Text Mining

Louise Almér Matilda Eriksson Astrid Johansson Algot Sandahl

December 10th 2020

1 Introduction

When dealing with large collections of texts that might be unstructured, there is a need for methods that can extract the information that is useful to the user. Such methods are called *text mining methods* and a typical application could be searching databases of abstracts to find relevant scientific papers. The user then composes a search phrase, a *query*, with relevant key words. The text mining method will then attempt to determine how these words relate to other, similar words and based on this collect the documents that it perceives as relevant to the query. The goal is to retrieve documents that are relevant while also excluding all documents that are irrelevant. This is known as the document retrieval problem.

In this project three different methods were implemented for document retrieval: *Latent Semantic Indexing (LSI)*, *Clustering* and *Non-negative Matrix Factorization (NNMF)*. The results of each method were analyzed and compared.

2 The Data Set

The data set that was used to evaluate the different methods consisted of abstracts from the MEDLINE database. MEDLINE is a database of research articles from different medical fields. A *term-document matrix* called A was given as well as a matrix q containing 30 medically related search queries. These were mapped to a dictionary containing all the words in the documents and queries. However, stopwords, which are words that do not add meaning to a sentence, such as *and*, *the* and *a*, had been removed. The words had also been stemmed, which means that they are reduced to their word stem. The abstracts and queries in their original format were also given. In addition to this, the data set contained data describing which documents were relevant to which queries—data that had been produced by hand. This data was used to assess the methods.

3 Theory

In this section, theoretical foundations and mathematical models that are relevant to the three methods LSI, clustering and NNMF are presented. The information in this section is based on [1].

3.1 Underlying Theory

The underlying theory presents the algorithms used for implementing the methods.

3.1.1 Singular Value Decomposition

Singular value decomposition (SVD) is a form of matrix decomposition that divides the matrix into its singular values and vectors. The SVD is often used in text-mining due to the advantage that the dominating part of the matrix is enhanced when projecting onto lower dimensions. The SVD has the form

$$A = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^T \tag{1}$$

where $U \in \mathbb{R}^{m\times m}$ is orthogonal and the columns of $U = (u_1, u_2, ..., u_m)$ are the left singular vectors, $V \in \mathbb{R}^{n\times n}$ is orthogonal and the columns of $V = (v_1, v_2, ..., v_n)$ are the right singular vectors and $\Sigma \in \mathbb{R}^{n\times n}$ is diagonal with the singular values, σ_i:

$$\Sigma = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_n), \quad \sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0. \tag{2}$$

3.1.2 Thin QR Decomposition

The QR decomposition uses orthogonal transformations to decompose a matrix into compact triangular form. The decomposition is described as

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix} \tag{3}$$

where $Q \in \mathbb{R}^{m\times m}$ is orthogonal and $R \in \mathbb{R}^{n\times n}$ is upper triangular. The matrix Q can be divided into two parts, Q_1 and Q_2, where Q_1 is the part of Q that remains after the decomposition. This is called the thin QR decomposition and is calculated with

$$A = Q_1 R \tag{4}$$

where $Q_1 \in \mathbb{R}^{m\times n}$. With the thin QR decomposition, the resulting orthogonal Q_1 matrix thereby has smaller dimensions than with the regular QR decomposition.

3.1.3 The k-means Algorithm

When using the k-means algorithm the data points, $(a_j)_{j=1}^n \in \mathbb{R}^m$, are stored in $A \in \mathbb{R}^{m\times n}$ where each column represents a data point. From A the data vectors can be partitioned into k clusters with $\Pi = (\pi_i)_{i=1}^k$ where $\pi_j = \{v \mid a_v \text{ belongs to cluster } j\}$. The clusters mean, *centroid*, can be calculated with

$$m_j = \frac{1}{n_j} \sum_{v \in \pi_j} a_v, \tag{5}$$

where the number of elements in π_j is n_j. The cluster π_j's tightness, *coherence*, can be described with the Euclidean distance and calculated as the sum $q_j = \sum_{v\in\pi_j} ||a_v - m_j||_2^2$. The coherence q_j is the quality of the cluster. If the vectors are close then the centroid q_j is small. For the overall coherence, the quality of the k-mean algorithm can be calculated with

$$Q(\Pi) = \sum_{j=1}^k q_j = \sum_{j=1}^k \sum_{v\in\pi_j} ||a_v - m_j||_2^2 \tag{6}$$

