Learning Bigrams from Unigrams

Xiaojin Zhu[†] and **Andrew B. Goldberg**[†] and **Michael Rabbat**[‡] and **Robert Nowak**[§]
[†]Department of Computer Sciences, University of Wisconsin-Madison
[‡]Department of Electrical and Computer Engineering, McGill University
[§]Department of Electrical and Computer Engineering, University of Wisconsin-Madison
{jerryzhu, goldberg}@cs.wisc.edu, michael.rabbat@mcgill.ca, nowak@ece.wisc.edu

Abstract

Traditional wisdom holds that once documents are turned into bag-of-words (unigram count) vectors, word orders are completely lost. We introduce an approach that, perhaps surprisingly, is able to learn a bigram language model from a set of bag-of-words documents. At its heart, our approach is an EM algorithm that seeks a model which maximizes the regularized marginal likelihood of the bag-of-words documents. In experiments on seven corpora, we observed that our learned bigram language models: i) achieve better test set perplexity than unigram models trained on the same bag-of-words documents, and are not far behind "oracle bigram models" trained on the corresponding ordered documents; ii) assign higher probabilities to sensible bigram word pairs; iii) improve the accuracy of ordered-document recovery from a bag-of-words. Our approach opens the door to novel phenomena, for example, privacy leakage from index files.

1 Introduction

A bag-of-words (BOW) is a basic document representation in natural language processing. In this paper, we consider a BOW in its simplest form, i.e., a *unigram count vector* or word histogram over the vocabulary. When performing the counting, word order is ignored. For example, the phrases "really neat" and "neat really" contribute equally to a BOW. Obviously, once a set of documents is turned into a set of BOWs, the word order information within them is completely lost—or is it?

In this paper, we show that one can in fact partly recover the order information. Specifically, *given a set of documents in unigram-count BOW representation, one can recover a non-trivial bigram language model (LM)*[1], which has part of the power of a bigram LM trained on ordered documents. At first glance this seems impossible: How can one learn bigram information from unigram counts? However, we will demonstrate that *multiple* BOW documents enable us to recover some higher-order information.

Our results have implications in a wide range of natural language problems, in particular document privacy. With the wide adoption of natural language applications like desktop search engines, software programs are increasingly indexing computer users' personal files for fast processing. Most index files include some variant of the BOW. As we demonstrate in this paper, if a malicious party gains access to BOW index files, it can recover more than just unigram frequencies: (i) the malicious party can recover a higher-order LM; (ii) with the LM it may attempt to recover the original ordered document from a BOW by finding the most-likely word permutation[2]. Future research will quantify the extent to which such a privacy breach is possible in theory, and will find solutions to prevent it.

There is a vast literature on language modeling; see, e.g., (Rosenfeld, 2000; Chen and Goodman, 1999; Brants et al., 2007; Roark et al., 2007). How-

[1]A trivial bigram LM is a unigram LM which ignores history: $P(v|u) = P(v)$.

[2]It is possible to use a generic higher-order LM, e.g., a trigram LM trained on standard English corpora, for this purpose. However, incorporating a user-specific LM helps.

Proceedings of ACL-08: HLT, pages 656–664,
Columbus, Ohio, USA, June 2008. ©2008 Association for Computational Linguistics

ever, to the best of our knowledge, none addresses this reverse direction of learning higher-order LMs from lower-order data. This work is inspired by recent advances in inferring network structure from co-occurrence data, for example, for computer networks and biological pathways (Rabbat et al., 2007).

2 Problem Formulation and Identifiability

We assume that a vocabulary of size W is given. For notational convenience, we include in the vocabulary a special "begin-of-document" symbol $\langle \mathrm{d} \rangle$ which appears only at the beginning of each document. The training corpus consists of a collection of n BOW documents $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Each BOW $\mathbf{x}_i$ is a vector $(x_{i1}, \ldots, x_{iW})$ where x_{iu} is the number of times word u occurs in document i. Our goal is to learn a bigram LM $\boldsymbol{\theta}$, represented as a $W \times W$ transition matrix with $\theta_{uv} = P(v|u)$, from the BOW corpus. Note $P(v|\langle \mathrm{d} \rangle)$ corresponds to the initial state probability for word v, and $P(\langle \mathrm{d} \rangle|u) = 0, \forall u$.

It is worth noting that traditionally one needs *ordered documents* to learn a bigram LM. A natural question that arises in our problem is whether or not a bigram LM can be recovered from the BOW corpus with any guarantee. Let $\mathcal{X}$ denote the space of all possible BOWs. As a toy example, consider $W = 3$ with the vocabulary $\{\langle \mathrm{d} \rangle, \mathrm{A}, \mathrm{B}\}$. Assuming all documents have equal length $|\mathbf{x}| = 4$ (including $\langle \mathrm{d} \rangle$), then $\mathcal{X} = \{(\langle \mathrm{d} \rangle$:1, A:3, B:0), ($\langle \mathrm{d} \rangle$:1, A:2, B:1), ($\langle \mathrm{d} \rangle$:1, A:1, B:2), ($\langle \mathrm{d} \rangle$:1, A:0, B:3)$\}$. Our training BOW corpus, when sufficiently large, provides the marginal distribution $\hat{p}(\mathbf{x})$ for $\mathbf{x} \in \mathcal{X}$. Can we recover a bigram LM from $\hat{p}(\mathbf{x})$?

