## 1. Summary

This paper focuses on automatic indexing and retrieval. It applies latent semantic analysis with singular value decomposition (SVD) on a term-by-document matrix (two-mode factor analysis). In this way, associated terms and documents are placed closely, and the space could be arranged to keep only major associate patterns for efficiency. Queries are viewed as pseudo documents to which returned documents are of high similarities.

There are 2 sides of difficulties people face: synonymy (different words, similar meaning) and polysemy (same word, different meanings), which negatively affect recall and precision of retrieval performance, respectively. These problems result from 1) incomplete identification of index terms, 2) lack of automatic method for polysemy, and 3) oversimplified treatment of word type (not considering redundancy).

In contrast to term-matching retrieval, this paper assumes that there exits latent semantic structure in document data (with random obscuring). With criteria of 1) adjustable representational richness, 2) explicit representation of both terms and documents, and 3) computational tractability for large datasets, the authors choose two-mode factor analysis among many candidates such as tree unfolding model and nonmetric multidimensional unfolding.

The basis of two-fold models, SVD, has strong correlations with eigen-analysis for one-fold models. Eigen-analysis takes one type of object, break down into *linearly independent* components (factors) with which *similarity* of original objects could be compute with *major* ones. SVD operates on an arbitrary matrix of *different* types of objects (terms and documents), orthogonal factors are derived, and dot product between vectors still represent object similarity. By keeping only *k* largest factors, term/query/document can be expressed by *k* factors (location in *k*-space defined by them). In order to process a query, it is viewed as a pseudo document so that similarity to it could be measured for each document. And those document with highest similarities are returned as results. In the section of Technical Details, formalized model is discussed along with methods for finding representations for pseudo documents and computing fundamental comparison quantities such as 1) two terms, 2) two documents, and 3) a term and a document.

Finally, the model is tested on two standard documents collections with method of straightforward term matching (SMART) and vector (Voorhees). Without refinements, the model still achieves “modestly encouraging” results. Format adjustment and incorporation of highly informative words might further improve performances.

## 2. Comments and Questions

With latent semantic analysis in the paper, although the synonymy problem is well treated, the *polysemy* problem is just *partially* solved. According to the paper, the meaning of a word could be conditioned not only by other words in the document and also by “other appropriate words in the query not used by the author of a particular relevant document”. By combining information from both the query and the document, LSI is able to understand the specific meaning of a given word. On the other hand, since each word (even with multiple meanings) corresponds to exactly one point in the *k*-space, it is represented as a weighted average of different meanings. Thus, if a real meaning is far from the average meaning, queries for this meaning is possible to be distorted and returned actually irrelevant documents.

For synonymy and polysemy problems, is it kind of analogous to terms of low and high document-frequency (respectively) in Week 2's paper, given both synonymy problem and using low-frequency terms as discriminators decrease recall, while both polysemy problem and using high-frequency terms as discriminators harm precision? If so, is there any underlying factor for such an analogy? For example, both high-frequency and multiple-meaning terms are able to confuse retriever to consider irrelevant documents with containing this term as relevant.

Moreover, I am curious about how to decide the number of factors the model should use. It seems that it will be computationally expensive if the number is too large while giving far from reasonable results if too small. Moreover, how will the factor number be affected by characteristic of document collections such as size of document collection, the average size of document and the word frequency of lexicon?