# Information Mapping

– Concept-based Information Retrieval based on Word Associations

Yasuhiro Takayama *, Raymond S. Flournoy †, Stefan Kaufmann ‡
Center for the Study of Language and Information
Stanford University

August 27, 1998

## 1 Introduction

Thinking and communicating are *situated* activities that always occur within some particular context [3]. In the Computational Semantics Laboratory at Center for the Study of Language and Information (CSLI), directed by Stanley Peters, we are developing models of language, computation and inference which take into account the context in which these activities occur. We then apply these techniques to problems in information retrieval, natural language processing (NLP), and communication among software agents.

Our various projects seek answers to a number of practical questions: How can I retrieve the documents I want from the Internet? How can I get a robot to understand my request based on the current context? How can software agents best communicate in order to solve complex problems cooperatively?

This document describes one of our subprojects: "Information mapping" (**InfoMap**, for short).

*CSLI visiting scholar, Mitsubishi Electric Corp.
†Computer Science Department and CSLI
‡Linguistics Department and CSLI

## 2 Information Mapping and Word Space

### 2.1 Associative Information Retrieval

The goal of the InfoMap project is intelligent, concept-based information retrieval. Currently, document retrieval from large text databases—such as library card catalogs or newspaper archives—is based on keyword search. A query is posed as a list of words, and any entries in the database which contain any or all of those specific words are returned. However, if we treat those query words not as literal strings of letters, but as representing *concepts*, then we can retrieve relevant documents even if they do not contain the specific words used in the query.

Our basic approach, developed by Hinrich Schütze [13], begins by recording the frequency of co-occurrence between words in the text; that is, the number of times two words appear "near" each other, e.g., in the same document. The distribution of co-occurrences between a word and some set of *content-bearing words* then serves as a profile of the word's usage, and thus of its meaning as well. By comparing the profiles of different words,

Table 1: An example of co-occurrence matrix

| | content-bearing words | | | |
|---|---|---|---|---|
| words | market | ··· | last | ··· |
| ··· | ··· | ··· | ··· | ··· |
| *sunday* | 97 | ··· | 215 | ··· |
| ··· | ··· | ··· | ··· | ··· |
| *weekend* | 201 | ··· | 408 | ··· |

we can construct a measure of how related those words are. Generalizing this word similarity derived from lexical co-occurrence, by comparing the query words' profiles to profiles generated for each document, we can return documents which we judge to be conceptually related to the query words, even if the words themselves do not appear in the text — this is what we call *associative information retrieval*.

## 2.2 Word Space and SVD

The lexical co-occurrences between a word and content-bearing words are recorded in the *co-occurrence matrix* which creates a high-dimensional space. This abstract space forms a concept space in which similar words (or more specifically, words with similar *distributional* behavior) have similar vectors (See Table 1).

The co-occurrence matrix suffers from two problems: too many word features and data sparseness. To solve these problems, we apply SVD (Singular Value Decomposition) [6] to the co-occurrence matrix as a tool for dimensionality reduction and generalization. SVD factors every $m$ by $n$ matrix $A$ into

$$\underset{m\times n}{A} = \underset{m\times m}{U}\ \underset{m\times n}{\Sigma}\ \underset{n\times n}{V^T} \qquad (1)$$

where the left matrix $U$ and the right matrix $V$ are orthogonal matrices and the singular matrix $\Sigma$ is diagonal.

Equation (1) shows the *full* SVD in linear algebra. We use the left orthogonal matrix $U$ as the reduced matrix, the output from the *partial* SVD (Figure 1). The rows of the reduced matrix – *word vectors* – approximate *associations* among the word senses. This reduced space from the previous concept space is called Word Space . It potentially reflects *associative* behavior of words captured through *second-order co-occurrence* information.

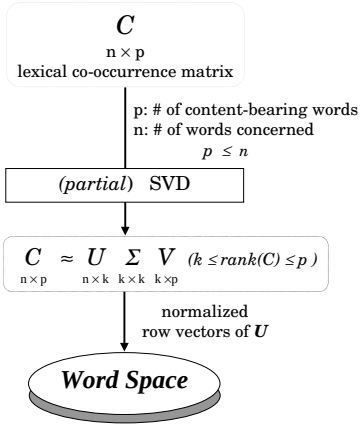


Figure 1: Partial SVD for Word Space

Another use of SVD in information retrieval is word by document matrix reduction for *LSI (Latent Semantic Indexing)* [2]. The difference between Word Space and LSI is discussed in [14].

