Document Recovery from Bag-of-Word Indices

Nathanael Fillmore, Andrew B. Goldberg, Xiaojin Zhu
Department of Computer Sciences
University of Wisconsin-Madison
{nathanae, goldberg, jerryzhu}@cs.wisc.edu

Abstract

Motivated by computer privacy issues, we present the novel problem of document recovery from an index: given only a document's bag-of-words (BOW) vector or other type of index, reconstruct the original *ordered* document. We investigate a variety of index types, including count-based BOW vectors, stopwords-removed count BOW vectors, indicator BOW vectors, and bigram count vectors. We formulate the problem as hypothesis rescoring using A* search with the Google Web 1T 5-gram corpus. Our experiments on five domains indicate that if original documents are short, the documents can be recovered with high accuracy.

1 Introduction

With the proliferation of text processing software such as desktop search engines, database systems, and spam filters, computer users' personal files are increasingly being indexed to enable fast searches and processing. With great popularity comes greater risk, though: few users know where the index files are maintained and what security measures are used to protect them. In this paper, we seek to answer the following question: if an adversary obtains a document's index, what can it learn about the original document?

We define the problem of *document recovery from an index* as follows: given only a document's bag-of-words (BOW) vector or other index, reconstruct the original *ordered* document. Our goal is to understand the extent to which recovery is possible under standard computational means using large-scale language resources (i.e., the Google Web 1T 5-gram corpus).

The risk to privacy is real. An adversary may hack into a machine and gain access to the indices.[1] Software with such indices include desktop search engines such as Apache Lucene (lucene.apache.org), some e-mail clients such as *sup* (sup.rubyforge.org) and *cone* (www.courier-mta.org/cone), and database management systems like MySQL and PostgreSQL. Even more alarmingly, certain types of text classifiers (e.g., nearest neighbor and kernel machines) willingly distribute indices with the trained model to end users (and potentially the adversary). In this case, the model consists of the prototype training examples or the support vectors – documents in index form. The concern is that the training documents might be confidential or private in nature, and people should not assume the model is secure. In fact, the model files produced by both SVM^{light} (svmlight.joachims.org) and LIBSVM (www.csie.ntu.edu.tw/~cjlin/libsvm) contain the support vectors in a readable ASCII format (even in the case of linear SVMs). Thus, anyone provided with a model file has access to a set of index files.

The approach adopted in this paper is similar to hypothesis rescoring, which has a long history in NLP. Nonetheless, this work provides several novel contributions: (1) A formulation of the problem of document recovery, which we believe is an important area for future NLP research. (2) A benchmark dataset that simulates domains where document recovery could be particularly detrimental

[1] Throughout this paper, we assume that encryption has been compromised by the adversary.

(e.g., personal e-mail, government documents, medical records). (3) A comprehensive empirical study using the Google Web 1T 5-gram corpus and the resulting "stupid backoff" language model (Brants et al., 2007). (4) A specialized A* decoder using novel search heuristics for document recovery from several types of indices.

2 Related work

In the NLP domain, there has been a great deal of research on hypothesis rescoring. In statistical machine translation (SMT), various decoding strategies have been explored (Knight, 1999; Germann, 2003), including traveling salesman formulation (Germann et al., 2001), beam search (Hoang and Koehn, 2008; Crego et al., 2005), and A* search (Och et al., 2001). Automatic speech recognition (ASR) systems also employ rescoring mechanisms, such as the ROVER system (Fiscus, 1997). Athanaselis et al. (2006) use language model to search for reorderings of words in the context of automatic grammar checking. Our work differs from all of the above in that we already know (at least some of) the words that must appear in the recovered document. While this may appear to simplify the problem, our problem may in fact be harder. In SMT, the set of possible permutations of target words is constrained based on the *ordered* source language document (Kanthak et al., 2005; Bangalore et al., 2007). We lack such constraints in our task. We also consider decoding indicator vectors and vectors with stopwords removed, where we must decide how many additional words to introduce without the guide of a source document.

The current work shares commonality with data mining and security research that attempts to recover hidden knowledge from published sources. Staddon et al. (2007) study what inferences can be made when newly published data is combined with existing information on the Web. David Naccache and Claire Whelan demoed software at Eurocrypt 2004 that predicted redacted words in a White House briefing regarding September 11. Although different than our notion of document recovery, these work also address questions of privacy leakage from obscured or modified text.

