DSIR: the First TREC-7 Attempt

Arnon Rungsawang
fenganr@ku.ac.th
Department of Computer Engineering, Faculty of Engineering
Kasetsart University, Paholyothin Rd., Bangkok, Thailand.

Abstract

This paper describes our first large-scale retrieval attempt in TREC-7 using DSIR. DSIR is a vector space based retrieval system in which semantic similarity between words, documents and queries, is interpreted in terms of geometric proximity of vectors in a multi-dimensional space. A co-occurrence matrix computed directly from the collection is used to build the underlying semantic space. We have implemented DSIR on a cluster of low-cost PC Pentium-class machines, and chosen the PVM message-passing library to manage our distributed DSIR version. Although our first adhoc retrieval results are quite poor in terms of recall-precision measure, we believe that more work and experiments have to be explored in order to obtain more promising retrieval performance.

1 Introduction

For our first large-scale text retrieval attempt in TREC-7 adhoc experiments, we use our own retrieval artifact, called "DSIR", a full-text retrieval system developed on a cluster of PC Pentiums at the department of Computer Science, Kasetsart University[1]. DSIR stands for "Distributional Semantics based Information Retrieval", a retrieval model based on vector space. In this model, the contents of retrievable objects, such as words, phrases, sentences, documents, are represented in a unified way by multi-dimensional vectors. These vectors are derived from a co-occurrence matrix computed on textual collection being indexed. Semantic proximity among objects is then simply interpreted in terms of geometric proximity between corresponding vectors in the multi-dimensional space, called the "meaning space".

Former source codes of DSIR algorithm are written in C, and Perl programming languages, running on a standard Unix machine. To achieve TREC-7 experiments, we reexport the distributed DSIR to a cluster of low-cost PC Pentiums, running Linux operating system. Each PC is hooked together through a low-cost Ethernet local area network. New distributed version is developed using PVM[2] [2], a widely used message-passing software package.

We organize this paper in the following ways. Section 2 gives a brief overview of fundamental concepts constituting DSIR model. Section 3 provides more detail about distributed DSIR implementation. Section 4 presents TREC-7 retrieval experiments and gives the results. Finally, section 5 concludes this paper.

2 DSIR Model

2.1 Basic Concept

Research in distributional semantics concerns with the utilization of distributional information extracted from textual collections to represent the meaning of linguistic

[1] Indeed, this retrieval model has been devised during the author's Ph.D. study at ENST-Paris, in France [8].

[2] Parallel Virtual Machine.

entities, e.g. words, phrases, sentences, documents. We assume that there exists a correlation between meaning of a word and its observable distributional characteristics within particular contexts in a given language [6]. These distributional characteristics can either be "occurrences" of that word itself, or its "co-occurrences" with the other words appearing within the documents.

In this retrieval approach, we are especially interested in using word contexts to characterize the meaning of a word [9, 10, 7, 8]. In general, every word has meaning. Each contributes its own meaning, according to its occurrence, to the whole content of the document in which it appears. Here, we choose "word" as an elementary entity that holds the meaning. We consider tokens of length at least two characters, beginning with an alphabet, excluding those words in a pre-defined non-significant word list, as words (i.e. keywords or index terms) that constitute a set of vocabulary chosen for indexing a document collection. Following the "distributional structure" definition of Harris [3, 1]:

> "*The distribution of an element will be understood as the sum of all its environments.*",

we denote the "context" of a word as a knowledge concerning its usage, i.e. how that word is used with the other words in order to compose the content of a document. We characterize word contexts on the basis of "co-occurrence statistic". This choice is made because it is a source of distributional information that is easily extracted from a document collection.

We then define the co-occurrence statistic of a word as the number of times that word co-occurs with one of its neighbors within a pre-defined boundary. We denote this boundary, the "distributional environment". Possible distributional environments can be sentences, paragraphs, sections, whole documents, or windows of k words.

The definition of this distributional environment is essential in our retrieval model. It is used to delimit the scope of the contexts which are of interest. Co-occurrences measured within distributional environment defined by a sentence will let the "local" context information of words written in the documents to be observed. On the other hand, co-occurrences measured within the environment of a paragraph or the whole document will let the "global" context information of words to be examined. A window of k words can be used to extract the information between local and global contexts.

