Information Diversity
alfa-NDCG, duplicate detection and document clustering

Web Search
Overview

Indexing

Information analysis

Crawler

Indexes

Ranking

Results

Query processing

Application

Query

Documents

User

Multimedia documents
Information diversity

• Corpus diversity
  • Usually, a corpus has many different topics discussed across different documents.
  • Organizing a corpus into groups of documents, unveils the diversity of topics covered by the corpus.

• Search results diversity
  • Many search results talk about the same information facts.
  • Grouping search results by their content, enables the computation of equally relevant documents, but more informative results.
Google News: automatic grouping gives an effective news presentation metaphor
For better navigation of search results

• For grouping search results thematically
  • clusty.com / Vivisimo
Measuring information diversity

• Diversity is difficult to evaluate.

• There is no defacto method to measure it.

• The goal is to measure how diverse is the information contained in the retrieved documents.
  • Assessment focus is not at the level of the documents.
Nuggets or information facts

• A **nugget** is an information fact
  • **Documents** contain many nuggets.
  • The same **nugget** can be present in many different documents.

• The goal is to retrieve a ranked list with many different nuggets at the top of the list

• Repeated nuggets will have a decreasing importance
The $\alpha$-nDCG metric for diversity and novelty

- The relevance of a document is determined by its $m$ nuggets

$$\sum_{j=1}^{m} N(d_i, n_j)$$

- In a rank, the relevance of a document in position $k$ is discounted by the number of times its nuggets occurred previously:

$$r_{j,k-1} = \sum_{i=1}^{k-1} N(d_i, n_j)$$

- A popular metric is the $\alpha$-nDCG, where each document at position $k$ is judged by its nuggets

$$G[k] = \sum_{j=1}^{m} \frac{N(d_k, n_j)}{\alpha r_{j,k-1}} \quad \alpha = 2$$
Example

• Top results for query “Norwegian Cruise Lines”

<table>
<thead>
<tr>
<th>Document Title</th>
<th>85.1</th>
<th>85.2</th>
<th>85.3</th>
<th>85.4</th>
<th>85.5</th>
<th>85.6</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. Carnival Re-Enters Norway Bidding</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>b. NORWEGIAN CRUISE LINE SAYS...</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>c. Carnival, Star Increase NCL Stake</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>d. Carnival, Star Solidify Control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>e. HOUSTON CRUISE INDUSTRY GETS...</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>f. TRAVELERS WIN IN CRUISE...</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>g. ARMCHAIR QUARTERBACKS NEED...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>h. EUROPE, CHRISTMAS ON SALE</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>i. TRAVEL DEALS AND DISCOUNTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>j. HAVE IT YOUR WAY ON THIS SHIP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

\[ r_{j,k-1} = \sum_{i=1}^{k-1} N(d_i, n_j) \]

\[ G[k] = \sum_{j=1}^{m} \frac{N(d_k, n_j)}{2^{r_{j,k-1}}} \]

• The relevance of each document is: \( G = \langle 2, \frac{1}{2}, \frac{1}{4}, 0, 2, \frac{1}{2}, 1, \frac{1}{4}, \ldots \rangle \).

• What would be the ideal ordering?

\( a-c-g-b-f-c-h-i-j-d \) \( G' = \langle 2, 2, 1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \ldots \rangle \).
Detecting duplicate information

• Clustering
  • Documents in the same group discuss similar topics.
  • Groups need to be large enough to support the creation of a cluster.