To answer this question, we first need to introduce a generative model for the BOWs. We assume that the BOW corpus is generated from a bigram LM $\boldsymbol{\theta}$ in two steps: (i) An *ordered* document is generated from the bigram LM $\boldsymbol{\theta}$; (ii) The document's unigram counts are collected to produce the BOW $\mathbf{x}$. Therefore, the probability of a BOW $\mathbf{x}$ being generated by $\boldsymbol{\theta}$ can be computed by marginalizing over *unique orderings* $\mathbf{z}$ of $\mathbf{x}$:

$$P(\mathbf{x}|\boldsymbol{\theta}) = \sum_{\mathbf{z} \in \sigma(\mathbf{x})} P(\mathbf{z}|\boldsymbol{\theta}) = \sum_{\mathbf{z} \in \sigma(\mathbf{x})} \prod_{j=2}^{|\mathbf{x}|} \theta_{z_{j-1}, z_j},$$

where $\sigma(\mathbf{x})$ is the set of unique orderings, and $|\mathbf{x}|$ is the document length. For example, if $\mathbf{x}$ =($\langle \mathrm{d} \rangle$:1, A:2, B:1) then $\sigma(\mathbf{x}) = \{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3\}$ with $\mathbf{z}_1 =$ "$\langle \mathrm{d} \rangle$ A A B", $\mathbf{z}_2 =$ "$\langle \mathrm{d} \rangle$ A B A", $\mathbf{z}_3 =$ "$\langle \mathrm{d} \rangle$ B A A". Bigram LM recovery then amounts to finding a $\boldsymbol{\theta}$ that satisfies the system of marginal-matching equations

$$P(\mathbf{x}|\boldsymbol{\theta}) = \hat{p}(\mathbf{x}) , \ \forall \mathbf{x} \in \mathcal{X}. \qquad (1)$$

As a concrete example where *one can exactly recover a bigram LM from BOWs*, consider our toy example again. We know there are only three free variables in our 3×3 bigram LM $\boldsymbol{\theta}$: $r = \theta_{\langle \mathrm{d} \rangle A}, p = \theta_{AA}, q = \theta_{BB}$, since the rest are determined by normalization. Suppose the documents are generated from a bigram LM with true parameters $r = 0.25, p = 0.9, q = 0.5$. If our BOW corpus is very large, we will observe that 20.25% of the BOWs are ($\langle \mathrm{d} \rangle$:1, A:3, B:0), 37.25% are ($\langle \mathrm{d} \rangle$:1, A:2, B:1), and 18.75% are ($\langle \mathrm{d} \rangle$:1, A:0, B:3). These numbers are computed using the definition of $P(\mathbf{x}|\boldsymbol{\theta})$. We solve the reverse problem of finding r, p, q from the system of equations (1), now explicitly written as

$$\begin{cases} rp^2 = 0.2025 \\ rp(1-p) + r(1-p)(1-q) \\ \qquad +(1-r)(1-q)p = 0.3725 \\ (1-r)q^2 = 0.1875. \end{cases}$$

The above system has only one valid solution, which is the correct set of bigram LM parameters $(r, p, q) = (0.25, 0.9, 0.5)$.

However, if the true parameters were $(r, p, q) = (0.1, 0.2, 0.3)$ with proportions of BOWs being 0.4%, 19.8%, 8.1%, respectively, it is easy to verify that the system would have multiple valid solutions: $(0.1, 0.2, 0.3)$, $(0.8819, 0.0673, 0.8283)$, and $(0.1180, 0.1841, 0.3030)$. In general, if $\hat{p}(\mathbf{x})$ is known from the training BOW corpus, when can we *guarantee* to uniquely recover the bigram LM $\boldsymbol{\theta}$? This is the question of *identifiability*, which means the transition matrix $\boldsymbol{\theta}$ satisfying (1) exists and is unique. Identifiability is related to finding unique solutions of a system of polynomial equations since (1) is such a system in the elements of $\boldsymbol{\theta}$. The details are beyond the scope of this paper, but applying the technique in (Basu and Boston, 2000), it is possible to show that for $W = 3$ (including $\langle \mathrm{d} \rangle$) we need longer documents ($|\mathbf{x}| \geq 5$) to ensure identifiability. The identifiability of more general cases is still an open research question.

3 Bigram Recovery Algorithm

In practice, the documents are not truly generated from a bigram LM, and the BOW corpus may be small. We therefore seek a maximum likelihood estimate of $\boldsymbol{\theta}$ or a regularized version of it. Equivalently, we no longer require equality in (1), but instead find $\boldsymbol{\theta}$ that makes the distribution $P(\mathbf{x}|\boldsymbol{\theta})$ as close to $\hat{p}(\mathbf{x})$ as possible. We formalize this notion below.