The k-means algorithm seeks the partitioning that has the optimal coherence, and the steps of the algorithm are as follows:

1. Choose an initial partitioning $\Pi^{(0)}$ randomly and compute the corresponding centroid vectors $(m_j^{(0)})_{j=1}^k$. Compute the coherence $Q(\Pi^{(0)})$ and put $t = 1$.
2. For each column vector a_j of A, find the closest centroid. If the closest centroid vector is $m_p^{(t-1)}$, assign the column vector a_j to the cluster $\pi_p^{(t)}$.
3. For the new partitioning $\Pi^{(t)}$, compute the centroid vectors $(m_j^{(t)})_{j=1}^k$.
4. Stop if $|Q(\Pi^{(t-1)}) - Q(\Pi^{(t)})| < tol$, otherwise increment t by 1 and return to step 2.

tol is a predefined tolerance value.

3.1.4 Nonnegative Matrix Factorization Algorithm

The purpose of a nonnegative matrix factorization is to find a rank-k approximation of a matrix A that only has nonnegative factors. The aim is thereby to solve

$$\min_{W\geq 0, H\geq 0} ||A - WH||_F \tag{7}$$

where $A \in \mathbb{R}^{m\times n}$, $W \in \mathbb{R}^{m\times k}$ and $H \in \mathbb{R}^{k\times n}$. This least linear squares problem can be solved with the *alternating nonnegative least squares algorithm*, which consists of the following steps:

1. Guess an initial value $W^{(1)}$
2. for i = 1, 2, 3, ... until convergence:
   a) Solve $\min_{H\geq 0} ||A - W^{(k)}H||_F$, giving $H^{(k)}$
   b) Solve $\min_{W\geq 0} ||A - WH^{(k)}||_F$, giving $W^{(k+1)}$

The initial value $W^{(1)}$ can be computed with the SVD of A. By retrieving the left matrix U from the SVD described in Section 3.1.1, the first column vector w_1 of $W^{(1)}$ can be approximated as the first column u_1 of U. The other columns in $W^{(1)}$ are approximated by performing $u_j v_j^T$ and replacing the negative values with zero. The first singular vector of this computation is the j:th column in $W^{(1)}$.

By performing a thin QR decomposition, $W = QR$ on $W^{(1)}$, the first value of H can be calculated with

$$H = R^{-1}Q^T A \tag{8}$$

Thereafter, new values of W and H are computed by the component-wise calculations

$$\begin{aligned} H_{ij} &= H_{ij} \frac{(W^T A)_{ij}}{(W^T WH)_{ij} + \epsilon} \\ W_{ij} &= W_{ij} \frac{(AH^T)_{ij}}{(WHH^T)_{ij} + \epsilon} \end{aligned} \tag{9}$$

until convergence. In every iteration before convergence, the negative values are removed from W and H. Furthermore, one of W and H has to be normalized and the other matrix has to be compensated with the normalization factor.

3.2 Methods

The implemented methods are based on the vector space model. The principal idea of the vector space model is to map each document to a column vector in a term-document matrix. The rows represent terms in the documents, where non-zero entries mean that the term can be found in the document. Provided a query q and the term-document matrix, the documents most relevant to the query can be found using query matching. This can be done by calculating the cosine of the angle between the query and each of the document vectors, deeming documents with cosines above a predefined threshold, tol, to be relevant. The cosine of the angle is calculated as

$$\cos(\theta(q, a_j)) = \frac{q^T a_j}{\|q\|_2 \, \|a_j\|_2} > \text{tol} \tag{10}$$

where q is the query and a_j is the document column vector. The accuracy of the model can be evaluated with precision, P, and recall, R, which are calculated by

$$P = \frac{D_r}{D_t}, \quad R = \frac{D_r}{N_r} \tag{11}$$

where D_r is the number of relevant documents retrieved, D_t is the total number of documents retrieved and N_r is the total number of relevant documents in the database.

