Document Clustering and Latent Semantic Indexing

CISC689/489-010, Lecture #18
Wednesday, April 22nd
Ben Carterette

Clustering Review

• Cluster documents according to similarity in feature space
• Types of clustering:
  – Flat or hierarchical
  – “Hard” or “soft”
• Clustering algorithms:
  – K-means: flat, “hard” clusters
  – Agglomerative or divisive hierarchical: hierarchical, softer clusters
K-Nearest Neighbor Clustering

• Alternative idea: fixed-size soft clusters
• \textit{K-nearest neighbor}: for each item \( j \), cluster it with the \( K \) things most similar to it
• K-nearest neighbor clustering forms one cluster per item
  – Clusters overlap
  – Does not necessarily have to cluster everything

5-Nearest Neighbor Clustering
Evaluating Clustering

• Clustering will never be 100% accurate
  – Documents will be placed in clusters they don’t belong in
  – Documents will be excluded from clusters they should be part of
  – A natural consequence of using term statistics to represent the information contained in documents

• Like retrieval and classification, clustering effectiveness must be evaluated

• Evaluating clustering is challenging, since it is an *unsupervised* learning task

### Evaluating Clustering

• If labels exist, can use standard IR metrics, such as precision and recall
  – In this case we are evaluating the ability of our algorithm to discover the “true” latent information

<table>
<thead>
<tr>
<th>Cluster 1</th>
<th>Class A</th>
<th>Class B</th>
<th>Class C</th>
<th>Class D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_1</td>
<td>B_1</td>
<td>C_1</td>
<td>D_1</td>
<td></td>
</tr>
<tr>
<td>Cluster 2</td>
<td>A_2</td>
<td>B_2</td>
<td>C_2</td>
<td>D_2</td>
</tr>
<tr>
<td>Cluster 3</td>
<td>A_3</td>
<td>B_3</td>
<td>C_3</td>
<td>D_3</td>
</tr>
<tr>
<td>Cluster 4</td>
<td>A_4</td>
<td>B_4</td>
<td>C_4</td>
<td>D_4</td>
</tr>
</tbody>
</table>

- \[ \text{prec}_{\text{cluster } i} = \frac{A_i}{A_i + B_i + C_i + D_i} \]
- \[ \text{rec}_{\text{cluster } i} = \frac{A_i}{A_i + A_2 + A_3 + A_4} \]

• This only works if you have some way to “match” clusters to classes

• What if there are fewer or more clusters than classes?
Evaluating Clusters

• “Purity”: the ratio between the number of documents from the dominant class in C to the size of C

\[ purity(C_i) = \frac{1}{|C_i|} \max_j |X| \text{ s.t. } X \in C_i \text{ and } X \in K_j \]

— C_i is a cluster; K_j is a class

• Not such a great measure
  — Does not take into account coherence of the class
  — Optimized by making N clusters, one for each document

Evaluating Clusters

• With no labeled data, evaluation is even more difficult

• Best approach:
  — Evaluate the system that the clustering is part of
  — E.g. if clustering is used to aid retrieval, evaluate the cluster-aided retrieval

• What kinds of systems use clustering?
Clusters in IR Systems

- Automatic clustering has several uses:
  - Improving efficiency
  - Improving effectiveness of index
  - Improving effectiveness of retrieval

Improving Efficiency

-Clusters can be a time- and space-saving device:
  - Cluster all documents in the corpus
  - Compare queries to cluster representations rather than document representations
  - Store cluster information in inverted lists rather than document information

\[ t_i \rightarrow (c_j \cdot f_{i,j}) (c_1 \cdot t_{f_{1,i}}, \ldots) \]

- Important in the 70s and 80s, not so much today
Improving Effectiveness

• Recall the cluster hypothesis:
  – “Closely associated documents tend to be relevant to the same requests”
• We may be able to improve retrieval by finding documents that are closely associated with documents that are likely to be relevant
  – Even if they don’t contain query terms
  – E.g. perhaps they contain related terms the user didn’t think of (“industry” → “companies”, “businesses”, “producers”, …)

Improving Effectiveness by Ranking Clusters

• As with clustering for efficiency, cluster all documents before indexing
• Store cluster information in inverted lists (but keep document information too)
• When a user enters a query, score the clusters
• Then score documents within the top-scoring clusters
Improving Effectiveness by Ranking Clusters

• Two approaches:
  – Rank the clusters from highest scoring to lowest scoring; within each cluster rank documents from highest scoring to lowest scoring
  – Rank the documents in the K highest-scoring clusters from highest score to lowest score

• Both tend to find relevant documents that are not found with non-clustering methods
  – But also miss relevant documents that are found with non-clustering methods