• Hashing
  • Traditional hashing methods only work with exact matches.
  • We need to detect near-duplicate documents.
Near-duplicate documents
Duplicate detection, hashing functions
Web Search
Duplicate documents

• The web is full of duplicated content

• Strict duplicate detection = exact match
  • Not as common

• But many, many cases of near duplicates
  • E.g., Last modified date the only difference between two copies of a page
Duplicate/near-duplicate detection

• Duplication: Exact match can be detected with fingerprints

• Near-Duplication: Approximate match
  • Compute syntactic similarity with an edit-distance measure
  • Use similarity threshold to detect near-duplicates
    • E.g., Similarity > 80% => Documents are “near duplicates”
    • Not transitive though sometimes used transitively
Computing similarity

• Features:
  • Segments of a document (natural or artificial breakpoints)
  • Shingles (Word N-Grams)
  • a rose is a rose is a rose → 4-grams are
    a_rose_is_a
    rose_is_a_rose
    is_a_rose_is
    a_rose_is_a

• Similarity measure between two docs: intersection of shingles
Jaccard coefficient

- Jaccard coefficient computes the similarity between sets.

$$ Jaccard(C_i, C_j) = \frac{|C_i \cap C_j|}{|C_i \cup C_j|} $$

- View sets as columns of a matrix A:
  - one row for each shingle in the universe
  - one column for each document
  - $a_{ij} = 1$ indicates presence of shingle $i$ in document $j$

- Example:  $Jaccard(C_1, C_2) = \frac{3}{6}$

<table>
<thead>
<tr>
<th>Shingles</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 0</td>
<td></td>
</tr>
<tr>
<td>1 1 0 1</td>
<td></td>
</tr>
<tr>
<td>0 1 0 1</td>
<td></td>
</tr>
<tr>
<td>0 0 0 1</td>
<td></td>
</tr>
<tr>
<td>1 0 0 1</td>
<td></td>
</tr>
<tr>
<td>1 1 1 0</td>
<td></td>
</tr>
<tr>
<td>1 0 1 0</td>
<td></td>
</tr>
</tbody>
</table>
Key Observation

• For columns $C_i$, $C_j$, four types of rows

<table>
<thead>
<tr>
<th></th>
<th>$C_i$</th>
<th>$C_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shingle A</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Shingle B</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Shingle C</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Shingle D</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

• Overload notation: $A = \#$ of rows of type A

• Claim

$$\text{Jaccard}(C_i, C_j) = \frac{A}{A + B + C}$$
Shingles + Set Intersection

• Computing exact set intersection of shingles between all pairs of documents is expensive

• Approximate using a cleverly chosen subset of shingles from each document (*a sketch*)

• Estimate Jaccard coefficient based on a short sketch
Sketch of a document

• Create a “sketch vector” (of size ~200) for each document

  • Documents that share $\geq t$ (say 80%) corresponding vector elements are deemed near duplicates

• For doc $D$, $\text{sketch}_D[i]$ is as follows:
  • Let $f$ map all shingles in the universe to $1..2^m$ (e.g., $f =$ fingerprinting)
  • Let $p_i$ be a random permutation on $1..2^m$
  • Pick $\text{MIN} \{p_i(f(s))\}$ over all shingles $s$ in $D$
Computing Sketch[i] for Doc1

Document 1

Start with 64-bit $f(\text{shingles})$

Permute on the number line with $\pi_i$

Pick the min value
Test if Doc1.Sketch[i] = Doc2.Sketch[i]

Are these equal?

Test for **200** random permutations: $\pi_1, \pi_2, \ldots, \pi_{200}$
However...

A = B iff the shingle with the MIN value in the union of Doc1 and Doc2 is common to both (i.e., lies in the intersection)

Claim: This happens with probability

\[
\frac{\text{Size of intersection}}{\text{Size of union}}
\]
Minimum hashing

• Random permutations are expensive
  • If we have 1 million documents and each document has 10,000 shingles... there’s ~1 billion different shingles.
  • One needs to store 200 random permutations
  • Doing all permutations is not actually needed.