The problem that arises when using the vector space model directly with the term-document matrix is that for large data sets, the comparison becomes increasingly expensive. To remedy this, a dimensionality reduction technique is used. In addition to improving the performance, this reduction can be done in such a way that the specific words used in the documents matter less, while the meaning is enhanced. The methods that are implemented in this project use this type of dimensionality reduction.

3.2.1 Latent Semantic Indexing

When using latent semantic indexing (LSI) it is assumed that the data contains an underlying latent semantic structure. By using SVD the data can be projected on lower dimensional space and find the latent semantic structure. The term-document matrix A can be written as (1).

It is assumed that matrix A contains noise that is lower than the accurate data. This means that A can be written as $A = A_0 + N$, where A_0 is the data without noise. The noise N can be avoided by using the k first values of A. The approximation for A can be written and simplified as

$$A = \sum_{i=1}^{n} \sigma_i u_i v_i^T \approx \sum_{i=1}^{k} \sigma_i u_i v_i^T = A_k \tag{12}$$

where

$$A_k = U_k \Sigma_k V_k^T =: U_k H_k. \tag{13}$$

The columns of H_k contains the orthogonal basis coordinates for each document and U_k's columns are in the document space of A. To compute the query matching we get

$$q^T A_k = q^T U_k H_k = (U_k^T q)^T H_k =: q_k^T H_k, \quad q_k = U_k^T q \tag{14}$$

and can then compute the query matching in k dimensions. The cosine can then be calculated as

$$\cos\theta_j = \frac{q_k^T h_j}{\|q_k\|_2 \|h_j\|_2} \tag{15}$$

where h_j is the j:th column of H_k.

3.2.2 Clustering

The clustering approach assumes that the documents represented by points in $\mathbb{R}^m$ form clusters in which the documents are similar in content. A way of representing each of these clusters is by its mean—its *centroid*. To find the centroids, the k-means algorithm is used, see Section 3.1.3. The k centroids span an approximate document space in which the query–document comparison can be performed. Let $C_k \in \mathbb{R}^{m\times k}$ be formed with the normalized centroids as columns, then the approximate representation of the documents, $\hat{G}_k$, in the new basis is obtained when

$$\left\| A - C_k \hat{G}_k \right\|_F \tag{16}$$

is minimized. Orthogonal columns are however a useful property and by first finding the thin QR decomposition of C_k:

$$C_k = P_k R, \quad P_k \in \mathbb{R}^{m\times k}, \quad R \in \mathbb{R}^{k\times k} \tag{17}$$

the basis P_k can be used instead. Then the documents are instead represented by G_k and the problem becomes

$$\min_{G_k} \|A - P_k G_k\|_F \tag{18}$$

or equivalently, n independent least squares,

$$\min_{g_j} \|a_j - P_k g_j\|_2, \quad j = 1, 2, ..., n \tag{19}$$

where a_j and g_j is column j of A and G_k respectively and n is the number of columns in Gk. P_k has orthogonal columns by definition, so $g_j = P_k^T a_j$ and (18) is solved by

$$G_k = P_k^T A \tag{20}$$

To compare a query to a document, $q^T A$ is approximated using the new basis:

$$q^T A \approx q^T P_k G_k = (P_k^T q)^T G_k =: q_k^T G_k. \tag{21}$$

The cosine is then calculated as

$$\cos\theta_j = \frac{q_k^T g_j}{\|q_k\|_2 \|g_j\|_2}, \quad q_k = P_k^T q. \tag{22}$$