3.1 The Objective Function

Given a BOW corpus $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, its normalized log likelihood under $\boldsymbol{\theta}$ is $\ell(\boldsymbol{\theta}) \equiv \frac{1}{C}\sum_{i=1}^n \log P(\mathbf{x}_i|\boldsymbol{\theta})$, where $C = \sum_{i=1}^n (|\mathbf{x}_i| - 1)$ is the corpus length excluding $\langle \mathrm{d} \rangle$'s. The idea is to find $\boldsymbol{\theta}$ that maximizes $\ell(\boldsymbol{\theta})$. This also brings $P(\mathbf{x}|\boldsymbol{\theta})$ closest to $\hat{p}(\mathbf{x})$ in the KL-divergence sense. However, to prevent overfitting, we regularize the problem so that $\boldsymbol{\theta}$ prefers to be close to a "prior" bigram LM $\boldsymbol{\phi}$. The prior $\boldsymbol{\phi}$ is also estimated from the BOW corpus, and is discussed in Section 3.4. We define the regularizer to be an asymmetric dissimilarity $\mathcal{D}(\boldsymbol{\phi}, \boldsymbol{\theta})$ between the prior $\boldsymbol{\phi}$ and the learned model $\boldsymbol{\theta}$. The dissimilarity is 0 if $\boldsymbol{\theta} = \boldsymbol{\phi}$, and increases as they diverge. Specifically, the KL-divergence between two word distributions conditioned on the same history u is $KL(\boldsymbol{\phi}_{u\cdot}\|\boldsymbol{\theta}_{u\cdot}) = \sum_{v=1}^W \phi_{uv} \log \frac{\phi_{uv}}{\theta_{uv}}$. We define $\mathcal{D}(\boldsymbol{\phi}, \boldsymbol{\theta})$ to be the average KL-divergence over all histories: $\mathcal{D}(\boldsymbol{\phi}, \boldsymbol{\theta}) \equiv \frac{1}{W}\sum_{u=1}^W KL(\boldsymbol{\phi}_{u\cdot}\|\boldsymbol{\theta}_{u\cdot})$, which is convex in $\boldsymbol{\theta}$ (Cover and Thomas, 1991). We will use the following derivative later: $\partial\mathcal{D}(\boldsymbol{\phi}, \boldsymbol{\theta})/\partial\theta_{uv} = -\phi_{uv}/(W\theta_{uv})$.

We are now ready to define the regularized optimization problem for recovering a bigram LM θ from the BOW corpus:

$$\begin{aligned} \max_{\boldsymbol{\theta}} \quad & \ell(\boldsymbol{\theta}) - \lambda\mathcal{D}(\boldsymbol{\phi}, \boldsymbol{\theta}) \\ \text{subject to} \quad & \boldsymbol{\theta}\mathbf{1} = \mathbf{1}, \quad \boldsymbol{\theta} \geq 0. \end{aligned} \tag{2}$$

The weight λ controls the strength of the prior. The constraints ensure that $\boldsymbol{\theta}$ is a valid bigram matrix, where $\mathbf{1}$ is an all-one vector, and the non-negativity constraint is element-wise. Equivalently, (2) can be viewed as the *maximum a posteriori* (MAP) estimate of $\boldsymbol{\theta}$, with independent Dirichlet priors for each row of $\boldsymbol{\theta}$: $p(\boldsymbol{\theta}_{u\cdot}) = \mathrm{Dir}(\boldsymbol{\theta}_{u\cdot}|\alpha_{u\cdot})$ and hyperparameters $\alpha_{uv} = \frac{\lambda C}{W}\phi_{uv} + 1$.

The summation over hidden ordered documents $\mathbf{z}$ in $P(\mathbf{x}|\boldsymbol{\theta})$ couples the variables and makes (2) a non-concave problem. We optimize $\boldsymbol{\theta}$ using an EM algorithm.

3.2 The EM Algorithm

We derive the EM algorithm for the optimization problem (2). Let $\mathcal{O}(\boldsymbol{\theta}) \equiv \ell(\boldsymbol{\theta}) - \lambda\mathcal{D}(\boldsymbol{\phi}, \boldsymbol{\theta})$ be the objective function. Let $\boldsymbol{\theta}^{(t-1)}$ be the bigram LM at iteration $t-1$. We can lower-bound $\mathcal{O}$ as follows:

$$\begin{aligned} & \mathcal{O}(\boldsymbol{\theta}) \\ = \; & \frac{1}{C}\sum_{i=1}^n \log \sum_{\mathbf{z}\in\sigma(\mathbf{x}_i)} P(\mathbf{z}|\boldsymbol{\theta}^{(t-1)}, \mathbf{x}) \frac{P(\mathbf{z}|\boldsymbol{\theta})}{P(\mathbf{z}|\boldsymbol{\theta}^{(t-1)}, \mathbf{x})} \\ & -\lambda\mathcal{D}(\boldsymbol{\phi}, \boldsymbol{\theta}) \\ \geq \; & \frac{1}{C}\sum_{i=1}^n \sum_{\mathbf{z}\in\sigma(\mathbf{x}_i)} P(\mathbf{z}|\boldsymbol{\theta}^{(t-1)}, \mathbf{x}) \log \frac{P(\mathbf{z}|\boldsymbol{\theta})}{P(\mathbf{z}|\boldsymbol{\theta}^{(t-1)}, \mathbf{x})} \\ & -\lambda\mathcal{D}(\boldsymbol{\phi}, \boldsymbol{\theta}) \\ \equiv \; & \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t-1)}). \end{aligned}$$

We used Jensen's inequality above since $\log()$ is concave. The lower bound $\mathcal{L}$ involves $P(\mathbf{z}|\boldsymbol{\theta}^{(t-1)}, \mathbf{x})$, the probability of hidden orderings of the BOW under the previous iteration's model. In the E-step of EM we compute $P(\mathbf{z}|\boldsymbol{\theta}^{(t-1)}, \mathbf{x})$, which will be discussed in Section 3.3. One can verify that $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t-1)})$ is *concave* in $\boldsymbol{\theta}$, unlike the original objective $\mathcal{O}(\boldsymbol{\theta})$. In addition, the lower bound "touches" the objective at $\boldsymbol{\theta}^{(t-1)}$, i.e., $\mathcal{L}(\boldsymbol{\theta}^{(t-1)}, \boldsymbol{\theta}^{(t-1)}) = \mathcal{O}(\boldsymbol{\theta}^{(t-1)})$.

