat{p}_{\text{ikn}}) - H(p,\hat{p}_{\text{sbs}})$	
	average	stddev
1,1,1,1,1,1,1	-0.0112	0.0036
10	0.1776	0.0406
10x2	0.4808	0.0246
10x3	0.8679	0.0633
10x5	1.5223	0.0708
10,5	0.3001	0.0361
10,7,5	0.5150	0.0499
10,7,5,3	0.6296	0.0311
30,20,10	1.7790	0.1691
30,20,10,5x3	1.3352	0.2011
30,20,10,5x3,1x3	1.0971	0.2303
30,20,10,5x3,1x9	0.6537	0.1141

Table 1: The cross-entropy decrease of SBS as compared with IKN for a number of different cluster structures

3.3 Tests on Real Data

The task considered is to predict POS sequences on the Wall Street Journal (WSJ) corpus. We extracted a weighted directed sub graph based on all senses in all syntactic categories from WordNet [Fellbaum, 1989] for $50,000$ words. Based on this information, a POS similarity graph (*PG*) was built as follows: using the Brown corpus[1], we first collected the possible POS categories for every word. Next we created a prior POS-tag distribution for all the words. Then for every unique word in the Brown corpus, we added the out-links to the POS graph according to the WordNet graph, using the weight of the link in the WordNet graph and the weight of the POS tags in both sides of the link. For example, if we arrive at the word *live* and if the link *live* $\rightarrow$ *bouncy* in the WordNet graph has weight w then for all possible POS tag combinations of *live* and *bouncy*, the link between the POS-pairs (x, y) is increased by $p(x|\textit{live}) \cdot p(y|\textit{bouncy}) \cdot w$. In the next step, we calculated the basis functions as outlined beforehand.

[1]The Penn Treebank Project: http://www.cis.upenn.edu/ treebank/

In another set of experiments a similarity graph was built by connecting with a constant weight those POS-tags that shared the same prefix (e.g. NN, NNP, NNS were grouped together).

The training and test corpora were kept separate, with the training corpus having $1,000$ sentences ($18,500$ POS-tags), whilst the test corpus consisted of $5,000$ sentences. The vocabulary size was 65.

When the similarity graph extracted from WordNet was used, SBS failed to achieve a good performance. Investigating the similarity graph, it turned out that it was very sparse, and it actually used 42 POS only out of the 65 POS tags. With the prefix-based similarity graph, the algorithm's performance became practically indistinguishable to that of IKN. We guess that a more principled similarity graph building method would be needed in order to achieve a performance improvements in this task. SBS in theory is better suited to problems with larger alphabets. Unfortunately, SMLR, the software tool used for training SBS scales rather poorly with the alphabet size, seriously limiting the range of feasible experiments.

Concentrating on the robustness of the estimate, we computed the perplexity reduction on those bigrams that occur only at most a few (say 4) times in the training corpus. It was found that the perplexity reduction of SBS compared to IKN is a mild 2%, whilst it is 4.5% as compared to the ML-trained bigram model. We also found if we have less sentences the SBS performs even a little better.

4 Conclusion

We have proposed a novel smoothing method to train n-gram models: The method constructs basis functions in an automated way for a maximum-entropy model by the spectral decomposition of a weighted similarity graph. It was found that the method is able to outperform some competitors when the similarity information is substantially relevant, whilst it remained competitive even when the similarity information was not helpful.

The motivation of the method is the well-known fact that in continuous domains smoothness of the target function (or density) yields faster convergence. How is it possible to achieve similar results in discrete domains? The first problem is to define an appropriate notion of smoothness. Spectral graph theory suggests that such a notion can be defined based on the Laplacian of a similarity graph and that the eigendecomposition of a normalized Laplacian (and other related operators) yields an appropriate basis for functions that are "smooth", i.e. which respect the topology (geodesics) in the graph. To our best knowledge, this is the first work that suggests the use of automatically constructed basis functions (or features) in probabilistic sequence modeling in discrete domains. Although here we only considered the simplest case, i.e., constructing bigram models, the idea is that the underlying method can be generalized to other more complex problems, ranging from n-grams to even whole-sentence language models. For such models the similarity information must be constructed over extended sequences. One possibility to accomplish this might be to use appropriately constructed string kernels. However, this investigation is left for future work.

References

[Baker, 1977] C. T. H. Baker. *The numerical treatment of integral equations*. Oxford: Clarendon Press, 1977.

[Bellegarda *et al.*, 1996] J.R. Bellegarda, J.W. Butzberger, Zen-Lu Chow, N.B. Coccardo, and D. Naik. A novel word clustering algorithm based on latent semantic analysis. In *ICASSP-96*, volume 1, pages 172–175, 1996.

[Brown *et al.*, 1992] P.F. Brown, P.V. de Souza, R.L. Mercer, V.J Della Pietra, and J.S. Lai. Class based n-gram models of natural language. In *Computational linguistics*, volume 8, 1992.

[Chen and Goodman, 1996] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *ACL*, pages 310–318, 1996.

[Chung, 1997] F. R. K. Chung. *Spectral graph theory*. Regional Conference Series in Mathematics 92. American Mathematical Society, Providence, 1997.

[Coifman and Maggioni, 2006] Ronald R Coifman and Mauro Maggioni. Diffusion wavelets. *Applied Computational Harmonic Analysis*, 2006.

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

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

[Ding *et al.*, 2002] Chris H. Q. Ding, Xiaofeng He, Hongyuan Zha, and Horst D. Simon. Unsupervised learning: Self-aggregation in scaled principal component space. In *PKDD 2002*, pages 112–124, 2002.

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

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

[Krishnapuram *et al.*, 2005] B. Krishnapuram, L. Carin, M. A. T. Figueiredo, and A. J. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Trans. Pattern Anal. Mach. Intell*, 27(6):957–968, 2005.

[Lin, 2000] Yi Lin. Tensor product space ANOVA models. *Annals of Statistics*, 28:734–755, 2000.

[Toutanova *et al.*, 2004] Kristina Toutanova, Christopher D. Manning, and Andrew Y. Ng. Learning random walk models for inducing word dependency distributions. In *ICML '04*, page 103. ACM Press, 2004.

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