Closer to our problem, Zhu et al. (2008) showed that using only a large set of BOW vectors, an adversary can learn a simple bigram model and recover simple documents. In that paper, no external language resources were used, but a large set of BOW vectors was assumed. In the current paper, in contrast, we use the state-of-the-art Google Web 1T 5-gram collection to provide a rich source of background knowledge for recovering documents from individual BOWs separately.

Finally, Soricut (2006) uses A* search with an admissible heuristic to recover short documents from BOWs. Soricut uses a different admissible heuristic than us, only uses a trigram language model based on the Wall Street Journal corpus, and only recovers short documents from count BOWs.

3 Problem formulation

Let $\mathbf{d} = w_1^n$ be a document with n words. We are interested in four types of indices:

1. *Count BOW*. The index is a vector $\mathbf{x} = (x_1, \ldots, x_V) \in \mathbb{Z}_+^V$, where x_v is the number of times word $v \in \mathcal{V}$ occurs in $\mathbf{d}$ and $\mathcal{V}$ is the vocabulary of length V.

2. *Stopwords-removed count BOW*. The index $\mathbf{x} = (x_1, \ldots, x_V) \in \mathbb{Z}_+^V$, where x_v is 0 if $v \in$ stopwords, and the number of times v occurs in $\mathbf{d}$ otherwise.

3. *Indicator BOW*. The index $\mathbf{x} = (x_1, \ldots, x_V) \in \{0, 1\}^V$, where x_v is 1 if v occurs in $\mathbf{d}$, and 0 otherwise.

4. *Bigram count vector*. The index $\mathbf{x} = (x_{11}, \ldots, x_{VV}) \in \mathbb{Z}_+^{V^2}$, where x_{ij} is the number of times the ordered pair v_i, v_j occurs in $\mathbf{d}$.

For any index $\mathbf{x}$, the *feasible set* $F(\mathbf{x})$ is defined as the set of documents that are consistent with $\mathbf{x}$, that is, the set of documents each having index $\mathbf{x}$: $F(\mathbf{x}) = \{\mathbf{d} : \mathbf{x}(\mathbf{d}) = \mathbf{x}\}$. The *document recovery problem* is to find the best document $\mathbf{d}^*$ in the feasible set given an index $\mathbf{x}$: $\mathbf{d}^* = \arg\max_{\mathbf{d} \in F(\mathbf{x})} \mathrm{score}(\mathbf{d})$ for some score function. For large $F(\mathbf{x})$, $\mathbf{d}^*$ may be impractical to compute exactly. In what follows, we define the feasible sets, score functions, and methods of approximating $\mathbf{d}^*$ for each index type.

4 Recovery from count BOWs

4.1 A* search

For count BOWs, the feasible set $F(\mathbf{x})$ is the set of permutations on words in $\mathbf{x}$. This set will be large for indices of even moderate size, so it is impractical to enumerate documents $\mathbf{d} \in F(\mathbf{x})$. Instead, we approximately find the best document using memory-bounded A* search (Russell and Norvig, 2002). A* search finds the highest-score path between a start and goal in a search space. It requries a score g of the path from the start state to the current state, and a heuristic estimate h of the score of the best path from the current state to the goal state. If the estimate h always overestimates the true score of the best path to the goal, h is called an admissible heuristic, and A* search is guaranteed to find the maximal-score path from start to goal.

To find $\mathbf{d}^*$ using A* search, let each state in the search space represent a partial path $w_1^l = w_1, \ldots, w_l$, where l is the length of the partial path. The start state consists of the start-of-sentence symbol <S>. The "real" goal state is the original document, but since this original document is unknown, the first complete path of length n that is found is accepted as the goal. In the absence of search error, this will be the highest-score length-n path.

