# Course Recommendation System Using Document Clustering

**Victor Amaral**
Department of Electrical & Computer Engineering
Johns Hopkins University
Baltimore, MD 21218
vamaral1@jhu.edu

**Chandler Furman**
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
gfurman2@jhu.edu

## Abstract

We have implemented a course recommendation system using document clustering methods. We apply a stop word list and a tf-idf weighing scheme to create vectors of words comprising any part of the course description. We analyze several k-means variants to solve the problem and conclude that results are most accurate when a cosine similarity function is used.

## 1 Introduction

Course selection for students at universities often relies on an advisor or other students to suggest courses they think will be suitable. This is due to the fact that there is not an outside resource where a student can search courses by their course descriptions. Rather, they have to know specific information about the course such as the department name or the name of the instructor. This makes it difficult for the student to discover a course they didn't know existed. The purpose of this project is to recommend courses similar to a course that the student may have enjoyed.

We apply Information Retrieval concepts for preprocessing and and reading in data, Machine Learning algorithms for clustering of the data. For preprocessing, we experiment with a simple "handpicked" stop word list, as well as a tf-idf weighing scheme. In terms of clustering algorithms, we experiment with $\lambda$-means, spherical k-means, a k-means variant based on the cosine similarity, and bisecting k-means.

## 2 Methods and Algorithms

### 2.1 Information Retrieval

The first pre-processing technique we used was a stop word list. A stop word list is often used in information retrieval to ignore frequently occurring words that do not significantly contribute to the meaning of a particular document [7]. The general strategy for implementing a stop-word list is as follows: count the number of times each word appears in the corpus, sort the frequencies from increasing to decreasing order, take the most frequent terms and add them to the stop list.

Once the stop list was constructed we applied tf-idf to determine the weights of each word in the document vector. Tf-idf, short for term frequency-inverse document frequency, determines the relative frequency of words in a specific document compared to the frequency of that word over all of the documents [2]. In this context, we use tf-idf to determine the relative frequency of a word in a course instance with respect to the set of all available courses. This provides a way to measure the importance of a word in a course description. Given the set of available courses, $X$, a word $w$, and a course instance, $\mathbf{x} \in X$, we calculate:

$$w_d = f_{w,\mathbf{x}} \frac{\log(|X|/f_{w,X})}{|\mathbf{x}|} \tag{1}$$

where $f_{w,\mathbf{x}}$ is the frequency of word $w$ in course instance $\mathbf{x}$, $|X|$ is the number of courses in the course listings, $|\mathbf{x}|$ is the number of words in an instance(used for normalization), and $f_{w,C}$ is the frequency of word $w$ in the course listings.

## 2.2 Clustering

### $\lambda$ - Means

The first clustering method we run on our dataset is $\lambda$-means.[1] This algorithm assigns a course instance $\mathbf{x}$ to a cluster $k$ such that the Euclidean distance to each cluster is minimized. The default lambda value is based on the average distance of each training instance to the mean of the training data. Additionally, a threshold distance, $\lambda$, is set so that if the Euclidean distance of an instance $\mathbf{x}$ to every cluster is greater than $\lambda$, a new cluster is added with the cluster center, $\mu_{K+1}$, set as $\mathbf{x}$. The $\lambda$-means uses an EM-style iterative algorithm such that on each iteration, the algorithm computes 1 E-step and 1 M-step.

In the E-step, cluster assignments $r_{ck}$ are determined based on the current model parameters $\mu_k$. $r_{ck}$ is an indicator variable that is 1 if instance $\mathbf{x}$ belongs to cluster $k$ and 0 otherwise.

Suppose there are currently $K$ clusters. The indicator variables are set as follows. For $k \in \{1, \ldots, K\}$:

$$r_{ck} = \begin{cases} 1 \text{ if } k = \arg\min_j d(\mathbf{x}_i, \mu_j)^2 \leq \lambda \\ 0 \quad \text{otherwise} \end{cases} \tag{2}$$

where

$$d(\mathbf{x}_i, \mu_j) = \sqrt{\sum_{n=1}^{M} (\mathbf{x}_i^{(n)} - \mu_j^{(n)})^2} \tag{3}$$

which is the Euclidean distance where $n$ is the $n$th component of the vector. When computing this distance, we assume that if a word is not present in a course instance $\mathbf{x}_i$, then it has value 0.