The EM algorithm iteratively maximizes the lower bound, which is now a concave optimization problem: $\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t-1)})$, subject to $\boldsymbol{\theta}\mathbf{1} = \mathbf{1}$. The non-negativity constraints turn out to be automatically satisfied. Introducing Lagrange multipliers β_u for each history $u = 1 \ldots W$, we form the Lagrangian Δ:

$$\Delta \equiv \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t-1)}) - \sum_{u=1}^W \beta_u \left(\sum_{v=1}^W \theta_{uv} - 1\right).$$

Taking the partial derivative with respect to θ_{uv} and setting it to zero: $\partial\Delta/\partial\theta_{uv} = 0$, we arrive at the following update:

$$\theta_{uv} \propto \sum_{i=1}^n \sum_{\mathbf{z}\in\sigma(\mathbf{x}_i)} P(\mathbf{z}|\boldsymbol{\theta}^{(t-1)}, \mathbf{x}) c_{uv}(\mathbf{z}) + \frac{\lambda C}{W}\phi_{uv}. \tag{3}$$

```
Input: BOW documents ${\mathbf{x}_1, \ldots, \mathbf{x}_n}$, a prior bigram LM $\boldsymbol{\phi}$, weight $\lambda$.
  1. $t = 1$. Initialize $\boldsymbol{\theta}^{(0)} = \boldsymbol{\phi}$.
  2. Repeat until the objective $\mathcal{O}(\boldsymbol{\theta})$ converges:
      (a) (E-step) Compute $P(\mathbf{z}|\boldsymbol{\theta}^{(t-1)}, \mathbf{x})$ for $\mathbf{z} \in \sigma(\mathbf{x}_i), i = 1, \ldots, n$.
      (b) (M-step) Compute $\boldsymbol{\theta}^{(t)}$ using (3). Let $t = t + 1$.
Output: The recovered bigram LM $\boldsymbol{\theta}$.
```

Table 1: The EM algorithm

The normalization is over $v = 1 \ldots W$. We use $c_{uv}(\mathbf{z})$ to denote the number of times the bigram "uv" appears in the ordered document $\mathbf{z}$. This is the M-step of EM. Intuitively, the first term counts how often the bigram "uv" occurs, weighing each ordering by its probability under the previous model; the second term pulls the parameter towards the prior. If the weight of the prior $\lambda \to \infty$, we would have $\theta_{uv} = \phi_{uv}$. The update is related to the MAP estimate for a multinomial distribution with a Dirichlet prior, where we use the expected counts.

We initialize the EM algorithm with $\boldsymbol{\theta}^{(0)} = \boldsymbol{\phi}$. The EM algorithm is summarized in Table 1.

3.3 Approximate E-step

The E-step needs to compute the expected bigram counts of the form

$$\sum_{\mathbf{z} \in \sigma(\mathbf{x})} P(\mathbf{z}|\boldsymbol{\theta}, \mathbf{x}) c_{uv}(\mathbf{z}). \tag{4}$$

However, this poses a computational problem. The summation is over unique ordered documents. The number of unique ordered documents can be on the order of $|\mathbf{x}|!$, i.e., all permutations of the BOW. For a short document of length 15, this number is already 10^{12}. Clearly, brute-force enumeration is only feasible for very short documents. Approximation is necessary to handle longer ones.

A simple Monte Carlo approximation to (4) would involve sampling ordered documents $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_L$ according to $\mathbf{z}_i \sim P(\mathbf{z}|\boldsymbol{\theta}, \mathbf{x})$, and replacing (4) with $\sum_{i=1}^{L} c_{uv}(\mathbf{z}_i)/L$. This estimate is unbiased, and the variance decreases linearly with the number of samples, L. However, sampling directly from P is difficult.

Instead, we sample ordered documents $\mathbf{z}_i \sim R(\mathbf{z}_i|\boldsymbol{\theta}, \mathbf{x})$ from a distribution R which is easy to generate, and construct an approximation using *importance sampling* (see, e.g., (Liu, 2001)). With each sample, $\mathbf{z}_i$, we associate a weight $w_i \propto P(\mathbf{z}_i|\theta, \mathbf{x})/R(\mathbf{z}_i|\theta, \mathbf{x})$. The importance sampling approximation to (4) is then given by $(\sum_{i=1}^{L} w_i c_{uv}(z_i))/(\sum_{i=1}^{L} w_i)$. Re-weighting the samples in this fashion accounts for the fact that we are using a sampling distribution R which is different the target distribution P, and guarantees that our approximation is asymptotically unbiased.

The quality of an importance sampling approximation is closely related to how closely R resembles P; the more similar they are, the better the approximation, in general. Given a BOW $\mathbf{x}$ and our current bigram model estimate, $\boldsymbol{\theta}$, we generate one sample (an ordered document $\mathbf{z}_i$) by sequentially drawing words from the bag, with probabilities proportional to $\boldsymbol{\theta}$, but properly normalized to form a distribution based on which words remain in the bag. For example, suppose $\mathbf{x} = (\langle\text{d}\rangle\text{:1, A:2, B:1, C:1})$. Then we set $z_{i1} = \langle\text{d}\rangle$, and sample $z_{i2} = \text{A}$ with probability $2\theta_{\langle\text{d}\rangle\text{A}}/(2\theta_{\langle\text{d}\rangle\text{A}} + \theta_{\langle\text{d}\rangle\text{B}} + \theta_{\langle\text{d}\rangle\text{C}})$. Similarly, if $z_{i(j-1)} = u$ and if v is in the original BOW that hasn't been sampled yet, then we set the next word in the ordered document z_{ij} equal to v with probability proportional to $c_v \theta_{uv}$, where c_v is the count of v in the remaining BOW. For this scheme, one can verify (Rabbat et al., 2007) that the importance weight corresponding to a sampled ordered document $\mathbf{z}_i = (z_{i1}, \ldots, z_{i|\mathbf{x}|})$ is given by $w_i = \prod_{t=2}^{|x|} \sum_{i=t}^{|x|} \theta_{z_{t-1} z_i}$. In our implementation, the number of importance samples used for a document $\mathbf{x}$ is $10|\mathbf{x}|^2$ if the length of the document $|\mathbf{x}| > 8$; otherwise we enumerate $\sigma(\mathbf{x})$ without importance sampling.