The partial path score g of a partial document is the log probability of the partial document under a language model: $g(w_1^l) \equiv \log p(w_1^l) = \sum_{i=1}^{l} \log p(w_i | w_1^{i-1})$. We use a 5-gram language model with stupid backoff (Brants et al., 2007) to approximate the probabilities. Let $c(\cdot)$ be the n-gram count in the Google Web 1T 5-gram corpus, and $\alpha \in [0,1]$. Then the probability of a word w_i given its history w_{i-k+1}^{i-1} is as follows:

$$p(w_i | w_{i-k+1}^{i-1}) = \begin{cases} \frac{c(w_{i-k+1}^{i})}{c(w_{i-k+1}^{i-1})} & \text{if } c(w_{i-k+1}^{i}) > 0 \\ \alpha p(w_i | w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

if $k > 1$, and $p(w_i) = c(w_i) / \sum_{v \in \mathcal{V}} c(v)$ if $k = 1$. We use $\alpha = 0.4$, as suggested by the original paper.

We present two heuristics for h below: one that is admissible, and one that is not admissible but empirically performs better.

4.2 An admissible heuristic

We exploit properties of the stupid-backoff language model to construct an efficient admissible heuristic. Let $s(w_i | w_{i-k+1}^{i-1}) =$

$$\max\nolimits_{j=1}^{k} \alpha^{k-j} \begin{cases} \frac{c(w_{i-j+1}^{i})}{c(w_{i-j+1}^{i-1})} & \text{if } j > 1 \\ c(w_i) / \sum_{v \in \mathcal{V}} c(v) & \text{if } j = 1 \end{cases}$$

One can show that $s(w_i | w_{i-k+1}^{i-1}) \geq p(w_i | w_{i-k+1}^{i-1})$. Let $s^*(w) = \max_{v_1^4 \in \mathcal{V}} s(w | v_1^4) \geq \max_{v_1^4 \in \mathcal{V}} p(w | v_1^4)$ be an upper bound on the probability of the word $w \in \mathcal{V}$ under all possible 5-gram histories. Then the following is an admissible heuristic:

$$h^{\text{adm}}(w_1^l) = \sum_{w \in \mathbf{x} \setminus w_1^l} \log s^*(w), \tag{1}$$

where $\mathbf{x} \setminus w_1^l$ denotes the "remaining BOW", in which the count x_v is decremented for each occurrence of v in w_1^l.

The heuristic h^{adm} can be computed efficiently for each state, since the quantity s^* can be computed once, before A* search begins. While computing s^*, although there are V^4 possible permutations $v_1^4 \in \mathcal{V}$, we need only consider permutations for which n-gram counts exist; that is, s^* can be computed by scanning the list of n-grams used to construct our language model.

4.3 An empirical heuristic

Our second heuristic, h^{emp}, was originally proposed by Och et al. (2001) in the context of machine translation. To construct the heuristic h^{emp}, we first run A* search using h^{adm}. For each state w_1^l that is evaluated by this first search, we update the best-encountered probability for $v = w_l$: $p^*(v) \leftarrow \max\left(p^*(v), p(w_l | w_{l-4}^{l-1})\right)$, where $p^*(v)$ is initialized to $-\infty$. The second time we run A* search, we use these probabilities in the heuristic: $h^{\text{emp}}(w_1^l) = \sum_{v \in \mathbf{x} \setminus w_1^l} \log p^*(v)$.

The heuristic h^{emp} is not admissible, but it can give better results than h^{adm}. This is possible because pruning, described in Section 4.4, nullifies the optimality guarantee that is otherwise offered by A* search with an admissible heuristic. An admissible heuristic that is far from the true score may result in serious pruning, while a non-admissible heuristic that is generally closer to the true score may result in more moderate pruning and hence a better overall score. In this case, a non-admissible heuristic may perform better than an admissible heuristic. The heuristic h^{emp} is intended to give a more realistic estimate of the future path cost than h^{adm} does, and hence is expected to improve results.

4.4 Pruning strategies

Using A* search with h^{adm} or h^{emp}, the search space that needs to be explored is still huge and memory is frequently exhausted before a solution is found. In A* search, we store states in a priority queue $\mathcal{Q}$. Memory-bounded A* limits the size of this queue by removing states that look unpromising. We present three such pruning strategies.

The first strategy is to prune the state $w_1^l = \arg\min_{w_1^l \in \mathcal{Q}} g(w_1^l) + h(w_1^l)$ when the priority queue's size exceeds a threshold. However, as l increases, $g(w_1^l) + h(w_1^l)$ tends to increase as well, so this pruning strategy tends to prune long partial paths before shorter ones. This introduces a bias towards short partial paths, and can be undesirable. We call this first strategy $g + h$.