• **Answer**: implement permutations as random hash functions

  • For example: \( h_{a,b}(x) = ((a \cdot x + b) \mod p) \mod N \)

  where:

  \( a, b \) ... random integers
  \( p \) ... prime number (\( p > N \))
Min-Hashing example

<table>
<thead>
<tr>
<th>Permutations $\pi$</th>
<th>Input matrix</th>
<th>Signature matrix $M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 4 3</td>
<td>1 0 1 0</td>
<td>2 1 2 1</td>
</tr>
<tr>
<td>3 2 4</td>
<td>1 0 0 1</td>
<td>1 2 4 1</td>
</tr>
<tr>
<td>7 1 7</td>
<td>0 1 0 1</td>
<td>1 2 1 2</td>
</tr>
<tr>
<td>6 3 2 6</td>
<td>0 1 0 1</td>
<td></td>
</tr>
<tr>
<td>1 6 2 6</td>
<td>0 1 0 1</td>
<td></td>
</tr>
<tr>
<td>5 7 1 5</td>
<td>1 0 1 0</td>
<td></td>
</tr>
<tr>
<td>4 5 5</td>
<td>1 0 1 0</td>
<td></td>
</tr>
</tbody>
</table>

**Documents**

**Shingles**

**Input matrix**

**Signature matrix $M$**

**Similarities:**

<table>
<thead>
<tr>
<th>Col/Col</th>
<th>Sig/Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>0.75</td>
</tr>
<tr>
<td>2-4</td>
<td>0.75</td>
</tr>
<tr>
<td>1-2</td>
<td>0</td>
</tr>
<tr>
<td>3-4</td>
<td>0</td>
</tr>
<tr>
<td>Sig/Sig</td>
<td>0.67</td>
</tr>
<tr>
<td></td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
Minimum hashing - implementation

• Input: N documents
• Create n-grams (shingles)
• Pick 200 random permutations, as hash functions
  • Generate and store 200 random numbers, one for each hash function.
  • Hash function $i$ can be obtained with .hashCode() XOR random number $i$
• For each one of the 200 hash function permutation
  • Select the hashcode of the shingle with the lowest hashcode
• Compute N sketches: 200xN matrix
  • Each document is represented by 200 hashcodes (integers)
• Compute $N*(N-1)/2$ pairwise similarities
  • Each vector now has 200 integers from the hashes.
  • Each integer corresponds to the minimum shingle of a given hash permutation.
• Choose the closest ones.
Min-Hashing example with random hashing

<table>
<thead>
<tr>
<th>DocX shingles</th>
<th>hashA()</th>
<th>hashB()</th>
<th>hashC()</th>
<th>hashD()</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>a rose is a</td>
<td>103</td>
<td>19032</td>
<td>09743</td>
<td>98432</td>
<td></td>
</tr>
<tr>
<td>rose is a rose</td>
<td>1098</td>
<td>3456</td>
<td>89032</td>
<td>98743</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4539</td>
<td>6578</td>
<td>89327</td>
<td>21309</td>
<td></td>
</tr>
<tr>
<td></td>
<td>243</td>
<td>2435</td>
<td>93285</td>
<td>29873</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8876</td>
<td>7746</td>
<td>9832</td>
<td>98321</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2486</td>
<td>9823</td>
<td>30984</td>
<td>30282</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Doc X minHash signature: 103, 2435, 9743, 21309, ...
Notes

• At the end, you select near-duplicate candidates.
  • You still must do a direct comparison.
  • There is a chance to retrieve false positives.

• The N*(N-1)/2 pairwise similarities can be computationally prohibitive for large N.
  • Still manageable for small N, e.g. for search results.

