Venturing into Vector Space: Exploring Applications of Latent Semantic Analysis

Gautam Mittal
Analysis Honors
Henry M. Gunn High School
Palo Alto, CA 94306
gautam@mittal.net

1 The Vector Space Model

The field of information retrieval is primarily concerned with comparing a set of documents containing natural, unstructured language. Although the task of comparing documents to one another may be elementary for a human, computers do not have complex language processing cortices and have trouble processing information that cannot be modeled mathematically. The vector space model aims to solve the issue of representing abstract, unstructured natural language as a mathematically-operable entity that computers can process.

1.1 Constructing Document Vector Representations

In order to construct a vector space representation of a document, there needs to be a larger set of documents which this representation will be made in relation to. If there are n documents, the unique words used across all of the documents will be used to represent each dimension of the vector. The components of this vector will be the frequencies of each word in the document that will be represented as a vector.

For example, if there are two documents containing the text "vector vector" and "model vector", the corresponding vectors would be <2, 0> and <1, 1>, respectively. This is because the two unique words used across both documents were "vector" and "model", where "vector" represented the first dimension and "model" represented the second dimension. Although this was only a two-dimensional example, the vector space model can be easily scaled up to n-dimensions, with an any number of documents with n unique words.

1.2 Comparing Document Vectors in Space

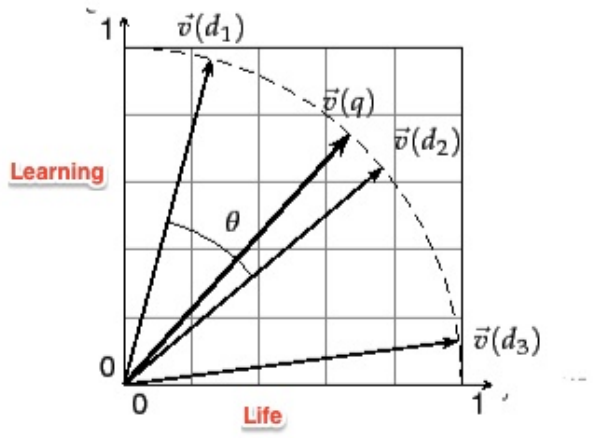
Now that the process for representing abstract text documents as vectors has been defined, a function or algorithm for comparing documents that returns a scalar "similarity" score must be established.

The **dot product** is an operation that takes two vectors and returns a scalar value, defined by the following:

$$a \cdot b = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + ... + a_n b_n \tag{1}$$

However, this is not a very good measure of vector similarity, because for vectors a = <1, 1>, b = <2, 0>, and c = <1, 0>, $a \cdot a = 2$, while $b \cdot c = 2$. When a is operated upon itself, the similarity score is 2. Therefore, we can assume that if a vector is compared to itself, the similarity score should be 2, but $b \cdot c$ disproves this principle because b and c are different vectors.

The much better approach is to find a method that returns a function of the angle between two vectors.



$$cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} \tag{2}$$

This similarity score is based on a vector's position in n-dimensional space and is fairly intuitive; vectors that are very similar will have a relatively small angle between them, while very different vectors will have a relatively large angle between them. Furthermore, by taking the cosine of this angle, the range of the similarity score will be [0, 1]. Vectors that have a cosine similarity close to 1 will be very similar (parallel in space) and vectors that have a cosine similarity close to 0 are very different (orthogonal). If documents can be represented as vectors, and vectors can be compared using their cosine similarity, then abstract documents can effectively be compared using this intuition. Cosine similarity is at the core of all primitive search engines, as it allows for very simple yet effective document comparison that scales well for any set of documents. The function for cosine similarity is defined as shown on the left.

Using these principles, we can formulate the basis for many basic applications involving information retrieval. However, there are some limitations to using cosine similarity alone.

1.3 Limitations of Sparse Vectors

Sparse vectors, or vectors that have lots of zero-valued entries, are not uncommon when building document vector representations because, obviously, not every document will contain every word in the English language. One of the primary issues with representing documents as sparse vectors with dimensions corresponding to unique words is dealing with synonymy. For example, if a document contains the word *car*, and a search query represented as its own document vector contains the word *automobile*, then the cosine similarity of the documents will be different, when the words actually are referring to the same thing. This is a limitation of using terms explicitly as dimensions of document vectors and relying on term counts alone.

1.4 Singular Value Decomposition and Dense Vectors

One way to mitigate the issue of synonymy is to find a more generalized representation of a vector that still maintains the same relation in space with regards to other document vectors in the set of documents. This kind of dense, generalized vector is called an **embedding**.

Embeddings rely on the intuition that words are no longer explicitly defined, but appear in vector space based on their document "topic" or "context". This context-based vector is based on the idea that the components and dimensions are no longer defined as representations of words, but as embeddings in space related to contexts of one another. Another way to think about this is to think of the word *mačka*, a word referred to in a document as a four-legged creature that purrs when pet. Without knowing what *mačka* means most people would identify a *mačka* as a cat. That is because of the words surrounding *mačka* provide enough context to infer that a *mačka* is likely a cat. Likewise,

embeddings break vectors down into their respective "context" vectors, therefore allowing words such as cat and *mačka* to be represented in a similar manner in space.