A second strategy is to prune based on the normalized score $\arg\min_{w_1^l \in \mathcal{Q}} \left(g(w_1^l) + h(w_1^l)\right)/l$ when the threshold is exceeded. This corrects the bias caused by differences in length among the states. We call this strategy $(g + h)/l$.

A third strategy is to prune short states first; that is, prune $\arg\min_{w_1^l \in \mathcal{Q}} \lambda\, l + g(w_1^l) + h(w_1^l)$. where λ is large enough that all states of length $l - 1$ are pruned before states of length l. (If this condition is met, the specific value of λ is unimportant.) This strategy directly exploits the earlier insight that short partial paths are most likely to be safe to prune. Note that using strategy $\lambda\, l + g + h$, after the memory limit has been reached, our A* search is similar to beam search.

4.5 Baseline

We compare our A* search procedure with a greedy baseline. The baseline constructs a document by repeatedly drawing a word, without replacement, from a count BOW and appending it to the partial document. At each iteration, the most likely word given the current partial document is drawn. Formally, each word w_i in the recovered document $\mathbf{d}$ is drawn such that $w_i = \arg\max_{v \in \mathbf{x} \setminus w_1^{i-1}} \log p(v|w_{i-4}^{i-1})$, where w_1^{i-1} is the current partial document.

5 Recovery from stopwords-removed count BOWs

In order for a document $\mathbf{d}$ to be in the feasible set $F(\mathbf{x})$ of a stopwords-removed count BOW $\mathbf{x}$, each word v in the vocabulary must occur $\mathbf{x}_v$ times in $\mathbf{d}$, unless v is a stopword. If v is a stopword, it can occur any number of times in the document. As a result, we also do not know n, the document length. This makes A* search difficult. For example, a short document that only uses words in $\mathbf{x}$ may have a higher language model score than a longer document that also uses stopwords. Conversely, a long document that repeats a high-probability phrase of stopwords like "and in the" may have a higher per-word score than a moderate-length document that uses fewer stopwords. In fact, per-word score can increase monotonically with partial document length. To simplify the problem and reduce the size of the feasible set, we infer document length from $\mathbf{x}$ and limit the feasible set to documents of that length.

To construct an estimator for document length n, given a stopwords-removed count BOW, we calculate the average ratio of document length (including stopwords) to BOW length (excluding stopwords) $\beta = \frac{1}{|D|} \sum_{\mathbf{d}_i \in D} \frac{n_i}{\mathbf{1}^\top \mathbf{x}_i}$ on a separate training set D of documents, where each document $\mathbf{d}_i$ has length n_i and a stopwords-removed BOW $\mathbf{x}_i$, and $\mathbf{1}$ is the all-one vector. Our document-length estimator, given a BOW $\mathbf{x}$ with unknown n and $\mathbf{d}$, simply scales BOW length by β: $\hat{n}(\mathbf{x}) = \beta \mathbf{1}^\top \mathbf{x}$.

To find the best document $\mathbf{d}^* \in F(\mathbf{x})$, we perform A* search as before, with one difference: successor states are generated by appending words drawn from the remaining BOW (without replacement), as before, but also from the stopwords list (with replacement). The heuristic h^{adm} is still admissible, although it is farther from the true future path score than before, since h^{adm} does not sum over stopwords' s^* scores.

6 Recovery from indicator BOWs

In order for a document $\mathbf{d}$ to be in the feasible set $F(\mathbf{x})$ of an indicator BOW $\mathbf{x}$, each word v in the vocabulary where $x_v = 1$ must occur at least once in $\mathbf{d}$, and each word v where $x_v = 0$ must not occur in $\mathbf{d}$. As before, document length n is unknown, so we infer it instead.