A specialized case of representing a word based on its contexts is that true synonyms will have identical contexts. Near-synonyms or related words will have just similar contexts. On the other hand, in case of a polysemy, its contexts are different because its meanings are in general invoked with different sets of words in different contexts. Representing the contents of documents on the basis of word contexts rather than just word occurrences thereby makes this retrieval model different from other standard keyword-based approaches. Documents in the collections should be retrieved without difficulty even if a query is composed of synonyms or related terms.

In our computational model, we use a co-occurrence matrix illustrated in Figure 1 to represent distributional information extracted from a document collection. Each row in this matrix represents the distribution of a word x_i, while each column represents the distribution of another word y_j which appears close to x_i. The intersection between row i and column j, i.e. the m_{ij}, records the co-occurrence frequency between x_i and y_j extracted from a document collection.

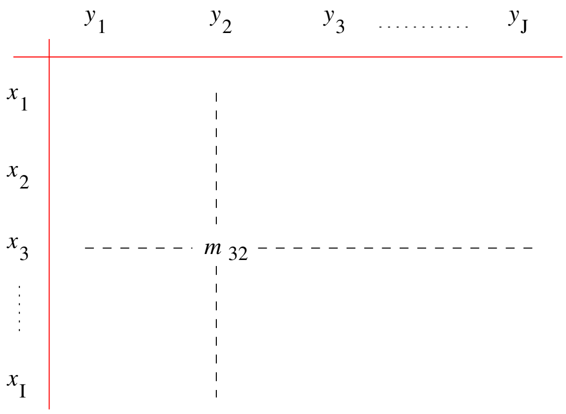

Figure 1: Co-occurrence matrix.

To represent meaning of a word according to its contexts by a vector, we depict each word distribution x_i corresponding to row i in the co-occurrence matrix by a vector $\vec{v}(x_i)$ using the sequence of $\{m_{ij} \mid j \in \mathrm{J}\}$ as its coordinate. Each dimension of this vector is associated to word y_j representing the column of the matrix. We hereafter call this vector, "co-occurrence vector".

Therefore, if a co-occurrence matrix built from a document collection consists of I rows representing I word distributions, and J columns representing J word distributions, the meaning of these I words can, by this way, be projected onto a vector space of J dimensions by I corresponding co-occurrence vectors. We name hereafter this vector space, "meaning space". Figure 2 is supposed to illustrate the first three vector representations corresponding to words x_1, x_2 and x_3 in the first three-dimensional meaning space associated with words y_1, y_2 and y_3, derived from a document collection.

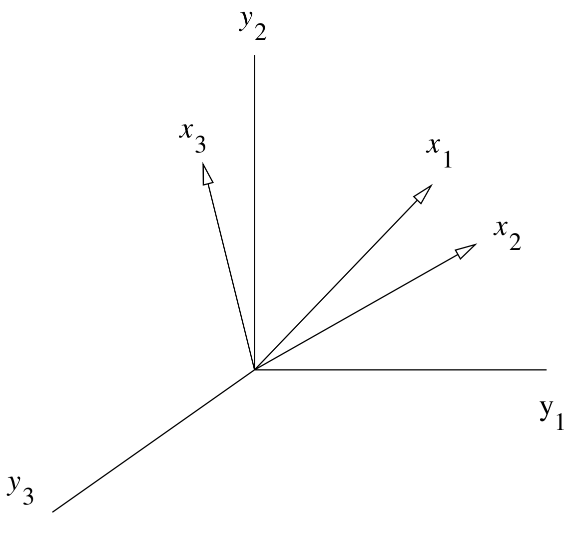

Figure 2: Vector representation of words in a meaning space.

2.2 Document Representation

A full-text document consists of words. Since we have already represented the meanings of words as vectors in the meaning space, our problem now is limited to define the vector representation of a document on the basis of the co-occurrence vectors of words of which that document is composed. We propose to define the vector representation of a document using the weighted vector sum of the co-occurrence vectors corresponding to words occurring in that document. Formally, if we choose I words, and J features, to index a collection of N documents, a document vector n is written by:

$$\vec{v}(d_n) = \Big(\sum_{i=1}^{\mathrm{I}} w(f_{ni}) m_{i1}, \sum_{i=1}^{\mathrm{I}} w(f_{ni}) m_{i2}, \sum_{i=1}^{\mathrm{I}} w(f_{ni}) m_{i3}, \dots, \sum_{i=1}^{\mathrm{I}} w(f_{ni}) m_{i\mathrm{J}} \Big) \tag{1}$$

where w(f_{ni}) is the weighting function addressing the importance of the word i in document n. Since a query can be considered as a specific document, its vector representation is derived in the same way as those of documents. Figure 3 below illustrates our document and query vector representations.

