Updating Rare Term Vector Replacement

Tobias Berka[†]
[†]Department of Computer Sciences
University of Salzburg
{tberka,marian}@cosy.sbg.ac.at

Marian Vajteršic[†‡]
[‡]Department of Informatics
Mathematical Institute
Slovak Academy of Sciences

Abstract

Rare term vector replacement (RTVR) is a novel technique for dimensionality reduction. In this paper, we introduce an updating algorithm for RTVR. It is capable of updating both the projection matrix for the reduction and the reduced corpus matrix directly, without having to recompute the expensive projection operation. We introduce an effective batch updating algorithm, and present performance measurements on a subset of the Reuters newswire corpus that show that a 12.5% to 50% split of the documents into corpus and update vectors leads to a three to four fold speedup over a complete rebuild. Thus, we have enabled optimized updating for rare term vector replacement.

1 Introduction

Rare term vector replacement (RTVR) is a recently developed linear dimensionality reduction technique for term frequency vectors (Berka and Vajteršic, 2011). It is easily and rapidly computed, patent-free, and produces a semantically meaningful multivariate space with a significantly reduced dimensionality. Furthermore, the method has been parallelized for data and task parallelism (Berka and Vajteršic, 2013).

The construction of this representation is based on document-scope term cooccurrences. Rare terms that occur only in δ documents or fewer are eliminated by replacing them with the average vectors of the documents that contain them. The replacement vectors of the rare terms are added to the original document vectors. Then, all rows for rare terms are dropped from all vectors. Only the common terms remain of the original term frequency vector space. By Zipf's law (Powers, 1998), we know that this operation will eliminate a high number of terms and lead to a highly condensed representation. The performance of the replacement now depends on the applicability of the *cluster hypothesis* (Raiber and Kurland, 2012; Rijsbergen, 1979), i.e., that documents in close proximity are semantically related. If the cluster hypothesis holds, then the centroid of the containing documents will act as a succint representation of all documents containing the replacement term.

The most prominent techniques for dimensionality reduction of text data in published literature are latent semantic indexing (LSI), see (Deerwester et al., 1990) and the related COV approach (Kobayashi et al., 2002). These two methods are applications of a principal component analysis to text using unbiased and biased correlation measures. Factor analysis based on the SVD applied to automated indexing has been reported as probabilistic latent semantic analysis (PLSA) (Hofmann, 1999). Updating operations for LSI are well understood (Zha and Simon, 1999), and are also being developed for PLSA (Bassiou and Kotropoulos, 2011), or other dimensionality reduction methods such as the kernel PCA (Mastronardi et al., 2010).

Through the connection between PLSA and Latent Dirichlet Allocation (LDA) (Girolami and Kabán, 2003), topic models are also related to dimensionality reduction. Positive matrix factorization methods are also used in various text analysis tasks (Zhang, 2010). Structurally, the generalized vector space model (GVSM) (Wong et al., 1985), is similar to RTVR because of the construction of index vectors by linear combination. The random index vector representation (Kanerva et al., 2000) is also based on cooccurrences, but operates on random initial vectors. Random projections can be used to further accelerate it (Sakai and Imiya, 2009), but this approach should be seen as complementary because it can be applied to other methods as well.

International Joint Conference on Natural Language Processing, pages 1002–1006,
Nagoya, Japan, 14-18 October 2013.

Our contribution is the following. In this paper, we rigorously define an algorithm for updating rare term vector replacement. Using our approach, both the replacement vectors and the reduced corpus matrix are updated directly. It is not necessary to explicitly recompute the projection into the reduced-dimensional space.

The remainder of this paper is structured as thus. Our main contribution is the updating algorithm in Section 2. Section 3 contains a theoretic and empirical performance evaluation. Lastly, we summarize our findings in Section 4.

2 Updating RTVR

Updating RTVR has to support the three basic operations of content management: (1) adding new documents, (2) changing existing documents, and (3) deleting obsolete documents.

