Automatic Clustering of News Reports

Joel Azzopardi

Department of Artificial Intelligence, University of Malta

Abstract. The automatic clustering of news reports from various web-based news sites into clusters according to the event they cover serves not only to facilitate browsing of news reports by a users but may also serve as an initial stage in other complex systems such as Multi-Document Summarization systems or Document Fusion systems. In contrast to the usual scenarios of document clustering whereby the document collections are static or quasi-static, news sites are continuously updated with reports concerning new events. Here, we present a News Report Clustering system which is able to receive a stream of news reports which it clusters on the fly according to the event they cover. New clusters are automatically created as necessary for news reports which are covering 'new', previously unreported events. We compare the results of our system to the results produced by a standard K-Means clustering system, and we show that our system performs significantly better than the standard K-Means system even though the K-Means system was supplied with the correct number of clusters that should be produced. In fact, our clustering system obtained an average of 11.95% better recall, 28.68% better precision and 0.89% less fallout than the standard K-Means clustering system.

1 Introduction

Whenever an event occurs, numerous news reports appear on a great number of different news sites on the World Wide Web (WWW) within minutes of the occurrence of that event. Every news agency has its own reporters on the field of action and its own sources. Therefore, every news report may contain information that is unique — i.e. found only in that report.

A reader interested in a particular event will search for different reports on that event to learn as much information about that event as possible. However, it is time consuming for a user to search every news site for reports covering a particular event. The presence of an automatic document clustering system will make this task much easier each since such a system can cluster together those reports from different sources which are covering the same event.

The use of such a system is beneficial not only to facilitate the browsing of a news report, but can also be used as a component within a system with more complex goals such as multi-document summarization and document fusion.

The usual methods used for document clustering, such as *Hierarchical Clustering* and *K-Means Clustering* require the collection on which clustering is to be

performed to be static. Therefore, these methods are not feasible for the clustering of news reports since the collection of news reports should be continuously updated with new news reports, and every new news report needs to clustered immediately for the system to work operationally. Moreover, *Hierarchical Clustering* and *K-Means Clustering* require the number of clusters to be known beforehand. When news reports are being received continuously, there can be no way of knowing beforehand the number of different events that are occurring.
To address these issues, we present a system which performs the automatic clustering of news reports. Our system produces a separate cluster for each event that is being covered within the news reports, and all news reports that are covering the same event are clustered together within one cluster. New news reports are read continuously from a number of different sources, and our system is able to detect when a new event is being reported and create new clusters accordingly.
We compared the clusters produced by our system to the clusters produced by a standard k-means system. We found that our system performed significantly better than the standard k-means system, even though the k-means systems had the number of clusters to produce for each corpus specified beforehand.
The remainder of this paper is divided as follows — in section 2 we give a brief overview of other document clustering systems. Section 3 contains a description of our approach to the document clustering, and the justification to this approach. Then in section 4 we describe the methodology of our system. The following section (section 5) contains a description of how we performed our evaluation and section 6 presents the results obtained. Finally, section 7 contains a brief discussion of the results obtained and the conclusions we drew from our results.

2 Related Work

The main purpose of document clustering is to generate hierarchies and facilitate browsing from a document collection [HP96]. Moreover, Document Clustering is also used to assist in Information Retrieval [HP96,SKK00,Sal72] — this is based on the proven fact that documents which are relevant to a particular query are found to be more similar to each other than to documents not relevant to that query [HP96,APR99].
The derivation of the appropriate set of categories into which a document collection is to be clustered is essential in Document Clustering so as to simplify the classification task [BB63]. This set of categories should be determined by the document collection itself, or by the system's purpose. [LA99] stresses for the need to have a system which can discover and approximate topic hierarchies using unsupervised clustering methods.
The phases in Document Clustering are [LA99]:

- 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.