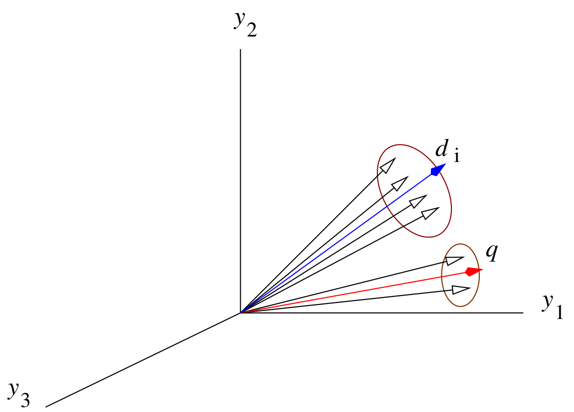

Figure 3: Document representation.

The document vector representation defined in Equation (1) can be seen as an approximation of semantic content of a document, because the (weighted) vector sum averages the direction of a set of vectors corresponding to words constituting that document. The intuition underlying this proposition is that a given document is composed of several words corresponding to different topics. If at least some of the words in a document are frequently used to described what the current topic is about then their corresponding co-occurrence vectors will pull the final vector sum towards the direction of that topic.

We also include the document vector components derived from the conventional vector space retrieval model [11] in our retrieval model. If we define $\vec{v}_{\text{DS}}$ as the component vector written in Equation (1), and $\vec{v}_{\text{VS}}$ as the component vector conventionally derived from the standard vector space method, our final document vector representation can be written as follows:

$$\vec{v}_{\text{DSIR}} = \mathcal{H} \,.\, \vec{v}_{\text{DS}} \;+\; (1-\mathcal{H}) \,.\, \vec{v}_{\text{VS}} \tag{2}$$

The $\mathcal{H}$ parameter, which we call "hybrid parameter", takes the real value between 0.0 and 1.0. When $\mathcal{H}$ is defined = 1.0, each document vector just takes the DS component vector. On the other hand, when $\mathcal{H}$ is defined = 0.0, each document vector is derived from conventional vector space retrieval model.

2.3 Document Retrieval

Since words, documents, and queries are represented as vectors in the same vector space, the basic retrieval operation in this retrieval model is then very simple; the query vector is compared to every document vector, and the documents whose vectors locate close to that query vector in the meaning space are presented to the user as relevant answer. These documents are returned in decreasing order of their closeness.

In addition, other similarity comparison can be obtained as well. For example, it is easy to combine traditional keyword matching method during any retrieval operation since each keyword also has a corresponding co-occurrence vector in the same meaning space. The user can first use his keyword(s) as query to filter for ranked documents which locate close to that keywords, and then select one (or several) of them as his new query to find the closest remaining documents of interest. In the same way, traditional relevance feedback [4] can easily be integrated into this retrieval model as well. During a retrieval process, the user can choose certain terms or documents (with weights) so that their vectors can simply be added up to the query vector to search more documents in the collection.

If we assume that there are chosen J distinct features for representing documents in a collection, a given document d_n can then be written as a J-dimensional vector of the form:

$$\vec{v}(d_n) = (d_{n1},\, d_{n2},\, d_{n3},\, \ldots\,,\, d_{n\text{J}}), \tag{3}$$

where d_{nj} represents the j^{th} element of document vector n. In the same way, a vector representation of a given query q can be written as a J-dimensional vector of the form:

$$\vec{v}(q) = (q_1,\, q_2,\, q_3,\, \ldots\,,\, q_{\text{J}}), \tag{4}$$

A typical vector similarity measure that we use in this retrieval model is the cosine similarity function. This function represents the cosine of the angle between vectors of query q and document d_n in a J-dimensional vector space, which is written by:

$$\texttt{sim}(\vec{v}(q), \vec{v}(d_n)) = \frac{\sum_{j=1}^{\text{J}} q_j \times d_{nj}}{\sqrt{\sum_{j=1}^{\text{J}} (q_j)^2 \times \sum_{j=1}^{\text{J}} (d_{nj})^2}} \tag{5}$$

3 DSIR Implementation

DSIR system consists of 4 parts; document preprocessing, co-occurrence matrix computation, document vector derivation, and document retrieval. We use a small Perl routine to arrange TREC adhoc collections to be ready for document preprocessing. Document preprocessing in DSIR integrates both Porter and Lovin stemmers, including standard stopword elimination.

