CHAPTER 5
TEXT CLUSTERING

5.1. K-Means Clustering of Documents

A simple and widely-used clustering method is the $k$-means algorithm (see, e.g., [Weiss 98, pp. 102-103], [Berry 97, pp. 192-205], [Johnson & Wichern 02 pp. 694-700]). In its basic form, the algorithm is as follows [Johnson & Wichern 02 p. 694]:

Algorithm 5.1:
1. Define $k$ initial clusters and calculate their centroids.
2. Proceed through the list of items, assigning an item to the cluster whose centroid (mean) is nearest. Recalculate the centroids for the cluster receiving the new item and for the cluster losing the item.
3. Repeat Step 2 until no more assignments take place.

Typically, cluster centroids are calculated averaging the coordinates along each dimension, i.e., \( \bar{c} = \frac{1}{m} (\bar{v}_1 + \ldots + \bar{v}_m) \). [Hsu 99] use $k$-means clustering to divide a group of news articles into $k$ sub-groups, employing such mean vectors for cluster representation. This approach is fine, but in the case of small documents, not necessary. Note that documents belonging to the Vector Space Model have the property \( \text{Sim}(\bar{d}, \bar{c}) = \text{Sim}(\vec{d}, m\bar{c}) \), therefore we can substitute the cluster centroid with the sum \( \bar{q} = \bar{v}_1 + \ldots + \bar{v}_m \). This modification results in a very practical way of cluster representation: simply concatenate all documents within a cluster to form, rather than a document set centroid, a term concatenation, where each cluster is represented by all sets of terms corresponding to the documents in the cluster pooled together. This way the existing inverted list can be fully exploited and document similarities to clusters can be calculated very fast by running $k$ queries corresponding to the $k$ term concatenations. Algorithm 5.2 below adjusts the $k$-means clustering algorithm to the case of document clustering.
Algorithm 5.2:
1. Define \( k \) initial clusters and calculate their term concatenations \( q_i, i = 1, \ldots, k \).
2. Perform \( k \) queries using \( q_i \) as the query documents. For each document \( d_j \) in the collection, \( j = 1, \ldots, n \), consider the similarities \( \text{Sim}(d_j, q_i) \) and assign \( d_j \) to the cluster whose term concatenation \( q_i \) is nearest. Recalculate the term unions for the cluster receiving the new document and for the cluster losing the document.
3. Repeat Step 2 until no more assignments take place.

In their \( k \)-means clustering of news articles, [Hsu 99] use the variance of the similarity values of the articles in each cluster as an evaluation criterion. This is a surprising choice, and possibly a result of misconception. While it is true that variance is a standard measure of within-cluster similarity in a typical \( k \)-means clustering setting, that variance is defined as the average squared distance of each element from the mean [Berry 97, p. 211]. In the case of text clustering, document similarity is already the vector-space equivalent of the Euclidean squared distance, so one simply has to consider the average within-cluster document similarity as a measure of text variation: documents in groups with a low average within-group similarity will be spread around whereas in groups with a high average within-group similarity, documents will be packed together. [Steinbach 00] also uses the average pairwise similarity as the basis for one measure for quantifying the goodness of clustering. Intuitively, one is tempted to divide this average within-cluster similarity by the average between-cluster similarity, and create the ratio \( \text{Clustering Strength} = \frac{\text{Average within-cluster similarity}}{\text{Average between-cluster similarity}} \). This is a Signal-to-Noise Ratio, since the within-cluster similarity is a measure of information (similar documents were discovered and placed in the same cluster) and the between-cluster similarity a measure of noise (more distant clusters would produce a more clear-cut clustering solution and closer clusters a more ambiguous one).