If the instance $\mathbf{x}_i$ is assigned to a new cluster, $K+1$, the indicator for this is:

$$r_{c,K+1} = \begin{cases} 1 \text{ if } \min_j d(\mathbf{x}_i, \mu_j)^2 > \lambda \\ 0 \quad \text{otherwise} \end{cases} \tag{4}$$

If $r_{c,K+1} = 1$, update $\mu_{K+1}$ by setting $\mu_{K+1} = \mathbf{x}_i$ and $K = K + 1$.

After the E-step, the mean vectors $\mu_k$ are updated for each cluster $k \in \{1, \ldots, K\}$ (where $K$ may have increased during the E-step). This forms the M-step, in which the model parameters $\mu_k$ are updated using the new cluster assignments:

$$\mu_k = \frac{\sum_c r_{ck} \mathbf{x}_i}{\sum_c r_{ck}} \tag{5}$$

This process will be repeated for a given number of iterations.

### Stochastic K-Means Algorithm (SKA)

The second clustering algorithm used is a stochastic k-means algorithm. SKA assigns the data point $\mathbf{x}_i$ to a cluster based on the minimum Euclidean distance between the data point and the cluster centers $\mu_k$. However, rather than making a full pass through the data before updating the cluster centers $\mu_k$, they are updated after each course instance. In this algorithm, we determine the number of clusters $k$ based on the Bayesian Information Criterion which will be described later.

We define an iteration to be a single pass through the entire dataset. An iteration is composed of many steps, where each step corresponds to clustering a single example. Letting $c$ be the number of available courses, $c$ steps will be taken in an iteration.

On each step of training, $\mathbf{x}_i$ is assigned to the closest cluster, which sets the indicator variable for the $K$ clusters as:

$$r_{ck} = \begin{cases} 1 \text{ if } k = \arg\min_j d(\mathbf{x}_i, \mu_j) \\ 0 \quad \text{otherwise} \end{cases} \tag{6}$$

where $d(\mathbf{x}_i, \mu_j)$ is the Euclidean distance (3) and $r_{ck}$ is initialized to zero. If a cluster becomes empty during learning, its mean is set to zero. After each iteration, $\mu_k$ is updated and we move on to the next course instance.

We determine the number of clusters to initialize based on the Bayesian Information Criterion (BIC). BIC is a statistical measure of how well a given model predicts the data. It does so by penalizing the complexity of the model based on its data likelihood function [5]. The complexity is defined as the number of parameters in the model. For our case, the models correspond to different sets of cluster centers of size $k$. The Gaussian distribution is used as the underlying model because k-means clusters in

[1] Description of $\lambda$-means and SKA are adapted from CS475 Homework 4 created by Prof. Mark Dredze

a spherical shape that is similar to a Gaussian, so the parameters are then the mean $m$ and the variance $\sigma^2$. The user inputs a range of values for $k$ and the algorithms gives which value of $k$ is best suitable for the data as well as the actual cluster centers $\{\mu_1, ..., \mu_k\}$. Let $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_c, \}$ be the set of available courses, $C = \{\mu_1, \mu_2, ..., \mu_k\}$ be the set of cluster centers. The BIC is defined as:

$$BIC = L(\Theta) - \frac{1}{2} klog(c) \tag{7}$$

where $L(\Theta)$ is the log likelihood function according to each model, $k$ is the number of clusters, and $c$ is the number of available courses.

**Spherical K-Means**

The third clustering algorithm used is called spherical k-means. This version of k-means uses the cosine similarity which tends to work better for document clustering [3].

The spherical k-means algorithm uses the cosine of the angle between document vectors as the similarity function. This means that given document A and document B, the similarity function will return a number from 0 to 1 indicating whether A and B are dissimilar or similar, respectively. It is computed by taking the dot product of two vectors and dividing it by the product of their magnitudes. The cosine similarity function has been proven to be more efficient for document clustering due to the sparsity in document vectors. The main difference from standard k-means is that we give more importance to the direction of the document vector rather than the magnitude. In fact, each document vector is normalized to be unit length. Additionally, the underlying probabilistic models are not Gaussians [4].

