

Unigram Language Models using Diffusion Smoothing over Graphs

Bruno Jedynak

Dept. of Appl. Mathematics and Statistics
Center for Imaging Sciences
Johns Hopkins University
Baltimore, MD 21218-2686
bruno.jedynak@jhu.edu

Damianos Karakos

Dept. of Electrical and Computer Engineering
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218-2686
damianos@jhu.edu

Abstract

We propose to use graph-based diffusion techniques with data-dependent kernels to build unigram language models. Our approach entails building graphs, where each vertex corresponds uniquely to a word from a closed vocabulary, and the existence of an edge (with an appropriate weight) between two words indicates some form of similarity between them. In one of our constructions, we place an edge between two words if the number of times these words were seen in a training set differs by at most one count. This graph construction results in a similarity matrix with small intrinsic dimension, since words with the same counts have the same neighbors. Experimental results from a benchmark task from language modeling show that our method is competitive with the Good-Turing estimator.

1 Diffusion over Graphs

1.1 Notation

Let $G = (V, E)$ be an undirected graph, where V is a finite set of vertices, and $E \subset V \times V$ is the set of edges. Also, let \mathcal{V} be a vocabulary of words, whose probabilities we want to estimate. Each vertex corresponds uniquely to a word, i.e., there is a one-to-one mapping between \mathcal{V} and V . Without loss of generality, we will use V to denote both the set of words and the set of vertices. Moreover, to sim-

plify notation, we assume that the letters x, y, z will always denote vertices of G .

The existence of an edge between x, y will be denoted by $x \sim y$. We assume that the graph is strongly connected (i.e., there is a path between any two vertices). Furthermore, we define a non-negative real valued function w over $V \times V$, which plays the role of the *similarity* between two words (the higher the value of $w(x, y)$, the more similar words x, y are). In the experimental results section, we will compare different measures of similarity between words which will result in different smoothing algorithms. The degree of a vertex is defined as

$$d(x) = \sum_{y \in V: x \sim y} w(x, y). \quad (1)$$

We assume that for any vertex x , $d(x) > 0$; that is, every word is similar to at least some other word.

1.2 Smoothing by Normalized Diffusion

The setting described here was introduced in (Szlam et al., 2006). First, we define a Markov chain $\{X_t\}$, which corresponds to a random walk over the graph G . Its initial value is equal to X_0 , which has distribution π_0 . (Although π_0 can be chosen arbitrarily, we assume in this paper that it is equal to the empirical, unsmoothed, distribution of words over a training set.) We then define the transition matrix as follows:

$$T(x, y) = P(X_1 = y | X_0 = x) = d^{-1}(x)w(x, y). \quad (2)$$

This transition matrix, together with π_0 , induces a distribution over V , which is equal to the distribu-

tion π_1 of X_1 :

$$\pi_1(y) = \sum_{x \in V} T(x, y) \pi_0(x). \quad (3)$$

This distribution can be construed as a *smoothed* version of π_0 , since the π_1 probability of an unseen word will always be non-zero, if it has a non-zero similarity to a seen word. In the same way, a whole sequence of distributions π_2, π_3, \dots can be computed; we only consider π_1 as our smoothed estimate in this paper. (One may wonder whether the *stationary* distribution of this Markov chain, i.e., the limiting distribution of X_t , as $t \rightarrow \infty$, has any significance; we do not address this question here, as this limiting distribution may have very little dependence on π_0 in the Markov chain cases under consideration.)

1.3 Smoothing by Kernel Diffusion

We assume here that for any vertex x , $w(x, x) = 0$ and that w is symmetric. Following (Kondor and Lafferty, 2002), we define the following matrix over $V \times V$

$$H(x, y) = w(x, y) \delta(x \sim y) - d(x) \delta(x = y), \quad (4)$$

where $\delta(u)$ is the delta function which takes the value 1 if property u is true, and 0 otherwise. The negative of the matrix H is called the Laplacian of the graph and plays a central role in spectral graph theory (Chung, 1997). We further define the heat equation over the graph G as

$$\frac{\partial}{\partial t} K_t = H K_t, \quad t > 0, \quad (5)$$

with initial condition $K_0 = I$, where K_t is a time-dependent square matrix of same dimension as H , and I is the identity matrix. $K_t(x, y)$ can be interpreted as the amount of heat that reaches vertex x at time t , when starting with a unit amount of heat concentrated at y . Using (1) and (4), the right hand side of (5) expands to

$$H K_t(x, y) = \sum_{z: z \sim x} w(x, z) (K_t(z, y) - K_t(x, y)). \quad (6)$$

From this equation, we see that the amount of heat at x will increase (resp. decrease) if the current amount of heat at x (namely $K_t(x, y)$) is smaller

(resp. larger) than the weighted average amount of heat at the neighbors of x , thus causing the system to reach a steady state.