3.2.3 Nonnegative Matrix Factorization

With the nonnegative matrix factorization approach, an approximate matrix factorization of the term-document matrix A is computed according to Section 3.1.4, where both W and H are nonnegative,

$$A \approx WH. \tag{23}$$

In order to compare the query vector q to the documents in the approximation, q must be represented in the same basis as W. By solving the least squares problem,

$$\min_{\hat{q}} \|q - W\hat{q}\|, \tag{24}$$

the reduced basis, $\hat{q}$, of q can be found. By performing a thin QR decomposition of W, the solution to the least squares problem becomes

$$\hat{q} = R^{-1} Q^T q. \tag{25}$$

The query vector $\hat{q}$ is then compared to a document vector by computing the cosine between them as

$$\cos\theta_j = \frac{\hat{q}^T h_j}{\|\hat{q}\|_2 \|h_j\|_2} \tag{26}$$

where h_j is j:th column vector in H.

4 Implementation

The three methods LSI, clustering and NNMF were used to calculate the cosine of the angle between the queries and documents in the given data set. The calculations were implemented in MATLAB and followed closely the theory described in Section 3.2.

4.1 Query Matching and Performance Modeling

For all three methods, the cosine of the angle between the queries and each document vector was calculated according to (10), in their respective bases. Each method computed approximations of q and the document column vectors a_j in different ways.

The precision and recall was calculated per query and method as described in (11). The tolerance for the cosines was linearly increased from 0 to 0.98 and the precision and recall was calculated for each tolerance value. For the purpose of analyzing the methods for all queries, the average precision for all queries was computed. This was done by setting an interval of the recall, starting at 5% and with steps of 5 going up to 90%. The precision for each query at these recall points was calculated with linear interpolation. Some of the precision values were NaN, since some values of tolerance did not return any documents, which resulted in division by zero according to (11). Therefore, before the linear interpolation, the NaN values had to be set to zero. After the interpolation, the sum of the precision for each query in the recall points were calculated. The average was computed by dividing the sum by the number of queries that did not have NaN as precision. The result was visualized by plotting the recall on the x-axis and the average precision on the y-axis using MATLAB's function *plot*.

4.2 Latent Semantic Indexing

The LSI method was implemented by calculating the truncated SVD using the MATLAB function *svds*. From the svds, the matrices U, Σ and V were retrieved and were used to calculate H_k according to (13) and q_k according to (14). The cosines were then calculated according to (15). Different values of the rank k were tested to achieve the best result of the LSI method.

4.3 Clustering

The clustering method was implemented in MATLAB with the help of the *kmeans*-function, which divides the documents into k clusters based on the distances between the documents. *kmeans* gives the matrix C_k, and using the thin QR decomposition the matrix P_k is retrieved and the cosines could then be calculated. This was all implemented according to Section 3.2.2.

According to MATLAB's documentation, the *kmeans*-function has several parameters that could be specified in order to improve the result. These were mainly the number of clusters k, the maximum number of iterations allowed when creating the clusters, the method used for choosing the initial cluster centroid positions and the distance metric used to calculate the distances between the documents.

The methods available for choosing the initial cluster centroids were 'plus', which implements the k-means++ algorithm for cluster center initialization to select k seeds, 'sample', which selects k observations from the data at random, 'cluster', which performs a preliminary clustering on a random 10% subsample using 'sample', and lastly 'uniform', which chooses points uniformly at random.

The distance metrics available were 'sqeuclidean', the squared euclidean distance, 'cityblock', the sum of the absolute differences, 'cosine', one minus the cosine of the angle between the points, and 'correlation', one minus the sample correlation between points [2].

Different combinations of these parameters where tested in order to achieve the best result.

4.4 Nonnegative Matrix Factorization

A method for computing the nonnegative matrix factorization was implemented according to the theory in Section 3.1.4. The initial value of $W^{(1)}$ was computed by doing a svd of A with the MATLAB function *svds* and then iterating over the following loop:

```
1. for j = 2,3,...,k
      a) C = U(:,j)*V(:,j)'
      b) C = C.*(C≥0)
      c) u, s, v = svds(C,1)
      d) W(:j) = abs(u)
```

With the approximation of W, the initial value of $H^{(1)}$ was calculated with (8) and the next values of W and H where calculated according to (9) until convergence. The convergence was decided by calculating the norm of the difference between the new value of W and the previous value. A threshold value was set and the loop was iterated over until the calculated norm reached the threshold.