Let $c$ be the number of available courses, $W$ be the number of words in the dataset, and $X = \{\mathbf{x}_1, \mathbf{x}_2, ...., \mathbf{x}_c\}$ be the course instances where each $\mathbf{x}_i \in \mathbb{R}^c$ and $||\mathbf{x}_i||_2 = 1$. We initialize the clusters $\mu_1 ... \mu_k$ and compute the cosine similarity of $\mathbf{x}_i$ with every cluster $\mu_k$ and assign it to the one with the largest value. More formally, let $s(x)$ denote the cosine similarity function defined by:

$$s(\mathbf{x}_i, \mu_k) = \frac{\sum_{i=1}^{W} \mathbf{x}_i^T \mu_k}{\|\mathbf{x}_i\| \|\mu_k\|} \tag{8}$$

$$r_{ck} = \begin{cases} 1 \text{ if } k = \arg\max_j s(\mathbf{x}_i, \mu_j) \\ 0 \quad \text{otherwise} \end{cases} \tag{9}$$

**Bisecting K-Means**

Bisecting k-means is a hierarchical clustering algorithm, meaning it does not require the number of clusters to be specified. Additionally, it is a divisive hierarchical clustering technique due to its recursive nature [6]. The algorithm takes in the dataset and recursively divides it by two. Given the set of courses $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_c, \}$, the data is partitioned into two sets $X_a$ and $X_b$ such that $X_a \cup X_b = X$ where the cluster center of a subset $X_i$ with size $N_i$ is defined by

$$\mathbf{w}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbf{x}_j$$

The classic bisecting k-means algorithm is as follows: First, randomly select a point $\mathbf{r_a} \in \mathbb{R}^\mathbf{n}$, compute the cluster center $\mathbf{w}$ of $X$, and $\mathbf{r}_b = \mathbf{w} - (\mathbf{r}_a - \mathbf{w})$. Next divide $X$ into two sub-clusters $X_a$ and $X_b$ according to the following rule:

$$\mathbf{x}_i \in \begin{cases} X_a \text{ if } \|\mathbf{x}_i - \mathbf{r}_a\| \leqslant \|\mathbf{x}_i - \mathbf{r}_b\| \\ X_b \quad \text{otherwise} \end{cases} \tag{10}$$

and compute the cluster centers of $X_a$ and $X_b$, $\mathbf{w}_a$ and $\mathbf{w}_b$. If $\mathbf{w}_a = \mathbf{r}_a$ and $\mathbf{w}_b = \mathbf{r}_b$, stop, otherwise recurse on the largest subset of $X$.

The classic bisecting k-means randomly picks a point in $\mathbb{R}^n$, but we chose to instead randomly pick two points from $X_a$ and $X_b$, cluster the data around those two points, and then recursively split the largest cluster for a certain number of iterations.

## 3 Implementation

### 3.1 Data Set Characteristics

When we began this project, we wanted to use course listings from Johns Hopkins University for our dataset. Unfortunately, the way it was formatted and its lack of accessibility made it unfeasible. We decided to find another university that had their course listings, and selected MIT because it was in a manageable format and did not require complicated web crawlers.[2] We then wrote a parser in Perl that removed unnecessary HTML blocks, took the

[2] Dataset came from http://web.mit.edu/catalog/subjects.html

remaining raw HTML, and separated it by courses getting rid of the remaining HTML. We then implemented a parser in java to separate the data in a more organized manner, so each course was broken up by different types of Features (see Table 1). Finally, some further pre-processing was done to remove punctuation and set every word to lower case.

In terms of content, this data set has 2233 courses and almost 12,000 unique words. Each course description has anywhere from 20 to 200 unique words. Furthermore, MIT is organized into 42 departments, which support anywhere from 20-150 courses each. We felt that this was generally reflective of JHU, and fit the needs of our project quite well.

### 3.2 Implementation Details

For the implementation of the clustering algorithms, we kept the basic skeleton of the cs475 package we used all semester for the homework assignments. Some major changes we made were in the DataReader so that we could support a FeatureVector mapping Strings to Doubles, as well as change the weights of our input data at runtime. Two examples of this are TF-IDF weighting, and stoplists, either of which we can turn on and off through command-line arguments. Furthermore, we also support different weightings of fields; for instance, if a student enjoyed a particular instructor, we can weight that field more heavily so that other courses that instructor teaches tend to show up more in our results.

We used our existing $\lambda$-means and SKA algorithms to analyze how they would perform in a document clustering setting. In addition, we wrote the code for the bisecting k-means algorithms, the BIC, and the spherical k-means algorithm.