The mathematical tool used for creating dense vector embeddings is called **singular value decomposition (SVD)**, which states that any $M \times N$ matrix A can be factored into three constituent matrices, U $(M \times M)$, $\sum$ $(M \times N)$, and V^T $(N \times N)$. Matrix U is a matrix whose columns are the eigenvectors of $A \cdot A^T$, Matrix $\sum$ is a diagonal matrix populated with the eigenvalues of $A^T \cdot A$ and $A \cdot A^T$, and Matrix V^T is a matrix whose columns are the eigenvectors of $A^T \cdot A$. Although the computation of the singular value decomposition is fairly straightforward, the proof for SVD is beyond the scope of this paper.

A set of documents can be represented as **term-document matrix**, where each column of the matrix is a document vector. In an $M \times N$ rectangular term-document matrix, there are M document vector dimensions, and N documents. Because we want to retain the same number of documents, we use what is known as the **context-document matrix**, or the V^T component instead of our term-document matrix because it has N column document vectors and has dense embeddings instead of raw term-count vectors to represent documents.

The process of using SVD to turn a raw term-document matrix of document vectors into a context-document matrix of document embedding vectors is known as **latent semantic analysis**, a technique used for a variety of applications.

2 Applications

There are many practical applications for the vector space model combined with latent semantic analysis. Because of the mathematical basis for representing text-based documents as vectors, computers can perform many complex operations to yield a variety of results.

2.1 Document Search Engines

One of the most common applications of the vector space model is searching through a set of documents or web pages. The most primitive of search engine simply turns text documents into their vector representations and converts the user search query into its own vector representation and then compares the query vector to all of the document vectors using cosine similarity. Documents that have a larger cosine similarity are closer in vector space while documents have a lower cosine similarity are further apart in vector space, and search rankings are organized accordingly. However, as discussed earlier, using raw document vector representations alone causes issues with word synonymy because the document vectors are sparse. To fix this problem, latent semantic analysis can be applied to the set of document and query vectors to account for synonymy and return search results based on document context and not based on explicit word frequencies alone.

2.2 Essay Evaluation

MAINFRAMES

Mainframes are primarily referred to large computers with rapid, advanced processing capabilities that can execute and perform tasks equivalent to many Personal Computers (PCs) machines networked together. It is characterized with high quantity Random Access Memory (RAM), very large secondary storage devices, and high-speed processors to cater for the needs of the computers under its service.

Consisting of advanced components, mainframes have the capability of running multiple large applications required by many and most enterprises and organizations. This is one of its advantages. Mainframes are also suitable to cater for those applications (programs) or files that are of very high

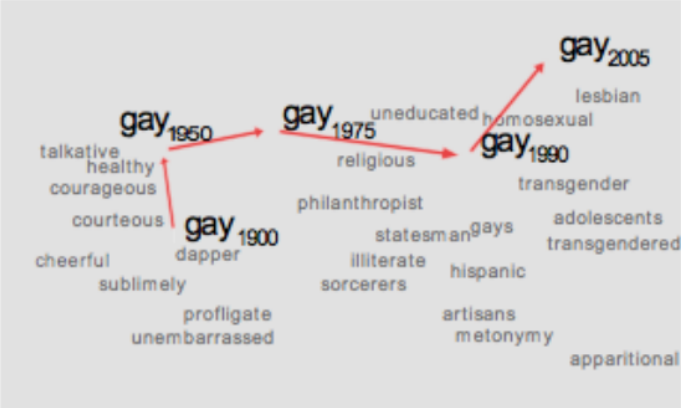
MAINFRAMES

Mainframes usually are referred those computers with fast, advanced processing capabilities that could perform by itself tasks that may require a lot of Personal Computers (PC) Machines. Usually mainframes would have lots of RAMs, very large secondary storage devices, and very fast processors to cater for the needs of those computers under its service.

Due to the advanced components mainframes have, these computers have the capability of running multiple large applications required by most enterprises, which is one of its advantage. Mainframes are also suitable to cater for those applications or files that are of very large demand

Another common application of vector space representations is essay grading and evaluation. Although the computer cannot glean information directly from the content of the essay, such as level of analytical thinking or quality of citations, etc., using vector representations can allow the computer to derive certain attributes of the essay that normally only a human could process. One of the key attributes that vector space models can extract from a set of two essays is their plagiarism score. Using latent semantic analysis and their respective vector representations, the context and approximate term similarity can be gleaned using cosine similarity. If two essay vectors have very similar positions in space, a very high plagiarism score will be given. This application is especially useful because the computer does not know anything about the essay content's meaning nor does it have a concept of grammar, yet it still is able to yield fairly human-like results when it comes to solving these language-based tasks.