We use support vector machine regression to infer document length $\hat{n}$ from our indicator BOW $\mathbf{x}$. We will train a function $f(\mathbf{x}) = n$ on a training set D of documents, where each document $\mathbf{d}$ has length n and an indicator BOW $\mathbf{x}$. However, $\mathbf{x}$ itself is unsuitable as a feature vector: it has too many dimensions, so our estimator will suffer from data sparseness if it uses $\mathbf{x}$ directly. Instead we create a "count-count" histogram $\mathbf{h} \in \mathbb{Z}_+^{11}$, where h_j is the number of words types $v \in \mathbf{x}$ such that $c_{\text{google}}(v)$, the Google 1T 5-gram corpus count of v, has j digits ($10^{j-1} \leq c_{\text{google}}(v) < 10^j$). The histogram $\mathbf{h}$ has 11 dimensions because no count in the Google corpus has more than 11 digits. For example, let $\mathbf{x} = (1011010\ldots)$, where the four nonzero word types are "aardvark", "an", "apple", and "ate", with Google counts of 93030, 1369286818, 6878789, and 4763100 respectively. Then $\mathbf{h} = (00001020010)$, since the count of "aardvark" has five digits, the counts of "apple" and "ate" have seven digits, and the count of "an" has ten digits. We use $\mathbf{h}$ along with BOW length $\mathbf{1}^\top\mathbf{x}$ as our feature vector. We use SVM^{light} (Joachims, 1998) to train a support vector machine regression model $\hat{f}(\mathbf{h}, \mathbf{1}^\top\mathbf{x})$ with a linear kernel. To predict document length given a new indicator BOW $\mathbf{x}$, we take the maximum of the prediction and the BOW length:

$$\hat{n} = \max(\text{round}(\hat{f}(\mathbf{h}, \mathbf{1}^\top\mathbf{x})), \mathbf{1}^\top\mathbf{x}).$$

To find the best document $\mathbf{d}^* \in F(\mathbf{x})$, we perform A* search as for count BOWs, but when generating successor states, we append words drawn from the remaining BOW with replacement rather than without replacement. Additionally, we require that each word type in the BOW be used at least once in the complete document. The heuristic h^{adm} must be changed slightly, since our BOW now contains 1s and 0s, not counts: we require the summation in equation 1 to range over a remaining BOW $\mathbf{x}\backslash w_1^l$ in which each entry is 1 if $x_v = 1$ and $v \notin w_1^l$, and 0 otherwise. Since word types that will be reused in the future partial path are not accounted for in this version of h^{adm}, this version is farther from the true future path score than h^{adm} in the case of a count BOW. Nevertheless, this h^{adm} is still admissible, since repeated word types will only decrease the future path score.

7 Recovery from bigram count vectors

The feasible set of a bigram count vector is highly constrained. Document length n can be recovered exactly: $n = \mathbf{1}^\top\mathbf{x} + 1$, since each word token in the original document except the last token is counted in the first position of one bigram. Additionally, the bigram w_1, w_2 must occur x_{w_1,w_2} times in $\mathbf{d}$ for all $\mathbf{d} \in F(\mathbf{x})$.

We use these constraints to make the problem easier to solve. To motivate our discussion, suppose that a bigram count vector $\mathbf{x}$ has count 1 for each of "the dog", "dog runs", "runs quickly", and "runs slowly". There is ambiguity: we do not know whether "the dog runs quickly" or "the dog runs slowly" occurred in the original document. We do still know that "the dog runs" occurred.

We call document fragments that are unambiguously determined by the bigram count vector *sticks*. Before we start our search procedure proper, we construct a stick t_i for each bigram w_{i_1}, w_{i_2} such that $x_{w_{i_1},w_{i_2}} > 0$. We start with $t_i = w_{i_1}, w_{i_2}$. If we can extend leftward without ambiguity, that is, if there exists a w_{i_3} such that (a) $x_{w_{i_3},w_{i_1}} > 0$, (b) the bigram w_{i_3}, w_{i_1} will occur no more than $x_{w_{i_3},w_{i_1}}$ times in the new stick, and (c) $x_{w_{i_4},w_{i_1}} = 0$ for all $w_{i_4} \in \mathcal{V}$, $w_{i_4} \neq w_{i_3}$, then we extend $t_i = w_{i_3}, w_{i_1}, w_{i_2}$. Similarly, if we can extend rightward without ambiguity, we extend $t_i = w_{i_1}, w_{i_2}, w_{i_3}$. A single stick can be extended both leftward and rightward. We repeatedly extend each stick until an ambiguity is reached. We extend rightward as far as possible before extending leftward.

Once we have our sticks, we search for the best document $\mathbf{d}^* \in F(\mathbf{x})$ using A* search without heuristic. We extend each partial path by appending an unused stick to the partial path, and ignoring the beginning of the stick if it overlaps with the end of the partial path.