The part that took up the most time was researching about document clustering techniques and figuring out which techniques we would apply and how we would do so. At times we tried to incorporate some ideas together but had issues. An example of this was that we wanted to use BIC with spherical k-means using a Gaussian model, but we quickly learned that the clusters are not based on a Gaussian-like spheres due to the cosine similarity.

| Feature | Type | Weight |
|---|---|---|
| Department | String | 2.0 |
| Course Number | Integer | 0.5 |
| Course Name | String | 2.0 |
| Pre-requisite | String | 2.0 |
| Description | String | 1.0 |
| Instructor | String | 3.0 |

Table 1: A list of different types of features. If a word falls into any of these categories, it is weighted according to the scheme in the right column.

## 4 Results

### 4.1 Methods

For each of these results, we chose to analyze an arbitrary class in the Chemical Engineering department, 10.02 (Philosophical History of Energy), which is also co-listed in the department of philosophy as 24.114. We felt this course was good because it would lend itself to interdepartmental interests. Furthermore, we felt our results generalized well to other courses, and in the interest of simplicity, decided that the most compelling way to present our results would be to analyze a single class.

To give our algorithms a base line, we use a similarity function that is popular in the Information Retrieval world, cosine similarity. It is important to note that this similarity measure is not the end-all authority on similarity measures. Rather, it gives us a base to compare our algorithms to existing methods.

Finally, please note that all of our data was collected using a stoplist and TF-IDF weighting, as well as the weighting scheme shown in Table 1. We tuned these parameters as we were implementing them to ensure they are helping us produce relevant result. However, for data collection, we decided to keep these things constant in order to focus on the algorithmic aspects of our implementation.

### 4.2 Results

First, we present the results of running our data set through cosine similarity. Table 2 is meant to give the reader a sense for what ”similarity” entails, and what a similarity measure might mean. These are the ten courses that are most similar to 10.02, Philosophical History of Energy.

| Sim | Course Title | Dept |
|---|---|---|
| 0.724 | Philosophical History of Energy | 24 |
| 0.399 | Separation processes | 10 |
| 0.368 | Numerical Methods Applied... | 10 |
| 0.368 | Integrated Chem. Eng. Topics II | 10 |
| 0.363 | Introduction to Chem. Eng. | 10 |
| 0.298 | Ethics for Engineers | 10 |
| 0.288 | Atmospheric Physics and Chem. | 10 |
| 0.287 | Chem. Eng. Projects Lab | 10 |
| 0.274 | Molecular, Cellular, and ... | 10 |
| 0.272 | Protein Folding and ... Disease | 10 |

Table 2: A list of courses that are most similar to 10.02, Philisophical History of Energy, by measure of Cosine Similarity with a customized weighting scheme. This course is listed in department "10", which is the Chemical Engineering department at MIT.

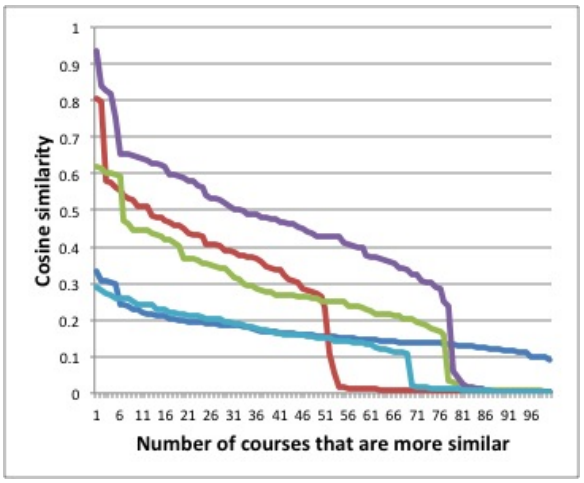


Figure 1: For five arbitrary courses, this figure shows the cosine similarity of the 100 most similar courses

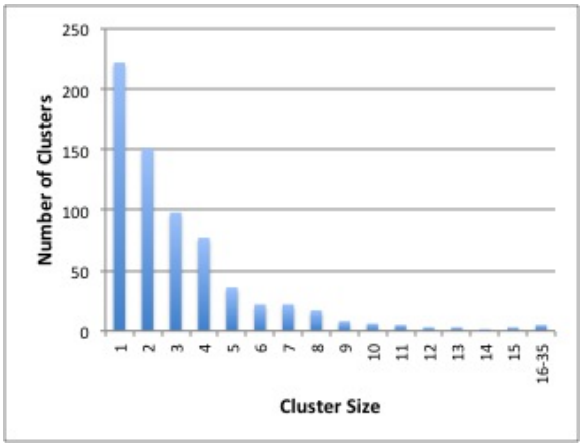


Figure 2: Frequency of cluster sizes resulting from a Lambda-Means trial

Figure 1 shows how similarity decreases as courses get more disparate. We observe that when the similarity drops below about 0.15, it indicates that we are now comparing the current course to courses outside of the department. For example, the purple line (which begins at y=0.93) is for a class in the Chemical Engineering department. There are 81 classes whose similarity to that class is greater than 0.15. Not coincidentally, this is about how many courses are offered by that department.