By clustering the word vectors based on their proximity, the Word Space can be used for the word sense disambiguation and thesaurus construction [13] [15].

## 2.3 SVD and Principal Component Analysis

SVD is not a direct statistical technique but rather a matrix factorization in linear algebra

[16]. When a matrix to be processed consists of some statistical observations, SVD becomes a powerful tool for statistical analysis.

SVD has a close relationship with *principal component analysis* (PCA), a feature reduction technique used in multivariate analysis. [9] [13]. Multivariate analysis concerns associations among multiple variables (features) with the goal of discovering relationships among the multivariate profiles of the data.

Suppose that matrix $X$ is a $p \times n$ matrix of observations (or a data matrix). If matrix $B$ is a matrix in mean-deviation form of the data matrix $X$, and if $A = (1/\sqrt{n-1})B^T$, then $A^T A$ becomes the *unbiased* covariance matrix $S$. (The superscript $T$ denotes transposition). We can calculate the eigenvalues and the eigenvectors by the eigenvalue decomposition from the $p \times p$ covariance matrix $S$.

Eigenvalue decomposition can be applied to the square matrices only, but SVD can be applied to any rectangular matrices. Thus the calculation of SVD is more convenient than eigenvalue decomposition.

SVD can be used as a tool for performing PCA. When we apply SVD to the matrix $A$, the square of the singular values of $A$ are the $p$ eigenvalues of the covariance matrix $S$, and the right singular vectors $[\mathbf{v}_1 \cdots \mathbf{v}_p]$ of $A$ are the coefficients of the principal components of the data in the matrix $X$. Then $\mathbf{v}_i^T X$ is the $i$-th principal component (See Figure 2).

In Word Space , we directly apply SVD to the original data matrix (i.e. lexical co-occurrence matrix $C$ in our case) instead of the matrix $A$, the mean-deviation form with a coefficient $1/\sqrt{n-1}$ (See Figure 1).

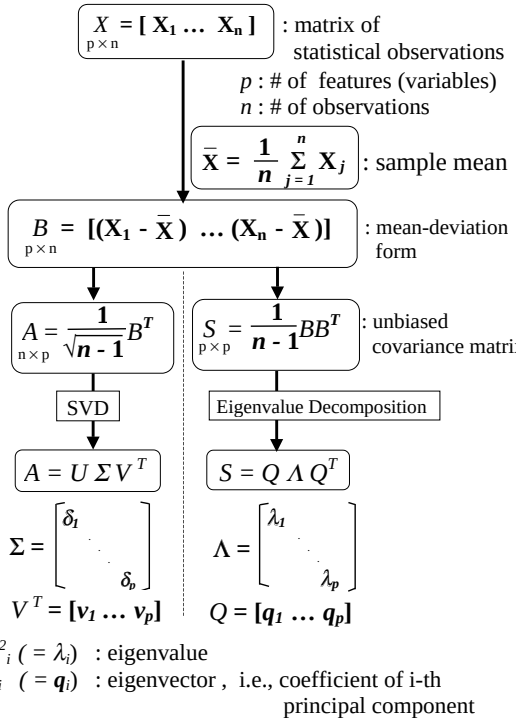


Figure 2: Relation between SVD and PCA

# 3 System Organization

The retrieval model of the InfoMap search engine is based on a *vector space model* [12], that is, the documents and the queries are represented as vectors in the high-dimensional space, just as the words are.

The search engine of InfoMap consists of the document registration phase that creates the Word Space (concept base) and the document retrieval phase, similar to other information retrieval systems. This section illustrates the functions of these phases.

## 3.1 Word Space based on lexical co-occurrence

The document registration phase of InfoMap is the Word Space (concept base) construction

functions based on lexical co-occurrence in the text corpus (Figure 3).