Once we know the correct document classifications, a clustering solution that was chosen using internal clustering quality measures can be validated externally. Quite a few external quality measures appear in the literature. One of them uses Information Theory arguments and computes the weighted average of cluster conditional entropies [Cover 90]. Another uses Theory of Information Retrieval arguments and combines the measures of Recall and Precision to form a so-called \( F \) measure [Larsen 99]. And another considers the operations of moving documents among clusters one at a time, and calculates the percentage of savings from using the clustering result to construct the correct classification using such operations, versus constructing the correct classification from scratch [Pantel 02].
Figure 5.1 illustrates two such clusters populated with documents with a high degree of within-cluster similarity. Because our Vector Space representation of documents uses the cosine of the angle between vectors as a measure of similarity, the usual $n$-dimensional Cartesian space with the rectangular coordinates gives way to a system of normalized polar coordinates where radius lengths do not matter, only angles do. Essentially all documents get projected on an $n$-dimensional spherical surface (a hyper-sphere) marked on Figure 5.1 as the document projection horizon. VSM’s view of the document clusters then becomes similar to an observer’s view of the starry night sky. Some stars (or planets, or even galaxies for that matter, since they all look so similar) appear to form clusters, but are they really close to each other? Well, stars that appear to participate in star clusters, for one, usually are! In the following sections we will investigate what happens with document clusters.

![Figure 5.1. Clusters of documents in the Vector Space.](image)

5.2. Internal Document Clustering Quality Measures

In $k$-means clustering, the choice of initial seeds can sometimes be critical in obtaining the best possible clusters. Figure 5.2 (a) illustrates a situation where 5 documents are to form 2 clusters with \{d_0\} and \{d_1\} as the seeds. At the end of the $k$-means algorithm, the resulting clusters are \{d_0\} and \{d_1, d_2, d_3, d_4\} as shown in Figure 5.2 (b). Had we chosen \{d_1\} and \{d_3\} as the initial seeds, the final clustering arrangement would have been \{d_0, d_1, d_2\} and \{d_3, d_4\}, as illustrated in Figure 5.2(c).
The latter arrangement is more favorable because it is more coherent. As we discussed earlier, the larger the similarity between a graded document and a new document, the better the grade estimation accuracy. Defining coherent clusters and having a solved case (graded document) in each cluster will, therefore, boost estimation accuracy, because more un-graded documents will be closer to their best match out of the Case Base.

Referring back to our H.323 exam question example, k-Means Text Clustering (Algorithm 5.2) was executed and the Average Within-Cluster Similarity, as well as the Clustering Strength was recorded. Figure 5.3 plots these two measures after 400 runs of the algorithm, for a fixed number of clusters $k = 7$. We observe quite a bit of variability among the runs which is not surprising, as the $k$-means algorithm is known to produce a number of different solutions. We also observe that the two measures appear to be highly correlated and there appears to be a clustering solution that maximizes both measures. Since it will not make much difference which one we choose to use in our document clustering algorithm, we will employ Average Within-
Figure 5.3. Plot of Clustering Strength vs. Average Within-Cluster Similarity for the H.323 data. Number of runs $M = 400$, number of clusters $k = 7$.

Figure 5.4. Histogram of Average Within-Cluster Similarity (Overall Similarity) for the H.323 data. Number of runs $M = 400$, number of clusters $k = 7$. 
Cluster Similarity and we will rename it to Overall Similarity, to be consistent with [Steinbach 00].

In view of the previous discussion, the \( k \)-means text clustering algorithm now can be amended as follows:

**Algorithm 5.3:**

For \( m = 1 \) to \( M \), where \( M \) is a moderately large number;
1. Select a randomly chosen set of \( k \) seed documents that define \( k \) initial clusters and calculate their term concatenations \( q_i, i = 1, \ldots, k \).
2. Perform \( k \) queries using \( q_i \) as the query documents. For each document \( d_j \) in the collection, \( j = 1, \ldots, n \), consider the similarities \( \text{Sim}(d_j, q_i) \) and assign \( d_j \) to the cluster whose term concatenation \( q_i \) is nearest. Use heuristics to deal with orphaned documents and empty clusters. Recalculate the term concatenations for the cluster receiving the new document and for the cluster losing the document.
3. Repeat Step 2 until no more assignments take place.
4. Calculate the Overall Similarity of the final clustering arrangement and save the clustering arrangement having the maximum Overall Similarity.

Next \( m \).