8 Experiments

8.1 Datasets

We collected a benchmark dataset consisting of 5 domains, each with 20 documents. The dataset sim-

Domain	s/d	w/d	Example
Medical	8	153	She also had some breathlessness . She also had some breathlessness . She also had some breathlessness . We have had some breathlessness .
			He was not wearing a helmet and was seen unconscious when paramedics arrived . He was unconscious when paramedics arrived and was seen not wearing a helmet . He was not wearing a helmet and was unconscious when paramedics arrived . seen . He was seen wearing a helmet was not unconscious when paramedics arrived .
CIA	22	487	These regiments are under a divisional headquarters called the 324 B Division . These are called the B 324 Division regiments under a divisional headquarters . B Division 324 These are called the regiments under a divisional headquarters . 324 regiments under Division B of A are called the divisional headquarters .
			The President had his breakfast during the meeting is [*sic*] the Situation Room Conference Room . The Situation Room is the President had his breakfast meeting during the Conference Room . The President had his breakfast . The Situation Room is the meeting during the Conference . It is during this meeting that the President had breakfast Situation Room Conference Room .
Email	11	228	I sincerely wish all of you the best in your future endeavors . I sincerely wish all of you the best in your future endeavors . I sincerely wish all of you the best of the best in all your future endeavors . sincerely wish all the best in future endeavors .
			PIRA is coming in May to do their semi - annual energy outlook . May PIRA is coming in to do their semi - annual energy outlook . May . PIRA is coming in to do their semi - annual energy outlook . PIRA is a semi - annual energy outlook for this coming May .
Stock	13	400	Our quality systems are ISO/TS16949 (2002 version) certified . Our quality systems are certified ISO/TS16949 (2002 version) . ISO/TS16949 Our quality systems are certified version) (2002) . ISO/TS16949 quality systems are certified to the 2002 and version ( ) .
			This department operates under the name of Stock Yards Trust Company . This department operates under the name of Trust Stock Yards Company . This department operates under the name of the Trust . Stock Yards Company . Trust in the Stock Yards Company operates under the name department .
SW	111	1771	B : uh - huh um - hum B : huh uh - um - hum . B : uh huh um - hum . Go to : A - B - um uh huh hum .
			A : halfway there so that 's good . A halfway there : so that 's good . A : halfway there so that 's good . so good : it 's halfway there .

Table 1: Statistics and example sentences from each domain. s/d = sentences per document; w/d = word tokens per document. The fourth column lists (1) the original document, (2) the document recovered from a count BOW, (3) the document recovered from an indicator BOW, and (4) the document recovered from a stopwords-removed count BOW.

ulates a collection of sensitive documents that an adversary might be interested in. The domains are:

1. Medical records: anatomy case studies from the University of Michigan's medical school at http://medical.med.umich.edu/courseinfo/clinical_index.html.

2. CIA: declassified CIA documents from Gale's Declassified Documents Reference System at http://galenet.galegroup.com/servlet/DDRS.

3. Email: messages from the Enron email corpus at http://www.cs.cmu.edu/~enron/.

4. Stock: annual reports aggregated by InvestorCalendar at http://investorcalendar.ar.wilink.com.

5. Switchboard (SW): telephone conversation transcripts from LDC online's free section of Switchboard.

Table 1 presents statistics and examples of each domain.

8.2 Experimental procedure

Document recovery is a difficult problem, and results are sensitive to the length of the documents. In order to compare our methods more effectively, we created synthetic documents of varying lengths N from each original document and tested on these synthetic documents. Each synthetic document is comprised of N contiguous real sentences from one original document, starting at a random sentence in the original. If the original document had fewer than N sentences, we took the entire document; for $N = 20$, this occurred 65 out of 100 times.

To evaluate the results of document recovery on each set of synthetic documents, we use BLEU4 (Papineni et al., 2001), comparing the recovered documents to the synthetic documents. Of course, BLEU4 is not the only possible metric. However, since our goal is to recover the exact document, BLEU4 seems to be a good choice for several reasons: (a) BLEU4 compares test documents with reference documents in terms of 4-gram precision. This correlates well with our goal to exactly recover each original document word-for-word. (b) BLEU4 penalizes recovered documents that are shorter than the originals. This is important in the case of indicator and stopwords-removed BOWs, where we need to recover the document's length as well as its contents. (c) BLEU4 is a standard metric for machine translation, and at the point of evaluation, machine translation and document recovery are quite similar. In both cases, we have a gold standard and want to measure how close our result is to the gold standard. A high BLEU4 score is, strictly speaking, neither necessary nor sufficient for a human to extract information from the recovered document; yet the same is also true in the case of machine translation, where BLEU4 has proved a useful metric.