The replacement vectors are weighted centroids, or weighted average vectors, of the document vectors containing their terms. The reduced document vectors are also linear combinations of the truncated original document vectors and the replacement vectors. At its core, the update algorithm is therefore a running average computation. We will use the mathematical notation summarized in Table 1 to describe the algorithm.

A key complication lies in the fact that the occurrence counts of the terms change. This means that some rare terms may become common terms, and therefore become part of the reduced-dimensional, projected term space. Dually, some common terms may become rare terms, and drop out of the projected space. We will refer to these terms as *promoted* and *demoted terms* P and Q. For demoted terms, we need to compute the replacement vectors from scratch during the update. But for all rare terms that are involved in an update, in the old or new vector, we need to change the replacement vector. These terms are called *affected* (rare) *terms* A_T.

We can represent changing a document by a tuple (i, v) containing a corpus matrix column $i \in \{1, ..., n\}$ and a new term frequency vector $v \in \mathbb{R}^{m'}$, where m' is the new number of terms. The other two updating operations can be cast into the same form by introducing two abstract symbols. We let ν denote the pseudo-column for adding new documents, i.e., (ν, d_{n+1}) denotes a new document that will be added as a new column of the corpus matrix. For deletions, we let ϵ denote an empty pseudo-vector for the deletion of an old document, i.e., (i, ϵ) signifies the deletion of the document in column i.

T	set of terms
D	set of documents
m	number of terms
n	number of documents
C	corpus matrix
$\mathcal{D}(t_i)$	set of documents containing t_i
$\mathcal{T}(d_j)$	non-zero terms in document d_j
N_i	occurrence count for term t_i
δ	occurrence count threshold
E	set of rare terms
τ_E	vector truncation removing indices in E
k	reduced dimensionality
π	index permutation mapping common features to reduced feature indices
R	replacement vectors
λ	normalizing factors
$\hat{C}$	reduced corpus
U	bulk update
(i, v)	update v for d_i
(ν, v)	insertion of document vector v
(i, ϵ)	deletion of document d_i
$\text{old}(i, v)$	old vector for i (or zero)
$\mathcal{T}_o(i, v)$	terms in the old vector
$\text{new}(i, v)$	new vector for i (or zero)
$\mathcal{T}_n(i, v)$	terms in the new vector
$\mathcal{T}(i, v)$	terms in both vectors
$\mathcal{T}(U)$	all terms $\bigcup_{(i,v)\in U} \mathcal{T}(i, v)$ in U
N'	new occurrence counts
E'	new rare terms
k'	new reduced dimensionality
π'	new index permutation
P	promoted terms $\{t \in E \mid t \notin E'\}$
Q	demoted terms $\{t \in E' \mid t \notin E\}$
A_T	affected terms $\mathcal{T}(U) \setminus (P \cup Q)$
σ	index permutation to remove demoted terms
e_i	i-th standard base

Table 1: Mathematical Notation

We associate every update $u = (i, v) \in U$ with two term frequency vectors. If the update is not an add document request, i.e., $i \neq \nu$, it is associated with an old document vector $\text{old}(i, v) = C_{1:m,i}$. If it is not a deletion, i.e., $v \neq \epsilon$, it is associated with a new document vector $\text{new}(i, v) = v$. We

Algorithm 1: Preparing the Update

delete or append terms in T, C, N, R, λ;
$N' := N$; $E' := E$;
for $u \in U$ **do**
  Used := Used $\cup\, \mathcal{T}(u)$;
  for $t_i \in (\mathcal{T}_o(u) \setminus \mathcal{T}_n(u))$ **do** N'_i--;
  for $t_i \in (\mathcal{T}_n(u) \setminus \mathcal{T}_o(u))$ **do** N'_i++;
for $t_i \in T$ **do**
  if $(N'_i > \delta) \wedge t_i \in E$ **then**
    $P := P \cup \{t_i\}$; $E' := E' \setminus \{t_i\}$;
  else if $(N'_i \leq \delta) \wedge t_i \notin E$ **then**
    $Q := Q \cup \{t_i\}$; $E' := E' \cup \{t_i\}$;
  else if $(N'_i \leq \delta) \wedge t_i \in$ *Used* **then**
    $A_T := A_T \cup \{t_i\}$;
$k'' := k - \|Q\|$; $k' := k'' + \|P\|$;
$\sigma := 1_{k'}$; $j := 1$; $l := k'' + 1$;
for $t_i \in T$ **do**
  if $t_i \in P$ **then** $\pi'(i) := l$++;
  else if $t_i \in A_T$ **then**
    $\sigma(j) := \pi(i)$;
    $\pi'(i) := j$++;
  else $\pi'(i) := -1$;