The distribution of Overall Similarity for the aforementioned 400 runs appears on Figure 5.4. Since the distribution appears to be approximately bell-shaped with asymptotic tails, it would pay off to try a moderately large number of runs before choosing the clustering solution where the maximum Overall Similarity is attained. (Note that, since Overall Similarity is an average of \( N \) similarities and \( N \) here is 28, due to the Central Limit Theorem its distribution is expected to be approximately Normal; in fact, the 400 measurements in our illustration pass all major statistical normality tests). Among these 400 runs, we would choose one that has overall similarity equal to 0.5933. This is the clustering solution with the highest internal quality. But will it be a solution associated with more precise grade estimates? We will investigate this issue in the next section.

### 5.3. External Document Clustering Quality Measures

Inspired by the approach in [Pantel 02], we will provide information related to both cost and benefit from our clustering solution. The benefit will be the grading precision and the cost will be the number of clusters, which represents the initial human effort. For the time being, let us assume that, with a clustering solution at hand, we will assign each cluster to the correct grade of its cluster representative document, which we will identify as the one which is most similar to its own cluster.
concatenate (centroid). As an illustration, Table 5.1 shows the best clustering solution for \( k = 5 \) chosen after \( M = 40 \) runs of Algorithm 3.1. Participation of documents into the clusters is indicated by indices equal to 1 on the left-hand side of Table 5.1. The best representative documents for the 5 clusters were selected as documents 11, 6, 27, 16, and 7, respectively, indicated by gray-highlighted cells. The right-hand side of the table shows the true (correct) grades of all documents. Choosing each cluster’s representative’s grade as the estimated grade for all members of the cluster, we get a total of 18 concordant (correctly estimated) documents, including the cluster representatives, which are assumed to be known during the estimation. This corresponds to a precision equal to \( 18/28 = 64\% \).

The choice of the number of clusters \( k \) can be critical in the final clustering performance, as collections of items often tend to have some natural clustering characteristics. In order to investigate the effect of the number of clusters on the clustering solution’s external quality, an external loop varying \( k \) is added to the text clustering algorithm, on top of a repetitions loop seeking to explore the variation in the external quality of the clustering arrangements:

Algorithm 5.4:

For a few alternative values of \( k \):

For \( r = 1 \) to \( R \), where \( R \) is a moderately large number;

For \( m = 1 \) to \( M \), where \( M \) is a moderately large number;

1. Select a randomly chosen set of \( k \) seed documents that define \( k \) initial clusters and calculate their term concatenations \( q_i, i = 1,\ldots, k \).

2. Perform \( k \) queries using \( q_i \) as the query documents. For each document \( d_j \) in the collection, \( j = 1,\ldots, n \), consider the similarities \( \text{Sim}(d_j, q_i) \) and assign \( d_j \) to the cluster whose term concatenation \( q_i \) is nearest. Use heuristics to deal with orphaned documents and empty clusters. Recalculate the term concatenations for the cluster receiving the new document and for the cluster losing the document.

3. Repeat Step 2 until no more assignments take place.

4. Calculate the Overall Similarity of the final clustering arrangement and save the clustering arrangement having the maximum Overall Similarity.

Next \( m \).

Calculate a measure of external quality of the final clustering arrangement obtained at the conclusion of the inner loop.

Next \( r \).

Next \( k \).
Figure 5.5 shows the average and the maximum grading precision of among the 28 documents of the H.323 example, when each answer’s grade is estimated simply as being equal to the grade of the representative of the cluster where it was placed (i.e., we do not apply adaptation). Figure 5.5 reveals that perhaps \( k = 6 \) is an optimal arrangement where the 28 documents form natural clusters around documents that represent 6 characteristic grading possibilities. The figure also demonstrates the dramatic improvement we can get in clustering classification precision if we increase the number of runs to a moderately large number (maximum line on Figure 5.5 corresponds to the best result after trying 100 * 50 = 5,000 runs).

Since the final grades of the documents in the collection are not yet known during the initial clustering stage, we will have to resort to evaluating clusters using Overall Similarity. How will this internal quality measure relate to the external quality?

<table>
<thead>
<tr>
<th>Doc</th>
<th>Cluster:</th>
<th>Cluster:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 5.1.** Selected Clustering of the H.323 exam answers and corresponding grades.
measure of grading precision? Figure 5.6 explores this relationship for the case of our H.323 example after 400 runs, for \( k = 6 \). Surprisingly, the correlation between the two measures does not improve with the increase in the number of clusters \( k \), but remains constant around 0.51 for \( k = 6, k = 12, \) and \( k = 15 \).