After the NNMF approximation had been calculated, it was used for dimensionality reduction as described in Section 3.2.3. The QR demposition of W was calculated with MATLAB's *qr* function. The cosines were then calculated according to (26).

The rank of the SVD which was computed when finding the initial value of $W^{(1)}$ and the thresholds were tested in order to find the best result for the method.

5 Result

This section presents the results of the parameter testing, an evaluation of the basis vectors for each method as well as a comparison between the implemented methods.

5.1 Parameter testing

The parameters used in the models affect the outcome and in this section the result of the parameter testing is presented.

5.1.1 Latent Semantic Indexing

The important parameter of the LSI method is the rank, k, of the lower-dimensional space. As seen in Figure 1, a rank around 100 gives the best results, whereas higher or lower values produce worse results. This is a tendency that continues for even lower or higher k than those shown in Figure 1.

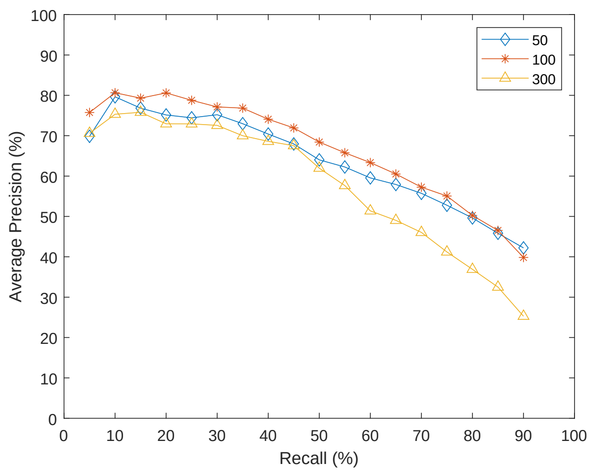


Figure 1: Examples of how different ranks affect the outcome from LSI

5.1.2 Clustering

The clustering algorithm was tested with different combinations of the parameters that could be explicitly chosen for the *kmeans*-function in order to find which combination gave the best result. The four different combinations of method for choosing the initial cluster centroid positions and distance metric that were considered to give the best result were 'Uniform' with 'Cosine', 'Sample' with 'Correlation', 'Cluster' with 'Cosine' and 'Cluster' with 'Correlation'. In general, a higher value for maximum number of iterations gave more consistently good results. An example of the result when using these four different combinations of parameters can be seen in Figure 2.

The clustering method was also tested for different values of the rank to see how that affected the result. A plot displaying an example of the output for three different values of the rank k can be seen in Figure 3.

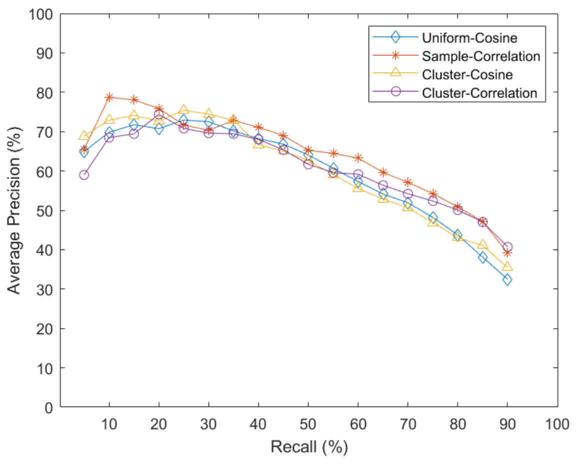


Figure 2: Example of the result of the clustering method with four different combinations of parameters for the *kmeans*-function. *MaxIter* was set to 100 and rank = 50.

For this plot, the maximum number of iterations was set to 100, the initial cluster centroid positions were determined by the method 'Cluster' and the distance metric was set to 'Correlation'.