All reported BLEU4 scores are averages within a set of synthetic documents for a single N and domain. In all our experiments, we set $\alpha = 0.4$ and $\lambda = 1000$. We use the small stopword list in (Manning and Schütze, 1999), which has 114 word types, 57 uppercase and 57 lowercase.

8.3 Results

Examples of a few recovered $N = 1$ documents are shown in Table 1. Tables 2 and 3 show the results of our experiments on different combinations of index types, heuristics, pruning strategies, domains, and document lengths, in terms of the BLEU4 scores. We make the following observations:

1. The results show two conditions under which we can recover documents with good success: (i) if the original document is short (Table 2, "$N = 1$" row), or (ii) if the index is a bigram count vector (Table 2, "bigram" column). Long documents, given a unigram BOW, are much more difficult to recover. This makes intuitive sense: when the original document is short or the index preserves ordering constraints, the feasible set is small, which makes recovery easier.

2. Among unigram BOWs, the index type affects the recovery rate. It is easiest to recover documents from count BOWs, somewhat harder to recover from indicator BOWs, and hardest of all from stopwords-removed count BOWS (Table 2, "counts, stopwords, indicator" columns). The fact that we must infer the document length from the BOW contributes to the difficulty of the latter two index types; when we artificially substitute the true original document length for our estimated document length, recovery improves, especially for short documents where each word is relatively more important (Table 2, "n" vs. "$\hat{n}$" columns).

3. The domains vary in difficulty of recovery. The medical and stock domains seem the easiest (rows in Table 2). This may be because they are both

more similar to general Web text than, for example, Switchboard, and our language model is trained on Web text.

4. Finally, Table 3 shows that our choice of heuristic and pruning strategy affects recovery. The empirical heuristic performs consistently better than the admissible heuristic. As for pruning strategies, $g + h$ is substantially the worst, but the other two strategies, $\lambda l + g + h$ and $(g + h)/l$, yield more or less equally good results. However, it is worth noting that A* search is much faster using $\lambda l + g + h$ than it is using $(g + h)/l$. For example, producing the third column of Table 3 took about an hour, while the second column took over a day.

9 Conclusion and Discussions

We have formulated the document recovery problem, constructed several initial attempts to solve it, and produced a benchmark test dataset which we plan to share with the research community after review. Our results for all conditions improve on the greedy baseline. Most importantly, we have shown that if original documents are short or the index is a bigram count vector, documents can be recovered from indices with high BLEU4 scores. This has important implications in security and privacy.

In future research, we would like to explore syntactic constraints to limit the feasible set and to improve our score function. We would also like to treat other types of indices, such as TF-IDF vectors or BOWs with stemming.

References

Theologos Athanaselis, Stelios Bakamidis, and Ioannis Dologlou. 2006. A fast algorithm for words reordering based on language model. In *Artificial Neural Networks - ICANN 2006*.

Srinivas Bangalore, Patrick Haffner, and Stephan Kanthak. 2007. Statistical machine translation through global lexical selection and sentence reconstruction. In *ACL*.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *EMNLP-CoNLL*.

Josep M. Crego, José Mariño, and Adrià Gispert. 2005. An ngram-based statistical machine translation decoder. In *Interspeech*.

J. Fiscus. 1997. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In *IEEE Workshop on Automatic Speech Recognition and Understanding*.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2001. Fast decoding and optimal decoding for machine translation. In *ACL*.

Ulrich Germann. 2003. Greedy decoding for statistical machine translation in almost linear time. In *NAACL*.

Hieu Hoang and Philipp Koehn. 2008. Design of the Moses decoder for statistical machine translation. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*.

T. Joachims. 1998. Making large-scale support vector machine learning practical. In A. Smola B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA.

Stephan Kanthak, David Vilar, Evgeny Matusov, Richard Zens, and Hermann Ney. 2005. Novel reordering approaches in phrase-based statistical machine translation. In *ACL Workshop on Building and Using Parallel Texts*.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4).