The heat equation (5) has a unique solution which is the matrix exponential $K_t = \exp(tH)$, (see (Kondor and Lafferty, 2002)) and which can be defined equivalently as

$$e^{tH} = \lim_{n \rightarrow +\infty} \left(I + \frac{tH}{n} \right)^n \quad (7)$$

or as

$$e^{tH} = I + tH + \frac{t^2}{2!} H^2 + \frac{t^3}{3!} H^3 + \dots \quad (8)$$

Moreover, if the initial condition is replaced by $K_0(x, y) = \pi_0(x) \delta(x = y)$ then the solution of the heat equation is given by the matrix product $\pi_1 = K_t \pi_0$. In the following, π_0 will be the empirical distribution over the training set and t will be chosen by trial and error. As before, π_1 will provide a smoothed version of π_0 .

2 Unigram Language Models

Let Tr be a training set of n tokens, and T a separate test set of m tokens. We denote by $n(x), m(x)$ the number of times the word x has been seen in the training and test set, respectively. We assume a closed vocabulary \mathcal{V} containing K words. A unigram model is a probability distribution π over the vocabulary \mathcal{V} . We measure its performance using the average code length (Cover and Thomas, 1991) measured on the test set:

$$l(\pi) = -\frac{1}{|T|} \sum_{x \in \mathcal{V}} m(x) \log_2 \pi(x). \quad (9)$$

The empirical distribution over the training set is

$$\pi_0(x) = \frac{n(x)}{n}. \quad (10)$$

This estimate assigns a probability 0 to all unseen words, which is undesirable, as it leads to zero probability of word sequences which can actually be observed in practice. A simple way to smooth such estimates is to add a small, not necessarily integer, count to each word leading to the so-called add- β estimate π_β , defined as

$$\pi_\beta(x) = \frac{n(x) + \beta}{n + \beta K}. \quad (11)$$

One may observe that

$$\pi_\beta(x) = (1 - \lambda)\pi_0(x) + \lambda \frac{1}{K}, \quad \text{with } \lambda = \frac{\beta K}{n + \beta K}. \quad (12)$$

Hence add- β estimators perform a linear interpolation between π_0 and the uniform distribution over the entire vocabulary.

In practice, a much more efficient smoothing method is the so-called Good-Turing (Orlitsky et al., 2003; McAllester and Schapire, 2000). The Good-Turing estimate is defined as

$$\begin{aligned} \pi_{GT}(x) &= \frac{r_{n(x)+1}(n(x) + 1)}{nr_{n(x)}}, \quad \text{if } n(x) < M \\ &= \alpha \pi_0(x), \quad \text{otherwise,} \end{aligned}$$

where r_j is the number of distinct words seen j times in the training set, and α is such that π_{GT} sums up to 1 over the vocabulary. The threshold M is empirically chosen, and usually lies between 5 and 10. (Choosing a much larger M decreases the performance considerably.)

The Good-Turing estimator is used frequently in practice, and we will compare our results against it. The add- β will provide a baseline, as well as an idea of the variation between different smoothers.

3 Graphs over sets of words

Our objective, in this section, is to show how to design various graphs on words; different choices for the edges and for the weight function w lead to different smoothings.

3.1 Full Graph and add- β Smoothers

The simplest possible choice is the complete graph, where all vertices are pair-wise connected. In the case of normalized diffusion, choosing

$$w(x, y) = \alpha \delta(x = y) + 1, \quad (13)$$

with $\alpha \neq 0$ leads to the add- β smoother with parameter $\beta = \alpha^{-1}n$.

In the case of kernel smoothing with the complete graph and $w \equiv 1$, one can show, see (Kondor and Lafferty, 2002) that

$$\begin{aligned} K_t(x, y) &= K^{-1} \left(1 + (K - 1)e^{-Kt} \right) \quad \text{if } x = y \\ &= K^{-1} \left(1 - e^{-Kt} \right) \quad \text{if } x \neq y. \end{aligned}$$

This leads to another add- β smoother.