Distributed-DSIR implementation uses master/slave or pool of tasks programming style on PVM platform [2]. During co-occurrence matrix computation, a big word-by-word co-occurrence matrix is partitioned into small portions, each can be fit in physical memory of n PC Pentium machines (see the machine configuration in Figure 4). Due to the fact that we have only one big local harddisk, each machine reads and performs co-occurrence computation on TREC collections via NFS[3]. A scheduler (or master), the machine at which the TREC collections is located, has responsible to manage the messages between machines in the pool. Finally, that scheduler accumulates all co-occurrent matrix portions from other machines and writes them out to its local disk.

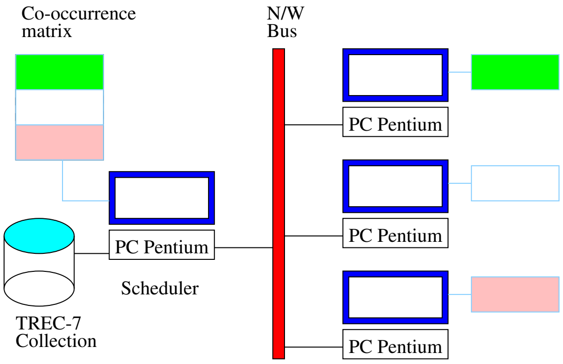

Figure 4: Distributed-DSIR machine configuration during co-occurrence matrix computation and document vector derivation.

[3]Read and write large amount of data via Network File System is one of the bottle-neck problems in our current distributed-DSIR implementation.

During document vector derivation, we still use the same machine configuration illustrated in Figure 4. The big co-occurrence matrix is partitioned, each small portion is distributed through other machine. The main scheduler then reads each TREC-7 document from the collections to see which co-occurrence vectors are needed to be retrieved from other machines to compose that document vector. A careful design of caching strategy during document vector derivation is necessary in order to reduce message passing between scheduler and other machine in the pool in order not to saturate the network bus.

Distributed document retrieval algorithm in DSIR is quite straightforward (see the machine configuration in Figure 5). Each machine in the pool, called "retrieval engine", reads portion of document vector into its main memory, and waits for retrieval command from the central scheduler. The central scheduler reads each query vector, and distributes it to every retrieval engine. Each retrieval engine retrieves and ranks its document vectors, the ones which are close to the query vector are ranked first, and send its ranking back to scheduler. Note that during this phase, retrieval engines perform their retrieval tasks in parallel. Then scheduler accumulates all rank lists from retrieval engines, and performs the final ranking scores.

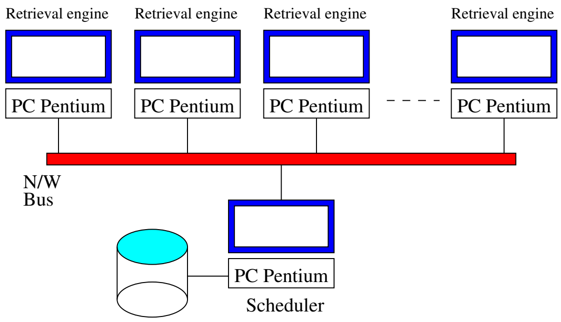

Figure 5: Distributed-DSIR machine configuration during document retrieval.

4 Experimental Results and Discussion

We participate quite lately our TREC-7 adhoc experiments, i.e. in May 1998. That means we have only 3 months before the official deadline to prepare and scale-up DSIR on our PC cluster to perform this large-scale text retrieval task. DSIR adopts standard SMART stoplist, and uses Lovin stemmer to pre-process all adhoc documents. For each run, two term sets are chosen by document and occurrence frequency criterion to build a co-occurrence matrix. We also applied the chi-square correcting weight, which we call "spatial transformation" [8], to every entry in the co-occurrence matrix. That means, each co-occurrence entry m_{ij} is transformed to $m_{ij}^{'} = (\frac{1}{r_i\sqrt{c_j}})\, m_{ij}$, where $r_i = \sum_j m_{ij}$ is defined as row total, and $c_j = \sum_i m_{ij}$ is defined as column total.

Document vectors are computed using formulae explained earlier in Equations (1) and (2). DSIR uses 'aaa.bbb' SMART weighting style [5] during indexing the documents. Query vectors are calculated in the same way, using words found in "title" and "description" field of the topics 351-400. We submitted two adhoc run; dsir07_a01 and dsir07_a02. Table 1 gives the values of different DSIR parameters, and results.