Scoring Clusters

• A cluster can be scored the same way as a document
  – Vector space model: cosine similarity between query vector and cluster centroid
  – Language model: \[ P(Q|C) = \prod_{t \in Q} P(t|C) = \prod_{t \in Q} (1 - \alpha_C) \frac{t^{f_t}}{|C|} + \alpha_C \frac{c^{f_t}}{|G|} \]
  – Or other ways:
    - \[ S(C, Q) = \max_{D_i \in C} S(D_i, Q) \]
    - \[ S(C, Q) = \min_{D_i \in C} S(D_i, Q) \]
    - Smoothing with collection
    - \[ S(C, Q) = \frac{1}{|C|} \sum_{D_i \in C} S(D_i, Q) \]
Using Clusters to Adjust Document

- Language models require background to smooth with
- Clusters provide a background, perhaps more focused than using entire collection
- Smooth document language model with cluster background first, then with collection background
  - \( P(w \mid D) \) – frequency of term in document
  - \( P(w \mid C) \) – frequency of term in all documents in a cluster
  - \( P(w \mid G) \) – frequency of term in all documents in the collection

Smoothing Document Scores

- Basic approach: linear interpolation
  \[
  P(w \mid D) = \lambda \frac{tf}{|D|} + \lambda_2 \sum_{D \in C} \frac{tf}{|D|} + \lambda_3 \frac{ctf}{|G|}
  \]
- Better approach: smooth cluster with collection, then smooth document with both
  \[
  P(w \mid D) = \lambda \frac{tf}{|D|} + (1 - \lambda) \left( \beta \frac{\sum_{D \in C} tf}{|D|} + (1 - \beta) \frac{ctf}{|G|} \right)
  \]
Query-Time Clustering

- Clustering the entire collection takes a long time
- Can only use simple algorithms like K-means
- Instead, cluster the top documents ranked for a query
- Use those clusters to re-score and re-rank the documents

Does it Work?

- Retrieving clusters: Verdict: no apparent improvement

<table>
<thead>
<tr>
<th>Collection</th>
<th>First-stage documents</th>
<th>Group-average</th>
<th>Single-linkage</th>
<th>Complete-linkage</th>
<th>centroid</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP (training)</td>
<td>0.1179</td>
<td>0.2161 (t=0.8)</td>
<td>0.2153 (t=0.8)</td>
<td>0.2130 (t=0.8)</td>
<td>0.2164 (t=0.7)</td>
</tr>
<tr>
<td>WSJ</td>
<td>0.2958</td>
<td>0.2992 (t=0.8)</td>
<td>0.2911 (t=0.8)</td>
<td>0.2849 (t=0.8)</td>
<td>0.2936 (t=0.8)</td>
</tr>
</tbody>
</table>

- Clustering at index time, use clusters for smoothing Verdict: improvement over baselines

<table>
<thead>
<tr>
<th>Collection</th>
<th>Simple-Index</th>
<th>QL-DM</th>
<th>QL-DMRM</th>
<th>%Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP (K=1000)</td>
<td>0.2771</td>
<td>0.2679</td>
<td>0.2500</td>
<td>-6.85%</td>
</tr>
<tr>
<td>WSJ (K=2000)</td>
<td>0.2792</td>
<td>0.2651</td>
<td>0.2500</td>
<td>-4.86%</td>
</tr>
<tr>
<td>FT (K=2000)</td>
<td>0.2796</td>
<td>0.2651</td>
<td>0.2500</td>
<td>-4.86%</td>
</tr>
<tr>
<td>TREC (K=2000)</td>
<td>0.2001</td>
<td>0.2001</td>
<td>0.2001</td>
<td>0.00%</td>
</tr>
<tr>
<td>LA (K=2000)</td>
<td>0.2771</td>
<td>0.2679</td>
<td>0.2500</td>
<td>-6.85%</td>
</tr>
<tr>
<td>FR (K=1000)</td>
<td>0.2644</td>
<td>0.2675</td>
<td>0.3316</td>
<td>-15.37%</td>
</tr>
</tbody>
</table>
Does it Work?