With Lambda-Means, we found that it was very difficult to control cluster size. Figure 2 shows that a significant portion of our classes (222 out of 2233 total) fell into their own cluster. In other words, of the 668 clusters returned, 222 of them contained one example. Furthermore, the quality of the results was uncertain. Providing quantitative measurements, here is an example: when we ran Lambda-means and examined the cluster containing 10.02 (classes that are similar to 10.02 by cosine similarity are shown in Table 2 ), it contained nine classes. Of these nine classes, six were in the same department. The other three classes included "Microeconomic Theory and Public Policy", "Introduction to Political Thought", and "System Safety". We felt that of the courses within Lambda-Means clusters were not always very similar to each other.

With Bisecting K-Means, we attempted two approaches. The first was to pick a random point in the cluster (rather than *generate* a random point) for $\mathbf{r}_a$, and use equation (10) to generate $\mathbf{r}_b$. We found the results to be unhelpful, and will discuss why in the next section. Results for our second method, where we generate a random $\mathbf{r}_a$ are shown in Table 3. Note that these results assume cosine similarity is "correct". We only report the results for one specific class, 10.02, as we feel they are representative of results for other courses.

Finally, we present our results for Spherical K-Means. In addition to Table 4, we also observed that our results were much closer to what we achieved with cosine similarity. Many of the courses in these clusters were also Chemical Engineering courses. Furthermore, given a large cluster size (i.e., with a

| Number of Iterations | Recall | Precision |
|---|---|---|
| 10 Iterations | 0.165 | 0.029 |
| 30 Iterations | 0.073 | 0.035 |
| 50 Iterations | 0.046 | 0.027 |
| 80 Iterations | 0.032 | 0.031 |

Table 3: Results of Bisecting K-Means for various numbers of iterations. We look at the cluster containing 10.02, and consider a member of that cluster to be "correct" if it has a similarity greater than 0.2 with 10.02. Results are averaged over twenty trials for each number of iterations.

| Number of Clusters ($k$) | Recall | Precision | % in ChemEng Department |
|---|---|---|---|
| 10 | 0.88 | 0.227 | 0.91 |
| 20 | 0.84 | 0.263 | 0.98 |
| 30 | 0.16 | 0.267 | 0.93 |
| 40 | 0.16 | 0.308 | 1.00 |
| 50 | 0.16 | 0.571 | 1.00 |
| 60 | 0.16 | 0.571 | 1.00 |
| 70 | 0.12 | 0.500 | 1.00 |

Table 4: Results of Spherical K-Means for various values of $k$. Recall and Precision are determined in the same manner as Table 3. The final column shows the percentage of courses in the cluster that were in the Chemical Engineering department.

smaller number of clusters total), we found that we were returning many of the same results that were being returned by cosine similarity.

### 4.3 Analysis

Cosine similarity, we felt, was a good basis for a similarity function. Simply by looking at the results of Table 2, we saw a number of courses that seemed very similar to 10.02, and might be good recommendations for someone that has taken that course. Its downside, we felt, was that it

Performance of our Lambda-Means algorithm was disappointing on this data set. We think that a lot of the problem was that the data was too sparse to get accurate results. Figure 2 shows the large number of clusters with only a few elements. We thought this might be because course descriptions can be very dissimilar from each other, and the algorithm might not be able to bring them together. Even more disappointing was that we were unable to tune our parameters to decrease the number of clusters and improve our results.

Bisecting K-Means, we felt, had a similar issue. In particular, our first method involved picking a random vector for $\mathbf{r}_a$, and using equation (10) to generate $\mathbf{r}_b$. However, when we do this, $\mathbf{r}_a$ will rarely have more than 100 defined dimensions (i.e., dimensions that are greater than 0), since course descriptions are rarely longer than that. However, since $\mathbf{w}$ is defined in every dimension, $\mathbf{r}_b$ will also be. This creates a heavy bias, and results in very uneven distributions when dividing a cluster in two.