The document features are usually represented by the set or a subset of the words they contain [Sal97,BB63,LA99,Sal72,VF95,APR99,TK05]. [BB63] advocates the use of pre-selected terms to represent each document. On the other hand, [Sal97,LA99,Sal72,VF95,TK05] extract all the terms from the documents to act as content-representatives — albeit using some filtering sometimes, such as using only the highest n terms. Besides the use of single terms as document labels, [Sal97] also suggests the use of term phrases.

[Sal97] claims that the importance of a term as a representative of the content within that directory is related to the occurrence frequency of that term within the document (or document excerpt) and the occurrence frequency of that term within the entire document collection. In view of this, the importance of a term may be calculated using the *Inverse Document Frequency* (IDF), which is the ratio of the term occurrence frequency within the document in question to the occurrence frequency of the term over the whole document collection. This IDF is also used for term weighting by [APR99,LA99,Gul05]. [Gul05] calculates the term weights by utilizing the TF.IDF measure that is centered on the DMOZ[1] Categories. The advantage of this approach is that one does not need to have the entire document collection at hand to weight the terms which will be used to represent the documents.

The occurrence frequencies and *IDF* information of each term may be stored in an Inverted Index. An Inverted Index is a sorted list of terms, and along with each term other related information such as the occurrence frequencies of that term and its *IDF* information. Inverted Indexes are used in standard document retrieval and they also include a postings list for each term — i.e. a list of links to the occurrences of that terms within the document collection [Sal72].

Besides the IDF, [Sal97] also describes the term specificity in the context of the representation of documents in high-dimensional space. If broad terms are used to represent documents, they will lead to very small distances between the points representing the documents in the high-dimensional space. On the other hand, if the terms used to represent documents are too specific, the points in the high dimensional space will be too far apart from each other. In view of all this, [Sal97] defines the *Term Discrimination* measure which is the difference occurring in space density if a term which was previously considered to be representing a document, is not considered anymore.

An alternative method of document representation is by using *Lexical Chains* [SC01]. A Lexical chain is a set of semantically related words found within a text. To build lexical chains representation of a document, each term within that document is processed chronologically, and it is added to an existing chain or made the seed to a new chain. The criteria used for adding a term to an existing chain is by identifying a semantic relationship (using WordNet[2]) between the term in question and the chain's seed term, or by establishing a co-occurrence relationship within close proximity of that term with the chain's terms. By analyzing

[1] Open Directory Project — *http://www.dmoz.org*
[2] *http://wordnet.princeton.edu/*

the lexical chains, one can identify those chains which have the most members as being representative of the more salient terms.
After the documents representations are constructed, the next step would be that finding the similarities between the different documents. In [LA99,APR99], this is done using Cosine Similarity (refer to [Sal71]). Quite similarly, [BB63] performs correlation between matrices of the index terms' occurrence frequencies. [SC01] finds document similarities by comparing the documents' lexical chains together. According to [SKK00], there are 2 main clustering techniques — namely:

- **Hierarchical Clustering** — this technique produces *hierarchies* and is further split into:
  - *Agglomerative* — whereby we start with each point being in a seperate cluster, and at each step, the most similar pair of clusters are merged together, and
  - *Divisive* — whereby we start with all the points being in a large single cluster, and at each step we split the cluster such as to maximize the intra-cluster similarity.
- **K-Means Clustering** — whereby we start with k points as the initial cluster centroids, and assign all the points to the nearest centroid. Then, a number of passes are made whereby the cluster centroid is recalculated and the cluster membership of each document is also recomputed. The application of this technique is also discussed in [LA99,Gul05,HP96,SC01,Sal72].

[Gul05] utilizes a variation of the K-Means Clustering approach whereby similarity thresholds are used to warrant cluster membership, and the number of clusters is not known beforehand. The use of similarity thresholds to warrant distances is also discussed in [LA99,SC01,HP96]. [HP96] also uses a junk cluster which will contain those documents that can not be clustered.
In contrast to utilizing the similarity between each document and the cluster centroids to decide if that document warrants membership within that cluster, [APR99] describes two other membership policies:

- *Single Link* — whereby a document is considered to be part of cluster if it is related to at least one document within that cluster,
- *Average Link* — whereby a document is considered to part of a cluster if it is related to at least the average number of documents within that cluster.

