Abstract

The purpose of this project was to group Wikipedia pages together based on their similarity, thereby reducing the search space for certain types of queries. We analyzed the pages’ characteristics by looking at internal link structure. From this data we identified communities of pages which were likely to be relevant to queries on other pages in the cluster, allowing us to accomplish our task without using any page content or metadata.
1 Introduction

As the world moves on, progress has become encumbered not by a lack of information, but an overabundance of it. When looking for any specific piece of data, we must wade our way through countless numbers of documents which have no relation to our query. Humans have the ability to look at and group data together such that we only search relevant material for relevant data. For example, we would not go to our philosophy texts for information about data structures. We would isolate it down to our computer science texts and from there narrow it further to texts about data structures before we begin doing word by word searches. This obviously makes information retrieval among vast amounts of data a realistic, potentially computable, endeavor.

Our goal is to teach a computer to do the same. Thus, our problem statement: for a set of documents $D$, find a set of communities $C$ for which all members of $C$ are highly related. The relation which groups each $C$ should be some queryable component.

Our method was to analyze provided data about documents to generate relatedness between the documents. We then found groups of documents that were related to each other and called that community of documents a cluster. Membership in such a cluster became an attribute of that document. Thus searching became not a task of finding the internals to all documents, but finding documents related to our query based on their cluster membership and searching those.

2 Dataset

For data we were provided with a large set of XML files from INEX[1]. These files contained pages from Wikipedia, distilled to a combination of link analysis and bags of words. The data came in 6 zip files and was subdivided into four main categories.

The first category consisted of files containing tags and trees. The tag file had lines of tag frequencies, written as:

\[
< \text{documentID} > < \text{treeID}1 > : < \text{tagIDn} > : < \text{frequency} >
\]

The tree file had lines of the tree structure, read as:

\[
< \text{documentID} > < \text{treeID} > < \text{depthfirsttraversal} > < \text{lengthoftheString} >
\]

The second category contained link data. Each of the files representing pages lists which pages the current page linked to. The format of the files was:

\[
< \text{documentID} > < \text{linkID}1 > ... < \text{linkIDn} >
\]
The third category contained entity information. Each file was encoded in
the following format:

\[
\begin{align*}
\text{< documentID >:} & \text{< featureID1 >:} \text{< frequency >} \ldots \text{< featureIDn >:} \text{< frequency >}
\end{align*}
\]

There were also 4 other files:

- entity.tag.freq.id - entity tags sparsely encoded
- entity.tag.stats - CSV of stats about tag features
- entity.text.freq.id - text inside entity tags sparsely encoded
- entity.text.stats - CSV of stats about text features

The last set of files was the File Bag-of-Words and the File Index.

Out of these, we used only the link data. Understanding the purpose and
formatting of each file was essential to discovering some characteristics of our
data set and determining the feasibility of potential approaches with different
parts of the data.

3 Problem Statement

For a set of documents \( D \), encoded in XML with link information, find the
communities \( C \) which consist of highly mutually-relevant documents.

4 Solution

We used fuzzy K-means clustering to group the pages. K-means clustering clus-
ters documents with an iterative two-step algorithm:

- Assign \( k \) random cluster centers
- Repeat until convergence:
  - Assign every point to the nearest cluster center
  - Use the assigned points to calculate new cluster centers

Fuzzy K-means clustering is any K-means that gives continuous probabilities
for a page belonging to a cluster, rather than just 1 or 0. This can be used to
assign a page to multiple clusters with different likelihoods, or to determine a
confidence level when each page is assigned to only one cluster.

We used K-Means clustering to analyze the data with sets of linked pages
being cluster centers. The weight of each link is calculated as the number of
times the page occurs divided by the total number of links in the cluster.
5 Implementation

We went through a few different solution possibilities to get to our current solution. First we tried using medoids as the cluster centers, and figured out the distance by using the link distance from one page to another. During this step we discovered most of the optimizations that we used in the final solution. We hash-indexed the links table for faster lookups, as it was used in every query. We also kept the links in memory, since the faster read time justified the memory use. Still, this method took an unreasonable amount of computation time, with a best-case performance of $n$ source nodes $\times \log n$ breadth-first iterations to reach all other nodes $\times$ constant hash search in the links table per iteration. The optimizations brought the time from 9 days per node to 5 minutes, but with 50000 nodes it was still infeasible.

The next thing we tried to do is use the reciprocal of the number of shared links. This had a few problems that wouldn’t allow this to work. First, the data was not normalized. Since the number of links that a page links to could be 5 for one of the pages and 500 from another, each of those pages sharing 5 links with the center would make them appear to be equally close to it. Another problem that occurred was that it could take a while for this algorithm to run, since we needed to join the (very large) links table to itself. The next problem was that we would get vastly different times to run the algorithm depending on the initial cluster centers that were used. The last problem is that once a cluster got bigger it got “hungry”. Placing more documents in a cluster would give it more links, increasing the number of other documents that would be assigned to it in the next iteration.

Finally, we moved to a K-means approach where the ‘center’ consisted of the most common destination pages. The distance from a page to a cluster center was calculated as 1 minus the sum of the weights of the shared destination pages, where the weights were the number of times each destination occurred, divided by the number of total destination pages in the cluster. The sum of all the weights will always be equal to 1. The new centers were found by counting the occurrences of each destination in the cluster pages’ links and taking the top 100. This solution got us an algorithm that converges and runs in about 20 minutes with 20 cluster centers, a reasonable amount of time.