Furthermore, bisecting k-means works by picking a random point, and dividing a cluster based on that point and the point that is opposite from the mean. When dimensions start to get very large, we feel this tends to water down the results. Although similar observations should end up together, because they are separated by a number of dimensions, it becomes less likely they end up in the same cluster. As a result, if two dissimilar courses happen to have a small number of words in common, they could easily end up in the same cluster.

Another problem with bisecting K-means is that, due to the random nature of bisecting clusters, it is possible that you could divide a very well-defined cluster into two. Currently, we choose to divide a cluster based on its size - i.e., the biggest cluster ends up getting divided. This is a crude heuristic, and an avenue for future work would be to come up with a metric metric to decide which clusters should be broken in two.

It is important to note that, although bisecting K-means did not produce results like cosine similarity, it was able to produce out-of-the-box results. Perhaps someone that took Philosophical History of Energy might be very interested in Introduction to Policital Thought. We find it less likely to be the case, but it is a good segue into the point that course descriptions may not be the best way to determine the similarity between two courses.

In contrast to the previous two algorithms, we were very pleased with our results for spherical K-means. On the one hand, it makes sense that it should perform closest to cosine similarity since they use identical similarity measures. However, we also were impressed because it was easy to tune parameters such as the number of clusters we wanted.

In each trial, it was able to produce results that cosine similarity also found similar, but was also able to find courses from across MIT to include in the grouping.

On the other hand, while this feature was nice for most cases, results could be quite disappointing for other cases. For example, when we set $k$ equal to 10 in our first trial, 91 percent of our results were in the Chemical Engineering department. Of those other 9 percent of results, the only suggestions those people have are in the Chemical Engineering department. This is due entirely to the nature of K-means, which assigns a single cluster to each observation. An avenue for future research would be to allow observations to have overlapping cluster assignments, at least to some degree.

## 5 Conclusion

### 5.1 Progress and Future Work

For this project, we were successfully able to achieve all our milestones outlined in our proposal. Our first project proposal was completely unrelated to the topic of this paper and involved the facility location problem but we were not able to find data to use. We then switched to the idea of a course recommendation system using document clustering. Our goal was to download class descriptions and report classes that are similar to a user input. We ran two different information retrieval algorithms, and four different clustering algorithms. Although our initial intent was to use data from the course listings at Johns Hopkins University, we were able to work around the complex formatting of the data by using course listings from MIT.

One thing that could be done in the future is to implement Latent Semantic Analysis (LSA) for preprocessing. LSA is a NLP technique that analyzes the similarity of documents and exploits the singular value decomposition for dimesionality reduction [1]. The similarity is computed, in this case, using a cosine function. In this context, we would apply LSA to each course instance the user provides to find similar courses from the corpus. The data would be represented as a matrix where each row represents a unique word and each column represents the course instance it belongs to. Each cell is the frequency of which it appears in that instance. Next, a transformation occurs according to a function that describes the word's frequency as well as its importance to the whole set of courses. This transformation requires the following steps: First, each cell of the matrix (the frequency of a word in a course instance) is converted to its $\log$. In the second step, the entropy of the word is computed as $-p \log p$ where $p$ is the sum of all entries in that row, and each cell entry is divided by the computed value. A singular value decomposition (SVD) is then applied to the matrix which reduces the dimensionality of the data. SVD breaks down the $w$ x $c$ matrix, $M$, of frequencies, into three matrices where $U$ and $V$ contains singular vectors and $S$ contains the singular values. This breakdown contains linearly independent components called factors. Assuming the singular values in $S$ are ordered by size, this allows for a dimensionality reduction by keeping the largest k values and setting the rest to zero and deleting the corresponding columns of $U$ and $V$ respectively [7].

## References


[1] Thomas K.Landauer, Peter W.Foltz, Darrell Laham 1998. *Introduction to Latent Semantic Analysis*

[2] Juan Ramos 2003. *Using TF-IDF to Determine Word Relevance in Document Queries*.

[3] Inderjit S. Dhillon, Yuqiang Guan, J.Kogan 2002. *Refining clusters in high dimensional text data*.

[4] Shi Zhong 2005. *Efficient Online Spherical K-Means Clustering*.

[5] Dan Pelleg, Andrew Moore 2000. *X-means: Extending K-means with Efficient Estimation of the Number of Clusters*.

[6] Sergio M. Savaresi, Daniel L. Boley 2004. *A comparative analysis on the bisecting K-means and the PDDP clustering algorithms*.

[7] A.N.K. Zaman, Pascal Matsakis, Charles Brown 2013. *Evaluation of stop word lists in text retrieval using Latent Semantic Indexing*.