2.3 Historical Word Analysis



In addition, the vector space model along with latent semantic analysis can allow us to understand the historical meanings of words based on the context of the documents within which they appear.

For example, the word "clue" referred to a ball of yarn in the mid-late seventeenth century, while the more contemporary meaning refers to a bit of evidence used to solve a mystery. A simple intuition to derive a word's intended meaning is to look at the words surrounding it in a document, or its context. Because latent semantic analysis can derive document context through generating vector embeddings from document vectors using singular value decomposition, finding the semantic shift of words over time is not a very difficult problem to solve. If we have a variety of documents from a certain period in history, take the seventeenth century for example, that involve the word "clue", a search engine will shift its semantic representation of the word in vector space to find documents contemporary documents that have context related to yarn or thread rather than returning documents related to solving mysteries. Thus, when a search query contains contexts related to yarn, documents containing "clue" will be returned. Because of the latent context of a set of documents from a certain period in history, the semantic shift of a word can be better understood based on the nature of the search results returned.

References

[1] Austin, David. "We Recommend a Singular Value Decomposition." American Mathematical Society. American Mathematical Society, n.d. Web. <ams.org/samplings/feature-column/fcarc-svd>.

[2] Babu, Pavithra, Sargur Srihari, Jim Collins, Harish Srinivasan, and Rohini Srihari. "Automatic Scoring of Automatic Scoring of Handwritten Essays Using Latent Handwritten Essays Using Latent Semantic Analysis Semantic Analysis." (n.d.): n. pag. Center of Excellence for Document Analysis and Recognition (CEDAR) Department of Computer Science and Engineering. University at Buffalo, State University of New York. Web. <http://www.cedar.buffalo.edu/ srihari/talks/DAS-Presentation.pdf>.

[3] "How Can You Explain the Singular Value Decomposition to Non-specialists?" Stack Exchange Mathematics. Stack Exchange, n.d. Web. <math.stackexchange.com/questions/261801/how-can-you-explain-the-singular-value-decomposition-to-non-specialists/261814>.

[4] "Is It True That Any Matrix Can Be Decomposed into Product of Rotation,reflection,shear,scaling and Projection Matrices?" Stack Exchange Mathematics. Stack Exchange, n.d. Web. 31 Mar. 2017. <http://math.stackexchange.com/questions/109108/is-it-true-that-any-matrix-can-be-decomposed-into-product-of-rotation-reflection>.

[5] Jurafsky, Dan. "Vector Semantics." Stanford University, Stanford, CA. Lecture.

[6] Jurafsky, Daniel, and James H. Martin. "Semantics with Dense Vectors." (2016): n. pag. Stanford University, 7 Nov. 2016. Web. <https://web.stanford.edu/ jurafsky/slp3/16.pdf>.

[7] Kalman, Dan. A Singularly Valuable Decomposition: The SVD of a Matrix. DataJobs. The American University, 13 Feb. 2002. Web. <https://datajobs.com/data-science-repo/SVD-[Dan-Kalman].pdf>.

[8] Khan, Sal. "Introduction to Eigenvalues and Eigenvectors." Khan Academy. Khan Academy, n.d. Web. 31 Mar. 2017. <https://www.khanacademy.org/math/linear-algebra/alternate-bases/eigen-everything/v/linear-algebra-introduction-to-eigenvalues-and-eigenvectors>.

[9] "Latent Semantic Analysis." Wikipedia. Wikimedia Foundation, 25 Mar. 2017. Web. 31 Mar. 2017. <https://en.wikipedia.org/wiki/Latent$_s emantic_a nalysis >$.

[10] "Latent Semantic Indexing (LSI) An Example." (n.d.): n. pag. Department of Systems Engineering and Engineering Management. The Chinese University of Hong Kong. Web. <http://www1.se.cuhk.edu.hk/ seem5680/lecture/LSI-Eg.pdf>.

[11] Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. "Latent Semantic Indexing." Latent Semantic Indexing. Cambridge University Press, 7 Apr. 2009. Web. 31 Mar. 2017. <http://nlp.stanford.edu/IR-book/html/htmledition/latent-semantic-indexing-1.html>.

[12] Manning, Christopher. "Matrix Decompositions and Latent Semantic Indexing." Introduction to Information Retrieval. N.p.: Cambridge UP, 2009. N. pag. Print.

[13] "Principal Component Analysis." Wikipedia. Wikimedia Foundation, 31 Mar. 2017. Web. 31 Mar. 2017. <https://en.wikipedia.org/wiki/Principal$_c omponent_a nalysis >$.

[14] "What Are Some Simple and Advanced Applications of Latent Semantic Analysis (LSA)?" Quora. Quora, n.d. Web. <https://www.quora.com/What-are-some-simple-and-advanced-applications-of-Latent-Semantic-Analysis-LSA>.

[15] "What Is a Good Explanation of Latent Semantic Indexing (LSI)?" Quora. Quora, n.d. Web. 31 Mar. 2017. <https://www.quora.com/What-is-a-good-explanation-of-Latent-Semantic-Indexing-LSI>.