Incremental Clustering of News Reports

by: Joel Azzopardi & Chris Staff

joel.azzopardi, chris.staff @um.edu.mt
Scenario:

News Event occurs: e.g. Shooting at French Jewish School (Monday 19/3/2012 round 08:00).

As time passes, news reports start being published on that event. E.g. By 15:50, there are over 1500 news report sources on that event as reported on Google News.
The Problem:

Different News Reports:
- written by different authors (each with his/her own point of view)
- may agree with or contradict each other
- may contain unique pieces of information

For a user to learn *all* the information about an event, he/she will have to read through all available news reports.

Many events occur every day, and multiple news sites publish reports on different events. => huge number of reports are published each day.
Proposed Solution: Event-Centric Clustering

Events are clustered into event-centric clusters where each cluster contains reports describing a single event.

Clustering reduces user time and effort.

It allows for Topic Detection and Tracking

It may serve as the initial component in:
- News Fusion Systems
- News Recommendation Systems

In Commerical Aggregators such as Google News, users do not have the ability to choose their own news sources.
Special Requirements

* Dynamic Collection – reports are downloaded continuously

* Stream-lined processing – cannot wait for the entire collection, & cannot perform more than one pass.

* Unknown number of final clusters – new events appear constantly on news.

* Efficiency and scalability – system cannot take too much time and/or resources
Document Clustering refers to the process whereby a collection of documents is clustered in an unsupervised manner into clusters such that documents within the same cluster are more similar to each other than to documents in other clusters [11, 23].

The presence of online sources of data continuously streaming out new documents, such as news sites and blogs, necessitates the use of incremental document clustering systems [17, 14].

The major issue encountered when performing Document Clustering is to perform effective clustering in an efficient manner [14]. New event detection systems always assume that there are enough resources available to perform the necessary computation.
The Document Clustering process typically consists of [13]:

* the extraction of features from the documents;

* the mapping of the documents to high-dimensional space; and

* the clustering of the points within the high-dimensional space.
Document features are usually represented by a set/ subset of their words – the *Bag-of-Words* Approach.

Usually, stop-words are removed, and terms are reduced to their stems using suffix-stripping routines (such as Porter's Stemming Routine [16]).


[9] utilizes the DMOZ to calculate term weights using *TF.IDF*. 
[20] defines two types of similarity measures, namely:

* **Statistical Similarity** – based on term and co-occurrence frequencies (e.g. using Cosine Similarity); and

* **Semantic Similarity** – using distance between terms meanings (e.g. using WordNet).

In most cases, Cosine Similarity measure is used as the distance function.

To calculate similarity between a document and a cluster, the cluster is usually represented using a centroid vector – generally represented as a weighted average of its members' vectors.
2 main clustering techniques [21]:

* **Hierarchical Clustering** – may be Agglomerative (bottom-up), or Divisive (top-down); and

* **Partitional Clustering** – one level of partitioning of data points is created.

**K-Means Clustering**, or a variation of it is one of the most commonly used technique. It starts with $k$ points as centroids, and assigns all points to the nearest cluster. It recalculates cluster centroid and memberships at each pass.

[9] utilizes a variation on K-Means, which uses similarity thresholds.
System Design Aspects

Our issues:
- dynamic collection,
- need to have stream-lined (incremental) processing,
- unknown number of clusters,
- each report can only belong to one cluster.

Standard Hierarchical Clustering algorithms do not allow for incremental clustering.

A variation of the K-Means approach with similarity thresholds is utilized. A new report that is not sufficiently similar to existing clusters is considered to be a new event.
System Design (cont.)

Use of *Bag-of-Words* representation with stop-word removal and suffix stripping.

Use of Cosine Similarity.

Use of Global Inverted Index to calculate $TF.IDF$ weights: If non-existent – index the first 70 reports (40% of 'international news' reports from 9 sources in 24 hrs)

Cluster centroid is represented by an average index:

$$w_{t,c} = \frac{\sum_{d \in D} (w_{t,d})}{|D|}$$
Methodology

News reports are downloaded periodically from a number of different RSS feeds.

Downloaded news reports are 'cleaned' from surrounding HTML and the content is stored as a text file.

Reports are indexed and terms are weighted using the $TF.IDF$.

The $IDF$ is calculated based on the current state of the Global Inverted Index.

The Global Inverted Index is updated with the terms from the current news report.
The cosine similarity of each report with the existing clusters is calculated. When a cluster is found to have a similarity greater than a pre-defined threshold, it is placed within that cluster.

If no such cluster is found, a new cluster is created to hold that report.

Clusters which have not been enlarged for some time are 'frozen'.
Evaluation

We utilised three different datasets of clustered news reports, and we compared how close the clusters produced by our system are to the clusters in these datasets:

* **Reuters-RCV1** corpus filtered from news reports belonging to more than 1 category

* **Yahoo! News** collection downloaded via RSS in December 2010 and January 2011


Metrics used: **Recall, Precision** and **F-measure**.

Baseline – standard K-Means with number of clusters to create given beforehand.
High Recall and low Precision

Recall scores of some categories was 1, or nearly 1 – the produced cluster is a very large cluster that encompasses the entire, or most of the reference cluster.

Since all (or most) of the reference cluster’s documents are in that large produced cluster, the recall is 1 (or nearly 1).

The large produced cluster contains also a large number of documents that do not form part of the reference cluster => very low precision.
Evaluation

Yahoo! News Results

<table>
<thead>
<tr>
<th></th>
<th>Recall</th>
<th>Precision</th>
<th>FM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our System</td>
<td>0.3851</td>
<td>0.2690</td>
<td>0.2159</td>
</tr>
<tr>
<td>Baseline System</td>
<td>0.4387</td>
<td>0.3018</td>
<td>0.3091</td>
</tr>
</tbody>
</table>

Better results than in Reuters-RCV1, but still low precision and F-Measure values.

Reason: *Yahoo! News* categories are more distinct in vector space – there is less overlap between the *Yahoo! News* categories in vector space than between the *Reuters-RCV1* categories.

As in *Reuters-RCV1*, the baseline system performs better than our system.
The 'Modified System' is a modification of our system whereby documents are clustered with the most similar cluster rather than the first 'similar' cluster encountered.

Much better results than in *Reuters-RCV1* and *Yahoo! News* – the clusters in this corpus are distinct from each other in vector space.

Our System performs much better than the baseline.

The Modified System performs slightly better than our original system.
Conclusions

Our system is very effective when clustering documents into highly specific clusters (such as event-centric clusters), but performs rather poorly into more general categories.

Performance difference between our original system and modified system is very small => more worth it to use original system since it is faster.

Results obtained may imply that our algorithm is similar to the one used by Google News – this could not be confirmed.

Clustering process described is fast – every news report is clustered in less than 1 second on a Linux 2.4GHz Quad-Core system and 8GB RAM.