3.2 Graphs based on counts

A more interesting way of designing the word graph is through a similarity function which is based on the training set. For the normalized diffusion case, we propose the following

$$w(x, y) = \delta(|n(x) - n(y)| \leq 1). \quad (14)$$

That is, 2 words are ‘‘similar’’ if they have been seen a number of times which differs by at most one. The obtained estimator is denoted by π_{ND} . After some algebraic manipulations, we obtain

$$\pi_{ND}(y) = \frac{1}{n} \sum_{j=n(y)-1}^{n(y)+1} \frac{j r_j}{r_{j-1} + r_j + r_{j+1}}. \quad (15)$$

This estimator has a Good-Turing ‘‘flavor’’. For example, the total mass associated with the unseen words is

$$\sum_{y;n(y)=0} \pi_1(y) = \frac{1}{n} \frac{r_1}{1 + \frac{r_1}{r_0} + \frac{r_2}{r_0}}. \quad (16)$$

Note that the estimate of the unseen mass, in the case of the Good-Turing estimator, is equal to $n^{-1}r_1$, which is very close to the above when the vocabulary is large compared to the size of the training set (as is usually the case in practice).

Similarly, in the case of kernel diffusion, we choose $w \equiv 1$ and

$$x \sim y \iff |n(x) - n(y)| \leq 1 \quad (17)$$

The time t is chosen to be $|V|^{-1}$. The smoother cannot be computed in closed form. We used the formula (7) with $n = 3$ in the experiments. Larger values of n did not improve the results.

4 Experimental Results

In our experiments, we used Sections 00-22 (consisting of $\sim 10^6$ words) of the UPenn Treebank corpus for training, and Sections 23-24 (consisting of $\sim 10^5$ words) for testing. We split the training set into 10 subsets, leading to 10 datasets of size $\sim 10^5$ tokens each. The first of these sets was further split in subsets of size $\sim 10^4$ tokens each. Averaged results are presented in the tables below for various choices of the training set size. We show the mean code-length, as well as the standard deviation (when

	mean code length	std
$\pi_{\beta}, \beta = 1$	12.94	0.05
π_{GT}	11.40	0.08
π_{ND}	11.42	0.08
π_{KD}	11.51	0.08

Table 1: Results with training set of size $\sim 10^4$.

	mean code length	std
$\pi_{\beta}, \beta = 1$	11.10	0.03
π_{GT}	10.68	0.06
π_{ND}	10.69	0.06
π_{KD}	10.74	0.08

Table 2: Results with training set of size $\sim 10^5$.

available). In all cases, we chose $K = 10^5$ as the fixed size of our vocabulary.

The results show that π_{ND} , the estimate obtained with the Normalized Diffusion, is competitive with the Good-Turing π_{GT} . We performed a Kolmogorov-Smirnov test in order to determine if the code-lengths obtained with π_{ND} and π_{GT} in Table 1 differ significantly. The result is negative (P-value = .65), and the same holds for the larger training set in Table 2 (P-value=.95). On the other hand, π_{KD} (obtained with Kernel Diffusion) is not as efficient, but still better than add- β with $\beta = 1$.

5 Concluding Remarks

We showed that diffusions on graphs can be useful for language modeling. They yield naturally smooth estimates, and, under a particular choice of the “similarity” function between words, they are competitive with the Good-Turing estimator, which is considered to be the state-of-the-art in unigram language modeling. We plan to perform more exper-

	mean code length
$\pi_{\beta}, \beta = 1$	10.34
π_{GT}	10.30
π_{ND}	10.30
π_{KD}	10.31

Table 3: Results with training set of size $\sim 10^6$.

iments with other definitions of similarity between words. For example, we expect similarities based on co-occurrence in documents, or based on notions of semantic closeness (computed, for instance, using the WordNet hierarchy) to yield significant improvements over estimators which are only based on word counts.

References

- F. Chung. 1997. *Spectral Graph Theory*. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society.
- Thomas M. Cover and Joy A. Thomas. 1991. *Elements of Information Theory*. John Wiley & Sons, Inc.
- Risi Imre Kondor and John Lafferty. 2002. Diffusion kernels on graphs and other discrete input spaces. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 315–322.
- David McAllester and Robert E. Schapire. 2000. On the convergence rate of Good-Turing estimators. In *Proc. 13th Annu. Conference on Comput. Learning Theory*.
- Alon Orlitsky, Narayana P. Santhanam, and Junan Zhang. 2003. Always Good Turing: Asymptotically optimal probability estimation. In *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*.
- Arthur D. Szlam, Mauro Maggioni, and Ronald R. Coifman. 2006. A general framework for adaptive regularization based on diffusion processes on graphs. Technical report, YALE/DCS/TR1365.