The algorithm generated two tables as output. The first table contained a row for each document that had the document id, the cluster number, and the distance to the cluster center. This allows us to calculate the mean MSE for

We evaluated cluster performance using the mean squared error (MSE). As this metric heavily penalizes distance from the mean, it has the effect of limiting cluster size unless the clusters are very tight. As a baseline performance measure, we used the MSE of a set of randomly assigned clusters.
the final clusters. The second table gave us the cluster centers and was used to
calculate where to reassign points. With the finished contents of that table, we
could assign "query" pages to clusters and return a limited search space which
is relevant to the input page.

6 Validation and Analysis

We verified the performance of our algorithm by calculating the mean square
error (MSE) for the number of clusters that we choose. To be minimally success-
ful, we would need a mean squared error that performs better than a random
mean squared error calculation.

The random mean squared error calculation was found in three steps. The
first step was to randomly assign pages to clusters. Next we figured out the
cluster centers for each cluster and found the distance to the cluster center from
each point in the cluster. Finally we found the mean squared error. The number
of clusters that we choose was be a manual assessment of many different runs
of varying cluster sizes.

We calculated the mean MSE by taking the normalized MSE for each cluster
in the set of k and summing all of those, then dividing by the number of clusters.
This gives an error figure that is portable across the the number of cluster
centers, and makes the random MSE portable to the others as well.

7 Results

The number of clusters is a user set amount. The more clusters there are
the more calculations have to be done. We started with smaller values for the
number of clusters and slowly increased it, comparing our values to randomly
assigned clusters as we went.

MSE for:

<table>
<thead>
<tr>
<th>Clusters</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>random</td>
<td>.986598</td>
</tr>
<tr>
<td>20 clusters</td>
<td>.765324</td>
</tr>
<tr>
<td>30 clusters</td>
<td>.759208</td>
</tr>
<tr>
<td>40 clusters</td>
<td>.732507</td>
</tr>
<tr>
<td>50 clusters</td>
<td>.699819</td>
</tr>
<tr>
<td>100 clusters</td>
<td>.659879</td>
</tr>
</tbody>
</table>

This means the improvement against random for each k value is:
20 clusters: 22.427%
30 clusters: 23.047%
40 clusters: 25.754%
50 clusters: 29.067%
100 clusters: 33.115%

We used the judging script provided by INEX to transform our clusters into category names based on the training set.

### 7.1 Metrics

ACC = Accuracy (Macro/Micro) - Points correct over number of points

ROC = Area under the receiver operating characteristic (ROC) curve - Likelihood that for any cluster, a random point that should be in the cluster will be given a higher rating than a point that should not be in the cluster.

PRF = F1 Measure

Mean Average Precision = Chance that category prediction assigns a point to the correct category.

Our results:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro ACC</td>
<td>0.96806358974359</td>
</tr>
<tr>
<td>Micro ACC</td>
<td>0.952105722745693</td>
</tr>
<tr>
<td>Macro ROC</td>
<td>0.622642307692038</td>
</tr>
<tr>
<td>Micro ROC</td>
<td>0.632244874095078</td>
</tr>
<tr>
<td>Macro PRF</td>
<td>0.273468974358974</td>
</tr>
<tr>
<td>Micro PRF</td>
<td>0.268778627404238</td>
</tr>
<tr>
<td>Mean Average Precision by document</td>
<td>0.4347</td>
</tr>
</tbody>
</table>

For reference, these are the numbers from XEROX PARC:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro ACC</td>
<td>0.974484615384615</td>
</tr>
<tr>
<td>Micro ACC</td>
<td>0.963074304354917</td>
</tr>
<tr>
<td>Macro ROC</td>
<td>0.748245128205128</td>
</tr>
<tr>
<td>Micro ROC</td>
<td>0.765147235486745</td>
</tr>
<tr>
<td>Macro PRF</td>
<td>0.571497948717949</td>
</tr>
<tr>
<td>Micro PRF</td>
<td>0.600440508010991</td>
</tr>
<tr>
<td>Mean Average Precision by document</td>
<td>0.678560202685951</td>
</tr>
</tbody>
</table>

We looked at the precision from other contestants and found that we usually had about 2/3 the precision.
8 Conclusion

The goal of our project was to improve search times on documents through automated clustering algorithms which analyzed data about the documents being searched. We were successful in implementing a solution which improved on random clustering by a significant degree. Though when gauged directly to the work of others who attempted the same research, our results were lacking. Accounting for our deficiencies in time and resources as well as the limited amount of data we did analyze, our work was quite effective.

While several metrics were analyzed, the most important was the mean average precision, which measured the likelihood that a document would be assigned the correct categories. Our algorithm achieved a success rate of almost 43.5%. Most top competitors achieved a percentage in the mid to high 60’s. Considering that the entirety of our analysis was done on the link level while other projects also used the bag of words information, reaching 2/3 of the precision of other groups was a great achievement.

While we failed to do bag of word analysis, the current structure of our program does not preclude it. Analyzation of the bag of words data could later be incorporated into our distance metric in order to further refine our results without requiring massive changes to the rest of our clustering algorithm. Though one could consider our failure to use all available data a deficiency to our project, our project was so designed that it can become the starting point for further analysis.
9 Bibliography

References