- Clustering at query time, use clusters for smoothing

<table>
<thead>
<tr>
<th>Collection</th>
<th>Threshold</th>
<th>QL-TDID</th>
<th>QL-CBID</th>
<th>%Imp</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td>0.2</td>
<td>0.2107</td>
<td>0.2221</td>
<td>+5.4%</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.2113</td>
<td>0.2211</td>
<td>+4.3%</td>
</tr>
<tr>
<td>WSJ</td>
<td>0.2</td>
<td>0.2563</td>
<td>0.2654</td>
<td>-11.0%</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.2563</td>
<td>0.2998</td>
<td>-11.5%</td>
</tr>
<tr>
<td>FR</td>
<td>0.2</td>
<td>0.2409</td>
<td>0.2807</td>
<td>+11.8%</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.2305</td>
<td>0.2210</td>
<td>+31.6%</td>
</tr>
</tbody>
</table>

Verdict: improvement over baselines

Clusters in IR – Summary

- Index-time clustering
  - Saves space in inverted file
  - Build topical hierarchies
  - Retrieve clusters of documents rather than individual documents

- Query-time clustering
  - After query is submitted, cluster results
  - Possibly detect subtopics or different interpretations of query
Latent Semantic Indexing

- Clustering only puts documents together based on term similarity
- Can we do more than that?
  - We’d like synonyms and highly related terms to count for more when calculating document similarity
  - And words that have multiple senses to count for less
- How can we do this?

Linear Algebra Background

- A vector space is defined by a set of linearly independent basis vectors
  - Linearly independent: no vector can be expressed as a linear combination of other vectors
- Every vector can be expressed as a linear combination of the basis vectors

- Example:

  $B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

  These three vectors form a 3-dimensional vector space

  $B = \begin{bmatrix} 1 & 2 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

  These three vectors form a 2-dimensional vector space
Bases in the Vector Space Model

• When we discussed the VSM, we assumed bases could be formed from terms
  – Document and query vectors are linear combinations of term basis vectors
• But basis vectors have to be linearly independent—do terms satisfy this?
  – Probably not
  – Two terms that always appear together are not independent
  – More insidious example:
    • “bush” appears in documents about landscaping and documents about politics
    • The vector for “bush” may be a linear combination of many vectors for terms related to landscaping and terms related to politics

Bases in the Vector Space Model

• Is there a better way to choose the bases?
• *Semantic concepts*
  – Find a group of topics or concepts that are orthogonal
    • E.g. “landscaping” and “politics” topics are probably linearly independent
  – Each concept forms a basis vector
  – A document vector is a linear combination of concept vectors
• Note: this is not easy to do
  – Solving this problem would probably solve all of AI
Finding Linearly Independent Bases

• There are methods for finding linearly independent basis vectors
• We will apply one method and assume that the bases it produces represent “concepts”

Linear Algebra Background

• Matrix-vector multiplication:
  \[ Ax = \begin{bmatrix} a_{11} & a_{12} & \cdots \\ a_{21} & a_{22} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots \\ a_{21}x_1 + a_{22}x_2 + \cdots \\ \vdots \end{bmatrix} = x_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \end{bmatrix} \]

  – A transforms \( x \)

• Eigenvalues and eigenvectors
  – Given a square matrix \( A \), the eigenvectors of \( A \) are the vectors \( x \) such that \( Ax \) is a scalar multiple of \( x \)
    • i.e. the vectors that are only transformed by length, not by direction
  \[ Ax = \lambda x \]
  Every \( x \) that satisfies this equation is an eigenvector.
  The \textit{eigenvalues} \( \lambda \) show how much \( A \) shortens or elongates \( x \).
Eigenvectors and Eigenvalues

• Example:
  \[
  \begin{bmatrix}
  6 & -2 \\
  4 & 0
  \end{bmatrix}
  \begin{bmatrix}
  1 \\
  2
  \end{bmatrix}
  =
  \begin{bmatrix}
  2 \\
  4
  \end{bmatrix}
  =
  2 \begin{bmatrix}
  1 \\
  2
  \end{bmatrix}
  \]

• \([ 1 \ 2 ]\)' is an eigenvector of the matrix, and 2 is an eigenvalue

• How many eigenvalues are there for an \( n \times n \) matrix?
  \[
  Ax = \lambda x \Longleftrightarrow (A - \lambda I)x = 0
  \]
  – \( x \) is nonzero only if determinant of \( A - \lambda I = 0 \)
  – The determinant is a polynomial in \( \lambda \) of degree \( n \)
  – Therefore there are at most \( n \) eigenvalues

Examples

• Example 1:
  \[
  A = \begin{bmatrix}
  30 & 0 & 0 \\
  0 & 20 & 0 \\
  0 & 0 & 1
  \end{bmatrix}
  \]
  eigenvalues = 30, 20, 1
  eigenvectors = \[
  \begin{bmatrix}
  1 \\
  0 \\
  0
  \end{bmatrix}
  \]

• Example 2:
  \[
  A = \begin{bmatrix}
  0.71 & 0.03 & -0.01 \\
  0.03 & 1.02 & 0.03 \\
  -0.01 & 0.03 & 0.70
  \end{bmatrix}
  \]
  eigenvalues = 1.03, 0.72, 0.69
  eigenvectors = \[
  \begin{bmatrix}
  -0.09 \\
  -0.99 \\
  0.09
  \end{bmatrix}
  \]

• Exercise: verify that \( Ax = \lambda x \) holds
Eigenvectors and Eigenvalues

• Eigenvectors of symmetric matrices are \textit{linearly independent}
  
  — Why? If an eigenvector $x$ could be written as a sum of other vectors, then $Ax = \lambda x$ would not be true—$x$ would not be an eigenvector in the first place!