It can be seen in Figure 3a that there is not a significant difference in the result when k is 40, 100 or 160. Looking at Figure 3b, it can again be seen that when k has values 150 and 300, the result seems to be about the same as for the values of k in Figure 3a, while $k = 20$ gives a significantly worse result. The conclusion drawn from this is that once a certain threshold is reached, dividing the documents into more clusters does not make a significant difference. Thus it seems for this data that increasing the number of clusters to above $k = 40$ does not lead to improved results and thus computational costs can be limited by keeping k at a low value.

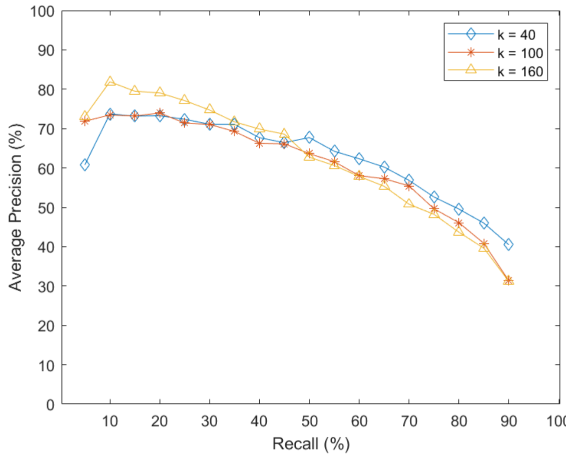
5.1.3 Nonnegative Matrix Factorization

The NNMF method was explored by varying two different parameters: the rank and the threshold. The rank k is the rank of the SVD that is used to compute the initial value of W. The result of different values of the rank k is displayed in Figure 4. The Figure shows that rank 100 gives the best result, while higher and lower values of the rank give worse results.

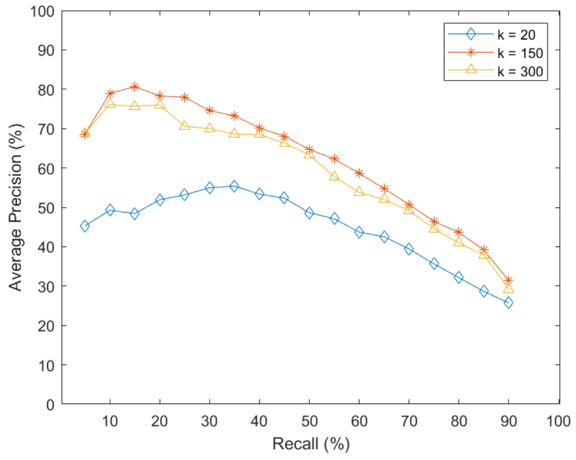
The threshold is the value that decides when W converges and therefore how many times W and H are recalculated before finding their final values. The NNMF method was tested for different threshold values, the result of which is displayed in Figure 5. The figure shows that the difference is small between the results obtained by the different threshold values.

5.2 Basis evaluation

To evaluate how well each method grouped together words about similar topics, the five most important



(a)



(b)

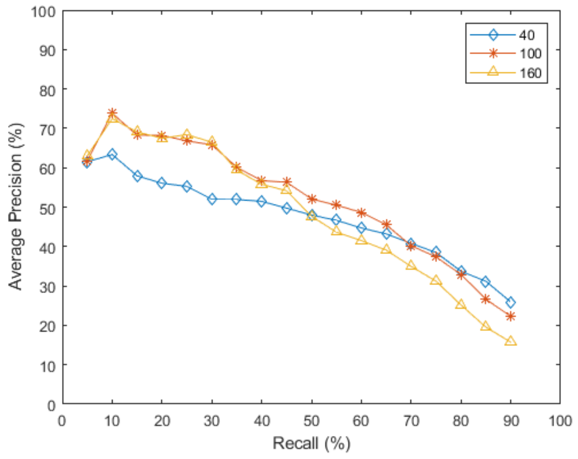
Figure 3: Example of the result of the clustering method with different values of the rank k.

words in each basis vector for each method were retrieved. The most important words are those with the highest absolute value in the basis vector. In this section, a selection of these groups of words are presented.