3.4 Prior Bigram LM $\boldsymbol{\phi}$

The quality of the EM solution $\boldsymbol{\theta}$ can depend on the prior bigram LM $\boldsymbol{\phi}$. To assess bigram recoverability from a BOW corpus alone, we consider only priors estimated from the corpus itself[3]. Like $\boldsymbol{\theta}$, $\boldsymbol{\phi}$ is a $W \times W$ transition matrix with $\phi_{uv} = P(v|u)$. When

[3]Priors based on general English text or domain-specific knowledge could be used in specific applications.

appropriate, we set the initial probability $\phi_{\langle \mathrm{d} \rangle v}$ proportional to the number of times word v appears in the BOW corpus. We consider three prior models:

Prior 1: Unigram $\boldsymbol{\phi}^{unigram}$. The most naïve $\boldsymbol{\phi}$ is a unigram LM which ignores word history. The probability for word v is estimated from the BOW corpus frequency of v, with add-1 smoothing: $\phi_{uv}^{unigram} \propto 1 + \sum_{i=1}^{n} x_{iv}$. We should point out that the unigram prior is an asymmetric bigram, i.e., $\phi_{uv}^{unigram} \neq \phi_{vu}^{unigram}$.

Prior 2: Frequency of Document Co-occurrence (FDC) $\boldsymbol{\phi}^{fdc}$. Let $\delta(u, v|\mathbf{x}) = 1$ if words $u \neq v$ co-occur (regardless of their counts) in BOW $\mathbf{x}$, and 0 otherwise. In the case $u = v$, $\delta(u, u|\mathbf{x}) = 1$ only if u appears at least twice in $\mathbf{x}$. Let $c_{uv}^{fdc} = \sum_{i=1}^{n} \delta(u, v|\mathbf{x}_i)$ be the number of BOWs in which u, v co-occur. The FDC prior is $\boldsymbol{\phi}_{uv}^{fdc} \propto c_{uv}^{fdc} + 1$. The co-occurrence counts c^{fdc} are symmetric, but $\boldsymbol{\phi}^{fdc}$ is asymmetric because of normalization. FDC captures some notion of potential transitions from u to v. FDC is in spirit similar to Kneser-Ney smoothing (Kneser and Ney, 1995) and other methods that accumulate indicators of document membership.

Prior 3: Permutation-Based (Perm) $\boldsymbol{\phi}^{perm}$. Recall that $c_{uv}(\mathbf{z})$ is the number of times the bigram "uv" appears in an ordered document $\mathbf{z}$. We define $c_{uv}^{perm} = \sum_{i=1}^{n} \mathrm{E}_{\mathbf{z} \in \sigma(\mathbf{x}_i)}[c_{uv}(\mathbf{z})]$, where the expectation is with respect to all unique orderings of each BOW. We make the zero-knowledge assumption of uniform probability over these orderings, rather than $P(\mathbf{z}|\boldsymbol{\theta})$ as in the EM algorithm described above. EM will refine these estimates, though, so this is a natural starting point. Space precludes a full discussion, but it can be proven that $c_{uv}^{perm} = \sum_{i=1}^{n} x_{iu} x_{iv} / |\mathbf{x}_i|$ if $u \neq v$, and $c_{uu}^{perm} = \sum_{i=1}^{n} x_{iu}(x_{iu} - 1)/|\mathbf{x}_i|$. Finally, $\phi_{uv}^{perm} \propto c_{uv}^{perm} + 1$.

3.5 Decoding Ordered Documents from BOWs

Given a BOW $\mathbf{x}$ and a bigram LM $\boldsymbol{\theta}$, we formulate document recovery as the problem $\mathbf{z}^* = \mathrm{argmax}_{\mathbf{z} \in \sigma(\mathbf{x})} P(\mathbf{z}|\boldsymbol{\theta})$. In fact, we can generate the top N candidate ordered documents in terms of $P(\mathbf{z}|\boldsymbol{\theta})$. We use A* search to construct such an N-best list (Russell and Norvig, 2003). Each state is an ordered, partial document. Its successor states append one more unused word in $\mathbf{x}$ to the partial document. The actual cost g from the start (empty document) to a state is the log probability of the partial document under bigram $\boldsymbol{\theta}$. We design a heuristic cost h from the state to the goal (complete document) that is *admissible*: the idea is to over-use the best bigram history for the remaining words in $\mathbf{x}$. Let the partial document end with word w_e. Let the count vector for the remaining BOW be $(c_1, \ldots, c_W)$. One admissible heuristic is $h = \log \prod_{u=1}^{W} P(u|bh(u); \boldsymbol{\theta})^{c_u}$, where the "best history" for word type u is $bh(u) = \mathrm{argmax}_v \theta_{vu}$, and v ranges over the word types with non-zero counts in $(c_1, \ldots, c_W)$, plus w_e. It is easy to see that h is an upper bound on the bigram log probability that the remaining words in $\mathbf{x}$ can achieve.