Run	Index	Features	$\mathcal{H}$	Weighting	Av. P
DSIR07_a01	11488	566	0.05	ntc.atc	0.0117
DSIR07_a02	15567	566	0.05	ntc.atc	0.0135

Table 1: Indexing parameters used in TREC-7, and results.

Considering our first official testing of DSIR over the TREC-7 collection set, we found that the recall/precision results were very bad. This level of performance is far below DSIR's typical performance tested over the old standard, very small size collections such as Cranfield and Time. We look forward to getting a more complete set of experiments and to spending more time understanding the situation in which DSIR had difficulty in identifying relevant documents. We believe that the word contexts that DSIR derived from the co-occurrence matrix built directly from the whole TREC-7 collections is confused by several domains that are specific to those TREC-7 collections themselves. DSIR should work better when derived word contexts are learned from a specific domain of interest. Thus, we plan to use DSIR to index and search TREC-7 collections separately so that each semantic space (i.e. meaning space) has been derived from word co-occurrences more correctly with respect to a certain domain.

5 Conclusion

In this paper, we introduce our DSIR retrieval system that has been tuned to perform large-scale text retrieval in TREC-7 adhoc track. DSIR is a distributional semantics based retrieval model in which semantic proximity is derived from a co-occurrence matrix calculated from the textual collection being indexed. Words, documents, and users' queries, are represented in a unified way by vectors in a multi-dimensional space. Retrieval is performed on the basis of the geometric proximity between vectors representing documents and the user's query; documents whose corresponding vectors are closed to that of query are returned as relevant answer.

To achieve TREC-7 adhoc experiments, we have reexported our distributed DSIR on a cluster of low-cost PC Pentium machines using PVM framework. Since the results of our first large-scale retrieval attempt in this TREC-7 are not quite successful, in terms of recall-precision measure, more work and experiments must be continued.

References

[1] J.P. Benzécri. Analyse Statistique des Données Linguistiques. In J.P. Benzécri & Collaborateur, editor, *Pratique de l'Analyse des Données*, volume 3 of *Linguistique & Lexicologie*. Dunod, Paris, 1981.

[2] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. PVM: Parallel Vir-

tual Machine–A Users' Guide and Tutorial for Networked Parallel Computing. The MIT Press, Cambridge, Massachusette, 1994.

[3] Z.S. Harris. Structure Distributionnelle. In M. Arrivé and J.C. Chevallier, editors, *Initiation à la linguistique: La grammaire*, Serie A3, pages 249–258. Klincksieck Linguistiques, Paris, 1975. French translation from original article of Z.S. Harris, "Distributional structure", in **Word**, number 2-3, 1954, by J. Throne and M. Arrivé.

[4] E. Ide. New Experiments in Relevance Feedback. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 16, pages 337–354. Prentice-Hall, Englewood Cliffs, 1971.

[5] J.H. Lee. Combining Multiple Evidence from Different Properties of Weighting Schemes. In E.A. Fox, editor, *Proceedings of the* 18^{th} *Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, Seattle, Washington, USA, July 1995.

[6] M. Rajman and A. Bonnet. New Tools for Text Analysis: Corpora-based Linguistics. In *The First Conference of the Association for Global Strategic Information*, Bad Kreuznach, Germany, November 1992.

[7] M. Rajman and A. Rungsawang. How to find the nearest by evaluating only few? Clusterization techniques used to improve the efficiency of an Information Retrieval based on Distributional Semantics. In *The Fifth Conference of International Federation of Classification Societies (IFCS-96)*, volume 1, Kyoto, Japan, March 1996.

[8] A. Rungsawang. Distributional Semantic based Information Retrieval. PhD thesis, ENST-Paris, Department of Computer Science, Paris, France, 1997.

[9] A. Rungsawang and M. Rajman. A New Approach for Textual Information Retrieval. In E.A. Fox, P. Ingwersen, and R. Fidel, editors, *Proceedings of the* 18^{th} *Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, in Poster session, Seattle Washington, USA, July 1995.

[10] A. Rungsawang and M. Rajman. Textual Information Retrieval Based on the Concept of the Distributional Semantics. In *Proceedings of the* 3^{th} *International Conference on Statistical Analysis of Textual Data*, Rome, Italy, December 1995.

[11] G. Salton, editor. The SMART RETRIEVAL SYSTEM, Experiments in Automatic Document Processing. Prentice-Hall, Englewood Cliffs, New Jersey, 1971.