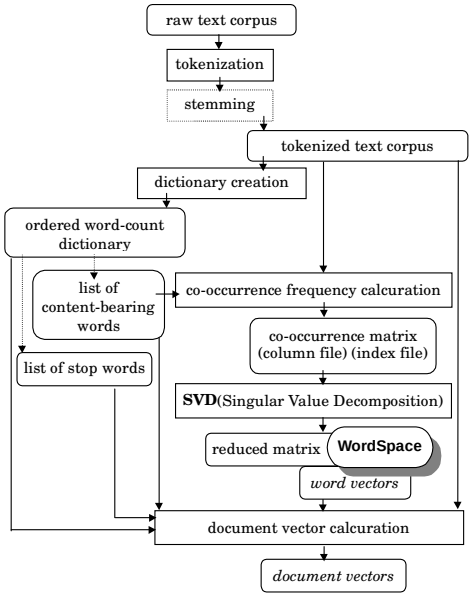


Figure 3: Word Space Construction

1. Tokenization of text corpus

   The first stage of processing produces a *tokenized corpus*. The corpus (collection of documents) can be tokenized by passing it through a tokenizer or a morphological analyzer. The *stemming* [10], [5] in the tokenizer is optional.

2. Calculation of word frequencies

   The second stage of processing produces a *word count dictionary*. The count dictionary is a word list of tokens and their frequencies in the corpus, ordered by frequency of appearance of the tokens.

3. Calculation of co-occurrence frequencies

   For each of the 20,000[1] most frequently occurring words in the corpus, a vector of 1,000 co-occurrence counts is created, and these vectors serve as profiles of each word's distribution. The 1,000 entries in the vector represent a set of 1,000 words which have been determined to be *content-bearing* in the following sense.

   The content-bearing words are chosen by considering either the word's total frequency of appearance in the corpus, the word's part-of-speech information, or a calculation of the relative concentration of the word within the documents in the corpus. This calculation – called the "dispersion" of a word – exploits the idea that words which are not distributed evenly throughout the documents in a corpus are more likely to be content-bearing.

   We choose the 51 to 1,050 most frequently occurring words in the corpus as a basic set of the content-bearing words.

   Each time one of the 20,000 count words appears within a *window* – a specific range around one of the content-bearing words – the appropriate count in its vector is incremented. A word falls within range if it is within a certain distance from the content-bearing word, or if it is within the same sentence, paragraph, or document as the content-bearing word.

   After all documents in the corpus have been processed, the square root of each count is taken to smooth out the effects of extreme numbers, and the vectors are written out to disk. So the actual $(i, j)$-th element of the co-occurrence matrix is

[1] The numbers of the dimensions in this document are example ones we used in our experiment. They can be changed by setting the parameters in the system configuration.

represented by a real value:

$$c_{ij} = \phi(cooc_{ij}) \tag{2}$$

where $cooc_{ij}$ is the co-occurrence count of word $i$ within a window from a content-bearing word $j$ throughout the corpus, and $\phi$ is the transformation of the count data. We use the square root as the basic transformation but other transformations might be useful. The standard setting of the window size is 51 (25 words to the left and to the right of the current word).

4. Analysis of the second-order co-occurrence

   The 20,000 vectors (the rows of the co-occurrence matrix) represent points in a 1,000-dimensional space. To make computations using the concept space more tractable, it is necessary to lower the dimensionality of the space. The tool we use for reducing the dimensionality of the co-occurrence count matrix is SVD [6].

   This calculation is done by feeding the matrix through the SVDPack software package[2] [1], a process which iteratively extracts the most important dimensional features to approximate the high-dimensional space with one of a much lower dimensionality.

   The left orthogonal matrix $U$, the output of the partial SVD in Figure (1) is now reduced to 100 dimensions. To calculate the normalized vectors, the rows of the reduced left matrix are divided by their lengths, converting them to unit vectors. These normalized left singular vectors serve as the word vectors $\mathbf{u}_i$ $(i = 1, \ldots, 20,000)$ in Word Space derived from the lexical co-occurrence.

[2]Copyright 1993, University of Tennessee, distributed through http://www.netlib.org.