The relationship between Grading Precision and Overall Similarity, although not an extremely strong one, appears to be consistent, independent of \( k \), moderately strong, and positive. A clustering solution that maximizes the internal measure of Overall Similarity results in a suboptimal, but nevertheless fairly good grading precision.

The next section presents the specification of a programming class for K-Means Text Clustering, which will be used subsequently in our Case-Based Grading engine.

### 5.4. Specification of a KMTC Class

Figure 5.7 shows the class diagram of a KMTC class. Only selected class members and public methods are shown.

![Grading Precision Graph](image)

**Figure 5.5.** Plot of *Grading Precision* as a function of the number of clusters \( k \). Number of runs \( M = 100 \), number of repetitions \( R = 50 \).
Figure 5.6. Plot of **Grading Precision vs. Overall Similarity** for the H.323 data. Number of runs \( M = 400 \), number of clusters \( k = 6 \).

**Fig. 5.7.** Class Diagram of KMTC class.
5.5. Implementing KMTC in Java

Figure 5.8 shows how the related Java classes should be compiled.

PERFORMING KMTC IN JAVA

1) Create a folder called D:\JavaApps\tm. In there, place files Porter.java, VSM.java, KMTC.java, examstoplist.txt, DBConnection.java, DocumentList.java, db.properties, and TM1_1.mdb.
2) Make sure your db.properties file has the correct database filename.
3) Compile Porter, VSM, DBConnection, DocumentList and KMTC.
4) Place the KMTCTestDB.java file in D:\JavaApps. In this file, make sure:
   a. The DBConnection constructor has the correct db.properties folder.
   b. The SQL statement references table and attribute names correctly.
   c. The stoplist filename in method vsm.loadStopwords() is correct.
   d. The IR query document is what you want it to be.
   e. The output file filename is what you want it to be.
5) Compile and run KMTCTestDB.java.
6) Look for the clustering results in the output file cluster1.out.

Figure 5.9 shows the KMTC results. The 38 documents are distributed over 6 clusters. In each cluster, a “cluster representative” document that is the closest to the cluster centroid has been selected.
5.6. Document Classification

Classifiers have been widely used in document categorization (indexing), also known as Text Categorization (TC). Document categorization is the assignment of natural language texts to one or more predefined categories based on their content. Methods for construction of document classifiers include inductive learning techniques [Cohen and Singer, 1996], [Lewis et al., 1996], probabilistic and decision tree classifiers [Lewis and Ringuette, 1994], genetic algorithms, neural networks [Wiener et al., 1995], and classifiers based on support vector machines.

A problem arises when the number of document categories becomes very large as text repositories grow in size. It is then common to use hierarchies, also called taxonomies, to manage such complexities [Chakrabarti et al., 1997], [Koller, 1997]. Taxonomies can be used to relieve the user from the burden of sifting specific information from the large and low quality response of most popular search engines [Raghavan, 1997]. Querying with respect to a taxonomy is more reliable than depending on the presence or absence of specific keywords.

Work on IR also includes statistical modeling of documents, clustering [Anick and Vaithyanathan, 1997], hierarchical classification analysis [Jardin and van Rijsbergen, 1971], thesaurus generation by term associations [Voorhees, 1993], and
query expansion. Supervised classification that incorporates class-membership information has been addressed in statistical decision theory, statistical pattern recognition, and machine learning. Training set loss functions and complexity penalties may require careful tuning since they may have a significant impact on the performance of a text classifier such as Linear Regression, Neural Network, Logistic Regression, Naïve Bayes, or K-Nearest Neighbor [Li and Yang 03]. In its general form, a supervised learning Text Categorization algorithm first assigns relevance scores to each document-category pair. It then uses a training set of documents categorized by the human supervisor. Finally, using the training categorization, it empirically estimates the right threshold in the relevance score that will determine the outcome of the binary decision of whether the document will be classified under a certain category. Such thresholds also need careful tuning [Yang 01].

References