will need to identify the terms in the old vector $\mathcal{T}_o(i, v)$, in the new vector $\mathcal{T}_n(i, v)$, and the joint set $\mathcal{T}(i, v)$. All terms in the old and new vectors for the entire update U is defined as $\mathcal{T}(U)$.

Our updating algorithm proceeds in three phases: (1) preparing the update, (2) downdating the reduced corpus and updating the replacement vectors, and (3) updating the reduced corpus. In Algorithm 1, we analyze a batch update and prepare the required sets of features and an index permutation σ to compact the reduced space.

Let us assume that we have a procedure Compact(A,m',n,σ), which applies the permutation σ to the row indices of a matrix. If we have a matrix $A \in \mathbb{R}^{m \times n}$ and compute $A' :=$ Compact$(A, m', n, \sigma) \in \mathbb{R}^{m' \times n}$, it holds that $A'_{i,j} = A_{\sigma(i),j}$. Let us further assume that truncation of the new elimination terms $\tau_{E'}$ respects the new index order established with σ, i.e., the implementation uses the global index permutation π' mapping all new features to indices in $\{1, ..., k'\}$.

We downdate the reduced corpus and update the replacement vectors with Algorithm 2. For existing documents, we downdate the reduced corpus by subtracting any terms that will change in the course of the update in Line 1. We then update the replacement vectors by adding the new document vector to any affected or demoted features in Line 2. Documents that do not change contribute to the construction of replacement vectors for demoted terms, which need to be built from scratch, in Line 3. These have to be inserted into rows k'' to k' of the reduced space, as done in Line 4. We add in any new documents in Line 5 and normalize the resulting replacement vectors.

Algorithm 3 updates the reduced vectors by adding all replacement vectors that have changed in Line 1. The exception are the promoted terms on rows k'' to k', which must be added for all rare terms that were otherwise unaffected by the update in Line 2. We then handle all deletions and insertions in Lines 3 and 4.

3 Performance Evaluation

Regarding the asymptotic complexity of our updating algorithm, we note the following. Assuming amortized constant time for set testing and insertion (Sedgewick, 2002), that the update is smaller than the corpus, i.e., $\|U\| < \|D\|$, and that the number of documents is greater than the number of terms, i.e., $n > m$, the update algorithm can be executed with a complexity of $O((\|D\| + \|U\|)\,\overline{\text{nnz}}\;\; k' + mk')$, where $\overline{\text{nnz}}$ is the expected number of non-zero elements in any document vector.

Since the performance is heavily dependent on the actual distribution of the non-zero elements, the observed performance may differ somewhat from this formal analysis. We have conducted performance measurements using an Intel i5-2557 with 4 GB or RAM running Max OS X 10.7.5. We have used the first 23,149 documents with 47,236 terms of the Reuters Corpus Volume I, version 2, in the pre-vectorized form (Lewis et al., 2004).

We randomly selected between 12.5% and 50% of all vectors for the batch update, using the remaining documents for the initial build. Table 2 summarizes our performance measurements averaged across ten runs per row.

The results clearly show that the updating algorithm outperforms a complete rebuild with the original construction algorithm by three-fold to four-fold performance improvement. Because smaller updates require less processing in the updating, and the workload for the rebuild remains the same, smaller batches have a larger speed-up than smaller batches.