• Therefore eigenvectors of symmetric matrices form a basis
  
  — Every vector in the space is a linear combination of the eigenvectors

Eigenvectors and Eigenvalues

• Eigenvalues of real-valued matrices are real numbers

• Eigenvalues of \textit{positive semidefinite} matrices are non-negative
Eigen Decompositions

- A square matrix $A$ can be decomposed into a matrix product $A = U\Lambda U^{-1}$ where
  - $U$ is a matrix with eigenvectors of $A$ as columns
  - $\Lambda$ is a matrix with eigenvalues of $A$ in decreasing order on the diagonal and 0 elsewhere
- A symmetric square matrix $A$ can be decomposed into a matrix product $A = Q\Lambda Q'$
  - $Q$ is a real orthogonal matrix with normalized eigenvectors as columns

So How Does This Apply to IR?

- If we had the right kind of matrix, a decomposition might be able to find orthogonal “concepts” in the document/term data
- Those concepts might be a better basis for a vector space model
- We could take the $K$ most important concepts (corresponding to the $K$ greatest eigenvalues) to reduce the dimensionality of the space
- ... but we don’t have square symmetric matrices in IR
Eigen Decomp in IR

- Arrange N documents and V terms into a V x N matrix called A, where $A_{ij} = \text{term weight of term } i \text{ in document } j$
  - This is neither square nor symmetric
- We can compute the following matrix products:
  - $AA' = \text{a V x V matrix of document similarities}$
  - $A'A = \text{a N x N matrix of term similarities}$
  - Note that these are both are square and symmetric—both have eigen decompositions

Singular Value Decomposition

- SVD uses $AA'$ and $A'A$ to compute an eigen decomposition of $A$
  - $AA'$ and $A'A$ have the same number m of eigenvectors
    - $m \leq \min(N, V)$
- Specifically: $A = U\Sigma V'$
  - $U = \text{a V x m matrix with eigenvectors of } AA' \text{ as cols}$
  - $V = \text{a N x m matrix with eigenvectors of } A'A \text{ as cols}$
  - $\Sigma = \text{an m x m matrix with the square roots of eigenvalues of } AA' \text{ on the diagonal}$
    - Eigenvalues of $AA' = \text{eigenvalues of } A'A$
SVD on the Document-Term Matrix

- Eigenvectors of $AA'$ represent “document concepts”
- Eigenvectors of $A'A$ represent “term concepts”
- Eigenvalues give the relative weight of the concepts
- The occurrence of a term in a document is a linear combination of “term concepts” and “document concepts”

Illustration

$$A = U \Sigma V'$$

Each column of $U$ is a “document concept”
Each row of $V'$ is a “term concept”

$$A_{ij} = \Sigma_{ij} \sum_{k=1}^{m} U_{ik} V_{kj}$$
Example

From Deerwester, Dumais, Harshman, “Indexing by Latent Semantic Analysis”

Example

Verify: $T_0 S_0 D_0 = \text{original document-term matrix}$
Optimal Dimensionality Reduction

- SVD can also be used for dimensionality reduction
- Reduce $\Sigma$ to the top-$k$ largest eigenvalues
- For documents and terms, this effectively reduces the dimensionality to the $k$ most important “concepts”
  - Furthermore, the reduction is optimal—it is the best reduction you could possibly do given the document-term matrix

Reducing to $k=2$ Concepts
Latent Semantic Analysis for Retrieval

Now D contains the new document vectors
- Each a 2-D vector of “concept weights”

To process a query Q:
- Use T to transform it to the 2-D concept space
- Weight the concepts using S
- $Q' = QT S^{-1}$
- Calculate cosine similarity between $Q'$ and each document

Example

Q = “human computer”
$Q' = [1 0 1 0 0 0 0 0 0 0 ]$
$Q'T = [1*0.22+1*0.24 1*-0.11+1*0.04 ]$
= [0.46 -0.07 ]
$Q'TS^{-1} = [0.46/3.34 -0.07/2.54 ] = [0.14 -0.03 ]$
LSI: Does it Work?

• Seems to be a little better than standard vector space model
  – Recall is good: intuitively it is retrieving “clusters” of documents related by topic, which improves recall
  – Precision is OK: it can find things that are not really related
• When it does not do well, it is hard to understand what happened
• It takes a very long time to do the SVD
  – In 1990, one day for ~10,000 documents