We use a memory-bounded A* search similar to (Russell, 1992), because long BOWs would otherwise quickly exhaust memory. When the priority queue grows larger than the bound, the worst states (in terms of $g + h$) in the queue are purged. This necessitates a double-ended priority queue that can pop either the maximum or minimum item. We use an efficient implementation with Splay trees (Chong and Sahni, 2000). We continue running A* after popping the goal state from its priority queue. Repeating this N times gives the N-best list.

4 Experiments

We show experimentally that the proposed algorithm is indeed able to recover reasonable bigram LMs from BOW corpora. We observe:

1. Good test set perplexity: Using test (held-out) set perplexity (PP) as an objective measure of LM quality, we demonstrate that our recovered bigram LMs are much better than naïve unigram LMs trained on the same BOW corpus. Furthermore, they are not far behind the "oracle" bigram LMs trained on *ordered documents* that correspond to the BOWs.

2. Sensible bigram pairs: We inspect the recovered bigram LMs and find that they assign higher probabilities to sensible bigram pairs (e.g., "i mean", "oh boy", "that's funny"), and lower probabilities to nonsense pairs (e.g., "i yep", "you let's", "right lot").

3. Document recovery from BOW: With the bigram LMs, we show improved accuracy in recovering ordered documents from BOWs.

We describe these experiments in detail below.

Corpus	$\|V\|$	# Docs	# Tokens	$\overline{\|\mathbf{x}\|}$
SV10	10	6775	7792	1.2
SV25	25	9778	13324	1.4
SV50	50	12442	20914	1.7
SV100	100	14602	28611	2.0
SV250	250	18933	51950	2.7
SV500	500	23669	89413	3.8
SumTime	882	3341	68815	20.6

Table 2: Corpora statistics: vocabulary size, document count, total token count, and mean document length.

4.1 Corpora and Protocols

We note that although in principle our algorithm works on large corpora, the current implementation does not scale well (Table 3 last column). We therefore experimented on seven corpora with relatively small vocabulary sizes, and with short documents (mostly one sentence per document). Table 2 lists statistics describing the corpora. The first six contain text transcripts of conversational telephone speech from the small vocabulary "SVitchboard 1" data set. King et al. constructed each corpus from the full Switchboard corpus, with the restriction that the sentences use only words in the corresponding vocabulary (King et al., 2005). We refer to these corpora as SV10, SV25, SV50, SV100, SV250, and SV500. The seventh corpus comes from the SumTime-Meteo data set (Sripada et al., 2003), which contains real weather forecasts for offshore oil rigs in the North Sea. For the SumTime corpus, we performed sentence segmentation to produce documents, removed punctuation, and replaced numeric digits with a special token.

For each of the seven corpora, we perform 5-fold cross validation. We use four folds other than the k-th fold as the training set to train (recover) bigram LMs, and the k-th fold as the test set for evaluation. This is repeated for $k = 1 \dots 5$, and we report the average cross validation results. We distinguish the original *ordered documents* (training set $\mathbf{z}_1, \dots \mathbf{z}_n$, test set $\mathbf{z}_{n+1}, \dots, \mathbf{z}_m$) and the corresponding BOWs (training set $\mathbf{x}_1 \dots \mathbf{x}_n$, test set $\mathbf{x}_{n+1} \dots \mathbf{x}_m$). In all experiments, we simply set the weight $\lambda = 1$ in (2). Given a training set and a test set, we perform the following steps:

1. Build prior LMs $\boldsymbol{\phi}^X$ from the training BOW corpus $\mathbf{x}_1, \dots \mathbf{x}_n$, for $X = unigram, fdc, perm$.

2. Recover the bigram LMs $\boldsymbol{\theta}^X$ with the EM algorithm in Table 1, from the training BOW corpus $\mathbf{x}_1, \dots \mathbf{x}_n$ and using the prior from step 1.

3. Compute the MAP bigram LM from the *ordered* training documents $\mathbf{z}_1, \dots \mathbf{z}_n$. We call this the "oracle" bigram LM because it uses order information (not available to our algorithm), and we use it as a lower-bound on perplexity.

4. Test all LMs on $\mathbf{z}_{n+1}, \dots, \mathbf{z}_m$ by perplexity.

4.2 Good Test Set Perplexity

Table 3 reports the 5-fold cross validation mean-test-set-PP values for all corpora, and the run time per EM iteration. Because of the long running time, we adopt the rule-of-thumb stopping criterion of "two EM iterations". First, we observe that all bigram LMs perform better than unigram LMs $\boldsymbol{\phi}^{unigram}$ even though they are trained on the same BOW corpus. Second, all recovered bigram LMs $\boldsymbol{\theta}^X$ improved upon their corresponding baselines $\boldsymbol{\phi}^X$. The difference across *every* row is statistically significant according to a two-tailed paired t-test with $p < 0.05$. The differences among PP($\boldsymbol{\theta}^X$) for the same corpus are also significant (except between $\boldsymbol{\theta}^{unigram}$ and $\boldsymbol{\theta}^{perm}$ for SV500). Finally, we observe that $\boldsymbol{\theta}^{perm}$ tends to be best for the smaller vocabulary corpora, whereas $\boldsymbol{\theta}^{fdc}$ dominates as the vocabulary grows.