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

Franz Josef Och, Nicola Ueffing, and Hermann Ney. 2001. An efficient A* search algorithm for statistical machine translation. In *Data-Driven Machine Translation Workshop*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Stuart J. Russell and Peter Norvig. 2002. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall.

Radu Soricut. 2006. *Natural language generation using an information-slim representation*. Ph.D. thesis, University of Southern California.

Jessica Staddon, Philippe Golle, and Bryce Zimny. 2007. Web-based inference detection. In *USENIX-SS*.

Xiaojin Zhu, Andrew B. Goldberg, Michael Rabbat, and Robert Nowak. 2008. Learning bigrams from unigrams. In *ACL*.

N	domain	counts	baseline	stopwords, n	stopwords, $\hat{n}$	indicator, n	indicator, $\hat{n}$	bigram
1	medical	0.5774	0.0821	0.3998	0.3427	0.5590	0.3971	0.9294
	CIA	0.3704	0.1338	0.2525	0.2178	0.3541	0.2584	0.8515
	email	0.4845	0.1340	0.3639	0.2575	0.4242	0.3393	0.8122
	stock	0.5329	0.1023	0.4279	0.4282	0.4099	0.3329	0.8247
	SW	0.4793	0.1280	0.2866	0.2314	0.4341	0.3534	0.8522
2	medical	0.4389	0.0972	0.2957	0.2521	0.3683	0.2885	0.7336
	CIA	0.3348	0.0867	0.2334	0.2148	0.2689	0.2187	0.7438
	email	0.4366	0.1949	0.3595	0.3304	0.3634	0.3216	0.7566
	stock	0.4768	0.1125	0.3522	0.3240	0.3671	0.3313	0.7415
	SW	0.3882	0.0700	0.2163	0.1923	0.2900	0.2751	0.6570
5	medical	0.3363	0.0876	0.2413	0.2254	0.1997	0.2001	0.6902
	CIA	0.2381	0.0630	0.1803	0.1845	0.1631	0.1644	0.6710
	email	0.2822	0.0934	0.1894	0.1742	0.1982	0.2021	0.6519
	stock	0.3570	0.0787	0.2826	0.2624	0.2002	0.1896	0.6782
	SW	0.1904	0.0815	0.1293	0.1135	0.0983	0.0965	0.6018
10	medical	0.2852	0.0835	0.2216	0.2038	0.1514	0.1543	0.6739
	CIA	0.1835	0.0682	0.1570	0.1427	0.1048	0.1050	0.6027
	email	0.2239	0.1104	0.1738	0.1621	0.1501	0.1513	0.6254
	stock	0.2713	0.0891	0.2190	0.2129	0.1291	0.1192	0.6331
	SW	0.1352	0.0617	0.1075	0.0997	0.0493	0.0449	0.4910
20	medical	0.2823	0.0839	0.2200	0.2003	0.1531	0.1479	0.6687
	CIA	0.1737	0.0771	0.1515	0.1355	0.0866	0.0736	0.5513
	email	0.1928	0.1005	0.1556	0.1458	0.1155	0.1181	0.5697
	stock	0.2533	0.0878	0.2015	0.2090	0.1024	0.1148	0.6126
	SW	0.1239	0.0602	0.0989	0.0937	0.0285	0.0189	0.4365

Table 2: BLEU4 scores for synthetic subsets of the all domains and for each index type. In all cases, heuristic h^{emp} and pruning strategy $\lambda\, l + g + h$ are used.

N	Admissible h			Empirical h		
	$g + h$	$(g + h)/l$	$\lambda\, l + g + h$	$g + h$	$(g + h)/l$	$\lambda\, l + g + h$
1	0.3302	0.5714	0.5971	0.4120	0.5975	0.5936
2	0.2149	0.4267	0.3861	0.2708	0.4673	0.4804
4	0.2099	0.2575	0.2264	0.2099	0.3586	0.3369
16	0.1510	0.1934	0.1858	0.1847	0.2729	0.2970

Table 3: BLEU4 scores for synthetic subsets of the medical domain, with count BOW, for each heuristic and pruning strategy presented in Section 4. These synthetic subsets are a different random sample than those in Table 2.