5. Creation of document vectors on Word Space

   (a) Each document is processed into a *document vector* of length 100. This is done by reading in the individual word vectors previously calculated for the 20,000 most frequently occurring words in the corpus, and summing the normalized vectors corresponding to each of the words in the document:

   $$\mathbf{d}_j = \sum_i w_{ij}\mathbf{u}_i \tag{3}$$

   where $\mathbf{d}_j$ is the document vector for document $j$, $w_{ij}$ is the weight for word $i$ in document $j$, and $\mathbf{u}_i$ is the word vector for word $i$ occurred in document $j$. The default weight $w_{ij}$ is 1. The $tf \cdot idf$ (term frequency $\cdot$ inverse document frequency) weight is used in [14].

   Optionally, one may choose to disregard the vectors of *stop words*, certain words that are expected to be so general or so common that they will not contribute informatively to the vector. We use the 1 to 50 most frequently occurring words in the corpus as a basic set of the stop words.

   After document vectors are calculated for each of the documents in the corpus, they are written to disk with the byte location of the document.

   (b) The 100-dimensional space which these vectors occupy embodies the document concept base derived from the corpus, and each of these vectors represents a specific location within this space corresponding to the meaning or subject matter of

the document. Furthermore, the formalism predicts that vectors which lie close to each other in the concept space correspond to documents which are somehow related in subject matter.

For simplicity, our explanation has included individual words as the dimensions of the co-occurrence matrix. Optionally, we also choose the statistically significant phrases based on a $\chi^2$-test [11] that is applied to a contingency table of the neighboring word counts [15].

To find the pairs that most frequently "stick" together, we count all neighbor words, and sort them by frequency, then calculate their $\chi^2$-value. A certain number (e.g. 5,000) of the top $\chi^2$-valued words are considered *sticky pairs*. We also allow these sticky pairs to be elements of the row dimension of the co-occurrence matrix.

## 3.2 Document retrieval on Word Space

The main stages of the document retrieval phase of InfoMap are the query vector calculation, the closeness calculation and the actual retrieval (Figure 4).

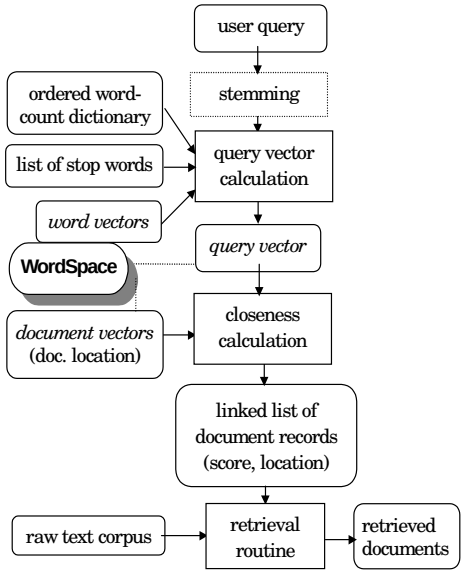


Figure 4: Document retrieval on Word Space

1. Query vector calculation

   To retrieve the documents from the corpus using the associations in Word Space , a query in the form of a list of words (either entered interactively or stored in a file) is translated into the corresponding set of the normalized word vectors, and these are summed to form a *query vector*:

$$\mathbf{q} = \sum_i w_i \mathbf{u}_i \qquad (4)$$

where $\mathbf{q}$ is the query vector, $w_i$ is the weight for word $i$ in the query (default weight is 1), and $\mathbf{u}_i$ is the word vector for word $i$ occurred in the query.

2. Closeness calculation

   The query vector is then compared with each of the document vectors and the documents whose vectors lie closest to the query vector are returned.

   The *closeness* of two vectors (the query vector $\mathbf{q}$ and an document vector $\mathbf{d}_j$) is determined by calculating the cosine of the angle between the vectors [3]

$$closeness(\mathbf{q}, \mathbf{d}_j) = \frac{\mathbf{q} \cdot \mathbf{d}_j}{\|\mathbf{q}\|\|\mathbf{d}_k\|} \quad (5)$$

   This routine requires the document vectors ($\mathbf{d}_j$'s) as its input and returns a linked-list of document records, ordered by closeness with the query vector($\mathbf{q}$). Each document record contains the cosine score for the document and the byte location of the document in the corpus.