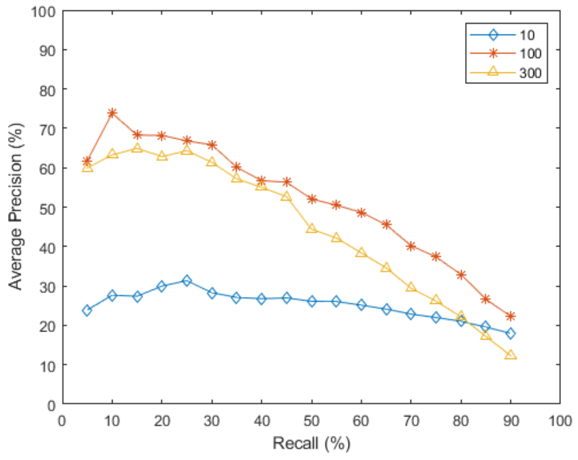
For the LSI, the basis vectors are found in the columns of U_k. For column 1 and 5, the words are presented in Table 1. The basis vectors for clustering can be found in the columns of P_k and represent the words in each cluster. The most important words in two columns, column 6 and 43, are presented in Table 2. For NNMF, the basis vectors are the columns of W. For two of those columns, column 9 and 46, the most important words are presented in Table 3.

Table 1: Examples of most important words from two basis vectors for the LSI method.

Column 1	cell, dna, growth, increas, patient
Column 5	infect, lymphocyt, marrow, phage, strain



(a)



(b)

Figure 4: Result of the NNMF method with different ranks, and threshold=1.

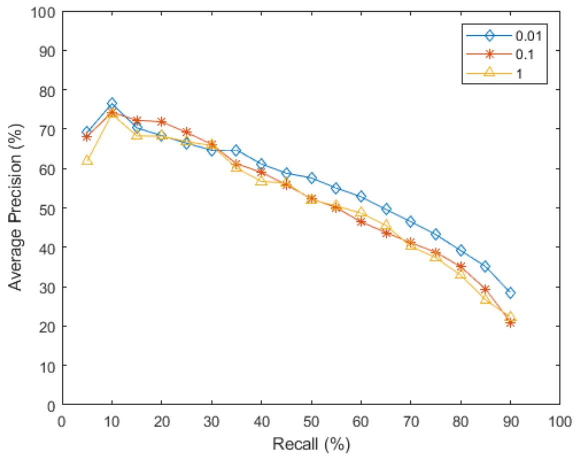


Figure 5: Result of the NNMF method with different thresholds, and rank = 100.

5.3 Method Comparison

To compare the accuracy of the methods, the average precision for each method was plotted in the same

Table 2: Examples of most important words from two basis vectors for the clustering method.

Column 6	oper, patient, procedur, surgeri, surgic
Column 43	breast, cancer, chemotherapi, patient, therapi

Table 3: Examples of most important words from two basis vectors for the NNMF method.

Column 9	cerebr, cistern, oxygen, pressur, tension
Column 46	donor, drug, homograft, reject, surviv

plot using the parameters that gave the best result in the parameter testing. Consequently, the ranks for LSI and NNMF were set to 100 and to 40 for clustering. Furthermore, the parameter *Start* was set to *cluster* and *Distance* to *correlation* for clustering and for NNMF, the threshold was set to 1. Figure 6 displays the plot.

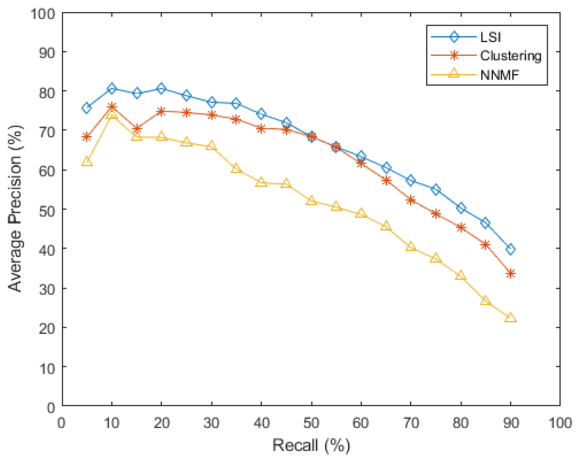


Figure 6: Result of all three methods. LSI has rank 100. Clustering has rank 40, 'Start'='Cluster' and 'Distance'='Correlation'. NNMF has rank 100 and threshold = 1.