Algorithm 2: Downdating the Reduced Corpus and Updating the Replacement Vectors

for $t_i \in A_T$ **do** $R_{*,i}$ *= λ_i;

$R' = \begin{bmatrix} \text{Compact}(R, k'', m, \sigma) \\ 0 \end{bmatrix} \in \mathbb{R}^{k' \times m}$;

for $t_i \in Q$ **do** $R'_{*,i} = 0$;

for $d_j \in D$ **do**

1 **for** $t_i \in \mathcal{T}(d_j)$ **do**

if $(j, d'_j) \in U \wedge t_i \in A_T$ **then**

for $l \in \{1, ..., k''\}$ **do**

$R'_{l,i}$ -= $C_{i,j} \tau_E(C_{*,j})_{\sigma(l)}$;

λ_i -= $|C_{i,j}|$;

if $(j, d'_j) \notin U \wedge t_i \in A_T \cup P$ **then**

$\hat{C}_{*,j}$ -= $C_{i,j} R_{*,i}$;

2 **if** $(j, d'_j) \in U$ **then**

for $t_i \in \{t_i \in T \mid (d'_j)_i \neq 0\}$ **do**

if $t_i \in Q \cup A_T$ **then**

for $l \in \{1, ..., k'\}$ **do**

$R'_{l,i}$ += $(d'_j)_i \tau_{E'}(d'_j)_l$;

λ_i += $|(d'_j)_i|$;

3 **else for** $t_i \in \mathcal{T}(d_j)$ **do**

if $t_i \in Q$ **then**

for $l \in \{1, ..., k'\}$ **do**

$R'_{l,i}$ += $C_{i,j} \tau_{E'}(C_{*,j})_l$;

λ_i += $|C_{i,j}|$;

else if $t \in E'$ **then**

4 **for** $l \in \{k'', ..., k'\}$ **do**

$R'_{l,i}$ += $C_{i,j} \tau_{E'}(C_{*,j})_l$;

if $(\mathcal{T}(d_j) \cap E') \subseteq P$ **then**

λ_i += $|C_{i,j}|$;

5 **for** $(\nu, v) \in U$ **do**

for $t_i \in \mathcal{T}_n(\nu, v) \cap E'$ **do**

for $l \in \{1, ..., k'\}$ **do**

$R'_{l,i}$ += $C_{i,j} \tau_{E'}(v)_l$;

λ_i += $|v_j|$;

for $t_i \in E'$ **do**

if $t_i \in P$ **then** $R'_{*,i} = \tau_{E'}(e_i)$;

else $R'_{*,i}$ /= λ_i;

Algorithm 3: Updating the Reduced Corpus

$\hat{C}' = \begin{bmatrix} \text{Compact}(\hat{C}, k'', n, \sigma) \\ 0 \end{bmatrix} \in \mathbb{R}^{k' \times n}$;

Old $= E' \setminus (P \cup A_T \cup Q)$;

1 **for** $d_j \in D$ **do**

if $(j, d'_j) \in U$ **then**

$\hat{C}'_{*,j} = \tau_{E'}(d'_j) + \sum_{t_i \in E'} (d'_j)_i R'_{*,i}$;

else

$\hat{C}'_{*,j}$ += $\sum_{t_i \in P} C_{i,j} \tau_{E'}(e_i)$;

$\hat{C}'_{*,j}$ += $\sum_{t_i \in Q \cup A_T} C_{i,j} R'_{*,i}$;

2 $\hat{C}'_{k'':k',j}$ += $\sum_{t_i \in \text{Old}} C_{i,j} R'_{k'':k',i}$;

3 **if** $(j, \epsilon) \in U$ **then**

$\hat{C}' = [\ \hat{C}'_{*,1:j-1} \quad \hat{C}'_{*,j+1:n}\ ]$;

n - -;

4 **for** $(\nu, v) \in U$ **do**

$v' = \tau_{E'}(v) + \sum_{t_i \in E'} v_i R'_{*,i}$;

$\hat{C}' = [\ \hat{C}' \quad v'\ ]$;

n ++;

Size	t_R	t_I	t_U	S
12.5%	51.41	43.59	11.44	4.49
25.0%	51.41	40.73	13.81	3.72
37.5%	51.41	34.57	15.15	3.39
50.0%	51.41	28.52	16.12	3.18

Table 2: Performance evaluation (rebuild time t_R, build time t_I, update time t_U and speed-up S).