3. Retrieval and display of documents

   The retrieval routine simply goes to the appropriate location in the document records and displays the documents as requested by the user.

Query vectors and document vectors are represented as normalized word vector sums (*centroids*). These vectors are called *context vectors* in general.

[3] In order to find similar words, the closeness (proximity) of word vectors are also calculated by the cosine measure.

# 4 Current and Future Work

Associations in Word Space are computed from unannotated text corpora in an unsupervised way as described in the previous sections. We would like to demonstrate that these word associations are useful for associative information retrieval.

Our experiments with InfoMap thus far mainly have used collections of newswire articles as a source of *general* associations. We are currently investigating how different training corpora affect the resulting search engines, in particular whether the use of personal email to train a search engine produces one which is tuned to reflect that user's interests – *personal* associations. A preliminary study with approximately a dozen human subjects is discussed in [4]. The *domain-specific* associations from topical corpora such as medical texts [7] is another interesting experiment.

In addition, we have applied the information mapping technique to term-list translation between English and Japanese [8]. As a future research topic, we are hoping to investigate how the concept space created by our technique can be used to do cross-lingual information retrieval.

# References


[1] Michael W. Berry: *Large Scale Singular Value Computations*, International Journal of Supercomputer Applications, 6:1, pp. 13-49, 1992.

[2] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman: *Indexing by latent semantic analysis.* Journal of the American Society for Information Science, 41(6):391-407, 1990.

[3] Keith Devlin: *Logic and Information.* Cambridge University Press, 1991.

[4] Raymond S. Flournoy, Ryan Ginstrom, Kenichi Imai, Stefan Kaufmann, Genichiro Kikui, Stanley Peters, Hinrich Schütze, Yasuhiro Takayama: *Personalization and Users' Semantic Expectations.* ACM SIGIR'98 Post-Conference Workshop on Query Input and User Expectations, Melbourne, Australia, August 28, 1998.

[5] William B. Frakes: *Stemming algorithms.* In W. B. Frakes and R. Baeza-Yates (Eds.), Information Retrieval, Data Structures and Algorithms, pp.131-160, Englewood Cliffs, NJ, Printice Hall, 1992.

[6] Gene H. Golub, Charles F. Van Loan: *Matrix Computation.* 3rd ed., The Johns Hopkins University Press, 1996.

[7] W. R. Hersh, C. Buckley, T. J. Leone, D. H. Hickam: *OHSUMED: An interactive retrieval evaluation and new large test collection for research.* Proceedings of the 17th Annual ACM SIGIR Conference '94, pp. 192-201, 1994.

[8] Genichiro Kikui: *Term-list Translation using Mono-lingual Word Co-occurrence Vectors.* Project Note, COLING-ACL '98, August 10-14, 1998.

[9] David C. Lay: *Linear Algebra and its applications.* revised ed., 1997.

[10] M. F. Porter: *An algorithm for suffix stripping.* Program, 14, pp.130-137, 1980.

[11] Fred L. Ramsey, Daniel W. Schafer: *The Statistical Sleuth – A Course in Methods of Data Analysis.* Duxbury Press, 1997.

[12] Gerard Salton, A. Wang, C. S. Yang: *A vector space model for automatic indexing.* Communications of the ACM, 18, pp.613-620, 1975.

[13] Hinrich Schütze: *Ambiguity in Language Learning: Computational and Cognitive Models.* PhD thesis, Stanford University, Department of Linguistics, July 1995. (Revised thesis, *Ambiguity Resolution in Language Learning: Computational and Cognitive Models*, CSLI Lecture Notes 71, CSLI Publications, 1997).

[14] Hinrich Schütze, Jan O. Pedersen: *A coocurrence-based thesaurus and two applications to information retrieval.* Information Processing & management, Vol.33, No.3, pp.307-318, 1997.

[15] Hinrich Schütze: *Automatic Word Sense Discrimination.* Computational Linguistics, Volume 24, Issue 1, pp.97-123, March 1998.

[16] Gilbert Strang: *Introduction to Linear Algebra.* Wellesley-Cambridge Press, 1993.