To see how much better we could do if we had *ordered* training documents $\mathbf{z}_1, \dots, \mathbf{z}_n$, we present the mean-test-set-PP of "oracle" bigram LMs in Table 4. We used three smoothing methods to obtain oracle LMs: absolute discounting using a constant of 0.5 (we experimented with other values, but 0.5 worked best), Good-Turing, and interpolated Witten-Bell as implemented in the SRILM toolkit (Stolcke, 2002). We see that our recovered LMs (trained on *unordered* BOW documents), especially for small vocabulary corpora, are close to the oracles (trained on *ordered* documents). For the larger datasets, the recovery task is more difficult, and the gap between the oracle LMs and the $\boldsymbol{\theta}$ LMs widens. Note that the oracle LMs do much better than the recovered LMs on the SumTime corpus; we suspect the difference is due to the larger vocabulary and significantly higher average sentence length (see Table 2).

4.3 Sensible Bigram Pairs

The next set of experiments compares the recovered bigram LMs to their corresponding prior LMs

Corpus	X	PP(ϕ^X)	PP(θ^X)	Time/ Iter
SV10	unigram	7.48	6.95	< 1s
	fdc	6.52	6.47	< 1s
	perm	6.50	6.45	< 1s
SV25	unigram	16.4	12.8	0.1s
	fdc	12.3	11.8	0.1s
	perm	12.2	11.7	0.1s
SV50	unigram	29.1	19.7	2s
	fdc	19.6	17.8	4s
	perm	19.5	17.7	5s
SV100	unigram	45.4	27.8	7s
	fdc	29.5	25.3	11s
	perm	30.0	25.6	11s
SV250	unigram	91.8	51.2	5m
	fdc	60.0	47.3	8m
	perm	65.4	49.7	8m
SV500	unigram	149.1	87.2	3h
	fdc	104.8	80.1	3h
	perm	123.9	87.4	3h
SumTime	unigram	129.7	81.8	4h
	fdc	103.2	77.7	4h
	perm	187.9	85.4	3h

Table 3: Mean test set perplexities of prior LMs and bigram LMs recovered after 2 EM iterations.

in terms of how they assign probabilities to word pairs. One naturally expects probabilities for frequently occurring bigrams to increase, while rare or nonsensical bigrams' probabilities should decrease. For a prior-bigram pair (ϕ, θ), we evaluate the change in probabilities by computing the ratio $\rho_{hw} = \frac{P(w|h,\theta)}{P(w|h,\phi)} = \frac{\theta_{hw}}{\phi_{hw}}$. For a given history h, we sort words w by this ratio rather than by actual bigram probability because the bigrams with the highest and lowest probabilities tend to stay the same, while the changes accounting for differences in PP scores are more noticeable by considering the ratio.

Due to space limitation, we present one specific result (FDC prior, fold 1) for the SV500 corpus in Table 5. Other results are similar. The table lists a few most frequent unigrams as history words h (left), and the words w with the smallest (center) and largest (right) ρ_{hw} ratio. Overall we see that our EM algorithm is forcing meaningless bigrams (e.g., "i goodness", "oh thing") to have lower probabilities, while assigning higher probabilities to sensible bigram pairs (e.g., "really good", "that's funny"). Note that the reverse of some common expressions (e.g., "right that's") also rise in probability, suggesting the algorithm detects that the two words are often adjacent, but lacks sufficient information to nail down the exact order.

Corpus	Absolute Discount	Good-Turing	Witten-Bell	θ^*
SV10	6.27	6.28	6.27	6.45
SV25	10.5	10.6	10.5	11.7
SV50	14.8	14.9	14.8	17.7
SV100	20.0	20.1	20.0	25.3
SV250	33.7	33.7	33.8	47.3
SV500	50.9	50.9	51.3	80.1
SumTime	10.8	10.5	10.6	77.7

Table 4: Mean test set perplexities for oracle bigram LMs trained on $\mathbf{z}_1, \ldots, \mathbf{z}_n$ and tested on $\mathbf{z}_{n+1}, \ldots, \mathbf{z}_m$. For reference, the rightmost column lists the best result using a recovered bigram LM (θ^{perm} for the first three corpora, θ^{fdc} for the latter four).

4.4 Document Recovery from BOW

We now play the role of the malicious party mentioned in the introduction. We show that, compared to their corresponding prior LMs, our recovered bigram LMs are better able to reconstruct ordered documents out of test BOWs $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_m$. We perform document recovery using 1-best A* decoding. We use "document accuracy" and "n-gram accuracy" (for $n = 2, 3$) as our evaluation criteria. We define document accuracy (Acc^{doc}) as the fraction of documents[4] for which the decoded document matches the true ordered document exactly. Similarly, n-gram accuracy (Acc^n) measures the fraction of all n-grams in test documents (with n or more words) that are recovered correctly.

For this evaluation, we compare models built for the SV500 corpus. Table 6 presents 5-fold cross validation average test-set accuracies. For each accuracy measure, we compare the prior LM with the recovered bigram LM. It is interesting to note that the FDC and Perm priors reconstruct documents surprisingly well, but we can always improve them by running our EM algorithm. The accuracies obtained by θ are statistically significantly better (via two-tailed paired t-tests with $p < 0.05$) than their corresponding priors ϕ in all cases except Acc^{doc} for θ^{perm} versus ϕ^{perm}. Furthermore, θ^{fdc} and θ^{perm} are significantly better than all other models in terms of all three reconstruction accuracy measures.