The figure shows that on average, the LSI har the highest average precision, followed by clustering and then NNMF. The difference between LSI and NNMF is at most 20% in the precision and is by consequence less for clustering.

6 Discussion

As seen in Section 5.3, on average LSI was the most accurate and NNMF the least accurate. However, this depends on the parameters used in all the methods. The difference in the average precision is however fairly small, which means that for individual queries, there can be a difference in which method is the most accurate. Furthermore, this result is only applicable to medical data, since no testing has been performed on other data sets.

6.1 Latent Semantic Indexing

The LSI method was the simplest implementation that was used. Since it did not have any random elements in the code the output was not altered between the testing with certain parameters. From Figure 1 it is noticeable that rank 100 gives the highest average precision for all recall values, except in one point where rank 50 gives a slightly higher precision. For a rank 300 the average precision is significantly lower for higher recall values than for rank 50 and 100. Therefore, rank 100 gives the most accurate result.

6.2 Clustering

For the clustering method there were several parameters that could be chosen for the *kmeans*-function. There is a degree of randomness involved in the choosing of the initial cluster centroids, leading to different results of the algorithm each time. In order to find a consistently good result, various combinations of the initial cluster centroid choosing methods and the distance metrics were tested several times. The combinations that seemed to produce fairly consistently good results are displayed in Figure 2, and it can be seen that it is difficult to determine if one combination produces better results than another. The randomness also gave slightly different plots each time, in which none of the combinations could be determined to be consistently better or worse than the others. As can be seen in Figure 3, varying the value of the rank between 40 and 300 does not appear to have a great effect on the result.

6.3 Nonnegative Matrix Factorization

The result of the NNMF in Figure 4a displays that ranks 40, 100 and 160 give good results. However, a rank of 100 gives a higher average precision than both 40 and 160. Figure 4b shows that ranks higher than 160 and lower than 40 decreases the average precision further. Rank $k = 100$ can therefore be seen as the best rank.

When experimenting with different thresholds, see Figure 5, it can be seen that a fairly similar result is obtained when using the three different threshold values 0.01, 0.1 and 1. A higher threshold will mean less iterations and will therefore be more computationally efficient. Since the result of the different thresholds are similar, threshold = 1 can be seen as the best threshold since it gives a good result and also results in few iterations.

6.4 Basis Evaluation

The purpose of the basis evaluation described in Section 5.2 is to evaluate how well the methods group together words about similar topics. With a limited medical knowledge, it is difficult to say with certainty how well the words are related. However, there appears to be some relation between the words in each group.

In the words for the LSI, see Table 1, the words in column 1 seem to be related to cellular science and the words in column 5 to bacterial infection.

For the words in Clustering, see Table 2, the words in cluster 6 seem related to surgeries, while the words in cluster 43 suggest that these documents discuss treatments for cancer. Since there is an element of randomness in the division of the clusters, these divisions will not be the same each time, but the results suggest that there does seem to be a correlation between the documents that are placed in the same cluster.

In Table 3 with words for the NNMF, column 9 have mainly words regarding the brain function and column 46 is about organ donation.

As a conclusion, each method seems to group together words on similar subjects.

7 Conclusion

All three methods gave relatively good results, but when comparing them to each other it is clear that the LSI gives the best average result, followed by the clustering method and lastly NNMF. However, the methods have only been tested on medical documents. Therefore, the result only applies to this type of data and the results might be different with documents on other topics.

References

[1] L. Eldén. *Matrix Methods in Data Mining and Pattern Recognition.* Society for Industrial and Applied Mathematics, second edition, 2019.

[2] MathWorks. kmeans. `https://se.mathworks.com/help/stats/kmeans.html`. Accessed: 2020-12-08.