4 Summary & Conclusions

In this paper, we have introduced an algorithm for updating rare term vector replacement. Our empirical performance evaluation demonstrates that batch updating is faster than a complete rebuild by a factor of three to four for our experiments. In our future research, we intend to develop hybrid updating algorithms similar to (Tougas and Spiteri, 2008). These algorithms initially compute fast, approximate updates, which are only later replaced by exact updates for efficiency. The final PCA of the augmented corpus $\hat{C}$ reported in (Berka and Vajteršic, 2011) remains an open problem in updating RTVR.

Acknowledgements

We acknowledge the support of the Slovak Ministry of Education and Slovak Academy of Sciences under VEGA grant no. 2/0003/11.

References

N. Bassiou and C. Kotropoulos. 2011. RPLSA: A Novel Updating Scheme for Probabilistic Latent Semantic Analysis. *Comput. Speech Lang.*, 25(4):741–760.

T. Berka and M. Vajteršic. 2011. Dimensionality Reduction for Information Retrieval using Vector Replacement of Rare Terms. In *Proc. TMW*.

T. Berka and M. Vajteršic. 2013. Parallel Rare Term Vector Replacement: Fast and Effective Dimensionality Reduction for Text. *J. Parallel Distr. Com.*, 73(3):341–351.

S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. 1990. Indexing by Latent Semantic Analysis. *JASIS*, 41(6):391–407.

M. Girolami and A. Kabán. 2003. On an Equivalence Between PLSI and LDA. In *Proc. SIGIR*, pages 433–434, USA. ACM.

T. Hofmann. 1999. Probabilistic Latent Semantic Indexing. In *Proc. SIGIR*, pages 50–57, USA. ACM.

P. Kanerva, J. Kristoferson, and A. Holst. 2000. Random Indexing of Text Samples for Latent Semantic Analysis. In *Proc. CogSci*, pages 103–106. Erlbaum.

M. Kobayashi, M. Aono, H. Takeuchi, and H. Samukawa. 2002. Matrix Computations for Information Retrieval and Major and Outlier Cluster Detection. *J. Comput. Appl. Math.*, 149(1):119 – 129.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. *JMLR*, 5:361–397.

N. Mastronardi, E. E. Tyrtyshnikov, and P. Van Dooren. 2010. A Fast Algorithm for Updating and Downsizing the Dominant Kernel Principal Components. *SIAM J. Matrix Anal. Appl.*, 31(5):2376–2399.

D. M. W. Powers. 1998. Applications and Explanations of Zipf's Law. In *Proc. NeMLaP3/CoNLL*, pages 151–160, USA. ACL.

F. Raiber and O. Kurland. 2012. Exploring the Cluster Hypothesis, and Cluster-Based Retrieval, Over the Web. In *Proc. CIKM*, pages 2507–2510, USA. ACM.

C. J. van Rijsbergen. 1979. *Information Retrieval*. Butterworths, London.

T. Sakai and A. Imiya. 2009. Fast Spectral Clustering with Random Projection and Sampling. In *Machine Learning and Data Mining in Pattern Recognition*, LLNCS, pages 372–384. Springer.

R. Sedgewick. 2002. *Algorithms in C++*. Addison Wesley, 3rd edition.

J. E. Tougas and R. J. Spiteri. 2008. Two Uses for Updating the Partial Singular Value Decomposition in Latent Semantic Indexing. *Appl. Numer. Math.*, 58(4):499–510.

S. K. M. Wong, W. Ziarko, and P. C. N. Wong. 1985. Generalized Vector Spaces Model in Information Retrieval. In *Proc. SIGIR*, pages 18–25, USA. ACM.

H. Zha and H. D. Simon. 1999. On Updating Problems in Latent Semantic Indexing. *SIAM J. Sci. Comput.*, 21(2):782–791.

Z.-Y. Zhang. 2010. Survey on the Variations and Applications of Nonnegative Matrix Factorization Variations of NMF. *Operations Research*, pages 317–323.