[4]We omit single-word documents from these computations.

h	w (smallest ρ_{hw})	w (largest ρ_{hw})
i	yep, bye-bye, ah, goodness, ahead	mean, guess, think, bet, agree
you	let's, us, fact, such, deal	thank, bet, know, can, do
right	as, lot, going, years, were	that's, all, right, now, you're
oh	thing, here, could, were, doing	boy, really, absolutely, gosh, great
that's	talking, home, haven't, than, care	funny, wonderful, true, interesting, amazing
really	now, more, yep, work, you're	sad, neat, not, good, it's

Table 5: The recovered bigram LM $\boldsymbol{\theta}^{fdc}$ decreases nonsense bigram probabilities (center column) and increases sensible ones (right column) compared to the prior $\boldsymbol{\phi}^{fdc}$ on the SV500 corpus.

$\boldsymbol{\phi}^{perm}$ reconstructions of test BOWs	$\boldsymbol{\theta}^{perm}$ reconstructions of test BOWs
just it's it's it's just going	**it's just it's just it's going**
it's probably out there else something	**it's probably something else out there**
the the have but it doesn't	**but it doesn't have the the**
you to talking nice was it yes	**yes it was nice talking to you**
that's well that's what i'm saying	**well that's that's what i'm saying**
a little more here home take	**a little more take home here**
and they can very be nice too	**and they can be very nice too**
i think well that's great i'm	**well i think that's great i'm**
but was he because only always	**but only because he was always**
that's think i don't i no	**no i don't i think that's**
that in and it it's interesting	**and it it's interesting that in**
that's right that's right that's difficult	right that's that's right that's difficult
so just not quite a year	so just not a quite year
well it is a big dog	well it is big a dog
so do you have a car	so you do have a car

Table 7: Subset of SV500 documents that only $\boldsymbol{\phi}^{perm}$ or $\boldsymbol{\theta}^{perm}$ (but not both) reconstructs correctly. The correct reconstructions are in bold.

	Acc^{doc}		Acc^2		Acc^3	
X	$\boldsymbol{\phi}^X$	$\boldsymbol{\theta}^X$	$\boldsymbol{\phi}^X$	$\boldsymbol{\theta}^X$	$\boldsymbol{\phi}^X$	$\boldsymbol{\theta}^X$
unigram	11.1	26.8	17.7	32.8	2.7	11.8
fdc	30.2	31.0	33.0	35.1	11.4	13.3
perm	30.9	31.5	32.7	34.8	11.5	13.1

Table 6: Percentage of correctly reconstructed documents, 2-grams and 3-grams from test BOWs in SV500, 5-fold cross validation. The same trends continue for 4-grams and 5-grams (not shown).

We conclude our experiments with a closer look at some BOWs for which $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ reconstruct differently. As a representative example, we compare $\boldsymbol{\theta}^{perm}$ to $\boldsymbol{\phi}^{perm}$ on one test set of the SV500 corpus. There are 92 documents that are correctly reconstructed by $\boldsymbol{\theta}^{perm}$ but not by $\boldsymbol{\phi}^{perm}$. In contrast, only 65 documents are accurately reordered by $\boldsymbol{\phi}^{perm}$ but not by $\boldsymbol{\theta}^{perm}$. Table 7 presents a subset of these documents with six or more words. Overall, we conclude that the recovered bigram LMs do a better job at reconstructing BOW documents.

5 Conclusions and Future Work

We presented an algorithm that learns bigram language models from BOWs. We plan to: i) investigate ways to speed up our algorithm; ii) extend it to trigram and higher-order models; iii) handle the mixture of BOW documents and some ordered documents (or phrases) when available; iv) adapt a general English LM to a special domain using only BOWs from that domain; and v) explore novel applications of our algorithm.

Acknowledgments

We thank Ben Liblit for tips on doubled-ended priority queues, and the anonymous reviewers for valuable comments. This work is supported in part by the Wisconsin Alumni Research Foundation, NSF CCF-0353079 and CCF-0728767, and the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

Samit Basu and Nigel Boston. 2000. Identifiability of polynomial systems. Technical report, University of Illinois at Urbana-Champaign.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Stanley F. Chen and Joshua T. Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13(4):359–393.

Kyun-Rak Chong and Sartaj Sahni. 2000. Correspondence-based data structures for double-ended priority queues. *The ACM Journal of Experimental Algorithmics*, 5(2).

Thomas M. Cover and Joy A. Thomas. 1991. *Elements of Information Theory*. John Wiley & Sons, Inc.

Simon King, Chris Bartels, and Jeff Bilmes. 2005. SVitchboard 1: Small vocabulary tasks from Switchboard 1. In *Interspeech 2005*, Lisbon, Portugal.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for M-gram language modeling. In *ICASSP*.

Jun S. Liu. 2001. *Monte Carlo Strategies in Scientific Computing*. Springer.

Michael Rabbat, Mário Figueiredo, and Robert Nowak. 2007. Inferring network structure from co-occurrences. In *Advances in Neural Information Processing Systems (NIPS) 20*.

Brian Roark, Murat Saraclar, and Michael Collins. 2007. Discriminative n-gram language modeling. *Computer Speech and Language*, 21(2):373–392.

Ronald Rosenfeld. 2000. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8).

Stuart Russell and Peter Norvig. 2003. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, second edition.

Stuart Russell. 1992. Efficient memory-bounded search methods. In *The 10th European Conference on Artificial Intelligence*.

Somayajulu G. Sripada, Ehud Reiter, Jim Hunter, and Jin Yu. 2003. Exploiting a parallel TEXT-DATA corpus. In *Proceedings of Corpus Linguistics*, pages 734–743, Lancaster, U.K.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of International Conference on Spoken Language Processing*, Denver, Colorado.