• LSH reduces the search space (the N documents).
Information clustering
k-Means, Purity and RandIndex
Web Search
Applications of clustering in IR

• Whole corpus analysis/navigation
  • Better user interface: search without typing

• For improving recall in search applications
  • Better search results (like pseudo RF)

• For better navigation of search results
  • Effective “user recall” will be higher

• For speeding up vector space retrieval
  • Cluster-based retrieval gives faster search
A data set with clear cluster structure

• How would you design an algorithm for finding the three clusters in this case?
Issues for clustering

• Representation for clustering
  • Document representation
    • Vector space? Normalization?
      • Centroids aren’t length normalized
  • Need a notion of similarity/distance

• How many clusters?
  • Fixed a priori?
  • Completely data driven?
    • Avoid “trivial” clusters - too large or small
      • If a cluster's too large, then for navigation purposes you've wasted an extra user click without whittling down the set of documents much.
Notion of similarity/distance

• Ideal: semantic similarity.

• Practical: term-statistical similarity
  • We will use cosine similarity.
  • Docs as vectors.
  • For many algorithms, easier to think in terms of a distance (rather than similarity) between docs.
  • We will mostly speak of Euclidean distance
    • But real implementations use cosine similarity
Partitioning algorithms

• Partitioning method: Construct a partition of \( n \) documents into a set of \( K \) clusters

• Given: a set of documents and the number \( K \)

• Find: a partition of \( K \) clusters that optimizes the chosen partitioning criterion
  • Globally optimal
    • Intractable for many objective functions
  • Effective heuristic methods: K-means and K-medoids algorithms
**K-Means**

- Assumes documents are real-valued vectors.
- Clusters based on centroids (aka the center of gravity or mean) of points in a cluster, $c$:

$$\mu(C) = \frac{1}{|c|} \sum_{x \in c} x$$

- Reassignment of instances to clusters is based on distance to the current cluster centroids.
  - (Or one can equivalently phrase it in terms of similarities)
**K-Means algorithm**

K-MEANS(\{\vec{x}_1, \ldots, \vec{x}_N\}, K)
1. \((\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_K) \leftarrow \text{SELECTRANDOMSEEDS}(\{\vec{x}_1, \ldots, \vec{x}_N\}, K)\)
2. \textbf{for} \(k \leftarrow 1\) \textbf{to} \(K\)
3. \hspace{1em} \textbf{do} \(\vec{\mu}_k \leftarrow \vec{s}_k\)
4. \hspace{1em} \textbf{while} \ stopping criterion has not been met
5. \hspace{1em} \textbf{do for} \(k \leftarrow 1\) \textbf{to} \(K\)
6. \hspace{2em} \textbf{do} \(\omega_k \leftarrow \{\}\)
7. \hspace{2em} \textbf{for} \(n \leftarrow 1\) \textbf{to} \(N\)
8. \hspace{3em} \textbf{do} \(j \leftarrow \text{arg min}_{j'} |\vec{\mu}_{j'} - \vec{x}_n|\)
9. \hspace{3em} \(\omega_j \leftarrow \omega_j \cup \{\vec{x}_n\}\) \ (reassignment of vectors)
10. \hspace{2em} \textbf{for} \(k \leftarrow 1\) \textbf{to} \(K\)
11. \hspace{3em} \textbf{do} \(\vec{\mu}_k \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}\) \ (recomputation of centroids)
12. \textbf{return} \(\{\vec{\mu}_1, \ldots, \vec{\mu}_K\}\)

(7) For each doc \(d_i\), (9) assign \(d_i\) to the cluster \(c_j\) such (8) that \(dist(x_i, s_j)\) is minimal.

(10) For each cluster \(c_j\), 

(11) \(s_j = \mu(c_j)\)

**Figure 16.5** The K-means algorithm. For most IR applications, the vectors \(\vec{x}_n \in \mathbb{R}^M\) should be length-normalized. Alternative methods of seed selection and initialization are discussed on page 364.
K-Means example (k=2)

Pick seeds
Reassign clusters
Compute centroids
Reassign clusters
Compute centroids
Reassign clusters
Converged!
Termination conditions

• A fixed number of iterations.

• Doc partition unchanged.

• Centroid positions don’t change.

• ... other criterion.
Seed choice

• Results can vary based on random seed selection.

• Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings.
  • Select good seeds using a heuristic (e.g., doc least similar to any existing mean)
  • **Try out multiple starting points**
  • Initialize with the results of another method.
How many clusters?

• Number of clusters $K$ is given
  • Partition $n$ docs into predetermined number of clusters

• Finding the “right” number of clusters is part of the problem
  • Given docs, partition into an “appropriate” number of subsets.
  • E.g., for query results - ideal value of $K$ not known up front - though UI may impose limits.

• Can usually take an algorithm for one flavor and convert to the other.
$K$ not specified in advance

- Solve an optimization problem:
  - penalize having lots of clusters
  - application dependent,
    - e.g., compressed summary of search results list.

- Trade-off between having more clusters (better focus within each cluster) and having too many clusters
Given a clustering, define the **benefit** for a doc to be the cosine similarity to its centroid.

- Benefit can also be defined as the probability that the document is drawn from the cluster model.

- Define the **Total Benefit** to be the sum of the individual doc Benefits.
Penalize lots of clusters

• For each cluster, we have a Cost $C$.

• Thus for a clustering with $K$ clusters, the Total Cost is $KC$.

• Define the Value of a clustering to be

$$\text{Value} = \text{Total Benefit} - \text{Total Cost}$$

• Find the clustering of highest value, over all choices of $K$.
  • Total benefit increases with increasing $K$. But can stop when it doesn’t increase by “much”. The Cost term enforces this.
Akaike Information Criterion (AIC)

\[ AIC = 2k - 2 \log \left( \sum_i p(x_i) \right) \]

- \( k \) = number of estimated parameters.
- \( p(x_i) \) is the probability of each sample under the computed model.

https://www.google.com/doodles/hirotugu-akaikes-90th-birthday
External criteria for clustering quality

• Quality measured by its ability to discover some or all of the hidden patterns or latent classes in gold standard data

• Assesses a clustering with respect to ground truth ... requires *some labeled data*

• Assume documents with $C$ gold standard classes, while our clustering algorithms produce $K$ clusters, $\omega_1, \omega_2, \ldots, \omega_K$ with $n_i$ members.
External evaluation of cluster quality

- Simple measure: purity, the ratio between the dominant class in the cluster $\pi_i$ and the size of cluster $\omega_i$

$$Purity(\omega_i) = \frac{1}{n_i} \max_j (n_{ij}) \quad j \in C$$

- Biased because having $n$ clusters maximizes purity
- Others are entropy of classes in clusters (or mutual information between classes and clusters)
Cluster I: Purity = 1/6 (max(5, 1, 0)) = 5/6

Cluster II: Purity = 1/6 (max(1, 4, 1)) = 4/6

Cluster III: Purity = 1/5 (max(2, 0, 3)) = 3/5
Rand Index measures between pair decisions.

<table>
<thead>
<tr>
<th>Number of points</th>
<th>Same Cluster in clustering</th>
<th>Different Clusters in clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same class in ground truth</td>
<td>20</td>
<td>24</td>
</tr>
<tr>
<td>Different classes in ground truth</td>
<td>20</td>
<td>72</td>
</tr>
</tbody>
</table>

\[ RandIndex = \frac{A + D}{A + B + C + D} = 0.68 \]
RandIndex and cluster F-measure

\[ RandInd = \frac{A}{A + B + C + D} \]

• Compare with standard Precision and Recall:

\[ P = \frac{A}{A + B} \quad R = \frac{A}{A + C} \]

• People also define and use a cluster F-measure, which is probably a better measure.

\[ F_{\text{measure}} = 2 \cdot \frac{1}{\frac{1}{P} + \frac{1}{R}} = 2 \cdot \frac{P \cdot R}{P + R} \]
Summary

• Measuring information diversity
  • alfa-NDCG

• Information near-duplicates
  • Shingling and Jaccard distance
  • Minimum Hashing

• Information clustering
  • k-Means clustering