In contrast to the Clustering Methods described above, [Gul05] discusses a news classification system which uses training but in a dynamic manner. The problem of [Gul05] is to classify news reports as they are received from the multiple sources. Now some news reports have classification information defined explicitly contained within themselves — for example an article may be marked to be part of "Sports News" or "U.S. News" or "British News". When such classification information is identified, the system switches to training mode. Then, when news reports are received which have no classification information contained in them, their classification is decided based on the training the system has incurred with the other news reports.

In the usual scenario in which Document Clustering is analyzed, there is a static document collection which is to be clustered. However, in the cases where the document collection is dynamic — i.e. documents are being added, and/or others are being retired (removed from the document area) continuously — things get more complicated. A case in point is when we have a system which is working on news reports which are continually being receiving from streams such as from RSS feeds, such as the News Search Engine system described in [Gul05]. As more documents are added to clusters, eventually a complete reorganization of clusters will be needed [Sal72,LA99]. Furthermore, [VF95] shows that when using the normal weighting schemes (using the vector space IR model), term weights updates are expensive — in fact, adding a single document can affect a large part of the Inverted Index.
The suggested solutions to this problem include:

- The use of pre-computed term weights without any update to these weights ([VF95]). This approach is also utilized by [Gul05] which uses term weights calculated by parsing the DMOZ. [Sal72] also discusses this option, whereby new documents can be added to clusters without changing the cluster representations.
- Updating only the weights of existing terms in the cluster profiles but without introducing new terms to the cluster profiles [Sal72].
- Keeping the Inverse Document Frequency (IDF) information separate from the document term weights, and IDF weighting is only applied to query terms, thus avoiding the recalculation of document term weights when a new document is introduced to the collection [VF95].
- Updating the Term weights and the terms' IDF information only intermittently [VF95].

3 Our Approach

As we already mentioned in section 1, we are presenting here a system which performs the automatic clustering of news reports. Within our system, news reports are being continuously downloaded from a number of different news sites. Our clustering system processes each report and clusters it with those other reports which are covering the same event, or otherwise clusters it into a new cluster on its own if the event covered by that report has not been covered any other report yet.
The main issues within our system are that:

- the document collection is dynamic since new news reports are being downloaded continuously and presented for clustering,
- since new news reports are being downloaded continuously, the news reports must be categorized on the fly since at no point will the downloading of news reports stop to allow for the clustering to proceed, and
- the number of clusters can not be known beforehand — when a news report describing a 'new' event appears, a new cluster must be created automatically for this report.

The clustering approach utilized by our system is a variation of the *K-Means Clustering* approach, and uses a similarity threshold between the individual document and the cluster centroid to warrant cluster membership, as described in [Gul05,LA99,SC01,Gul05,HP96]. In our opinion, *Hierarchical Clustering* is not appropriate for a dynamic system such as ours since hierarchical clustering will require the entire document collection to be present before it can start clustering the documents, and the knowledge before hand of the level where one needs to stop the clustering procedure. The K-Means approach has been adapted to work on a dynamic collection — new clusters are created and necessary, and old clusters are not considered for processing anymore if they have not been modified after a certain period of time.

In this way, our system can be adapted to perform new topic detection and tracking as well. If a major event occurs in the news such as a terrorist attack on a city in the United States, all reports covering this event will be clustered together in a single cluster. Moreover, news reports which are issued later on in time but contain more information that has been uncovered are also clustered within this same cluster. On the other hand whenever a report appears which is covering a new event (hence a new topic), a new cluster is created for that report.

The size of each produced cluster depends on the amount of reports issued that are related to the event being covered by the reports within that cluster. The number of reports issued is dependent on various factors such as the significance of the event in question, and also certain events such as the uncovering of a new piece of information on that event also triggers new interest in that event. The temporal effects of news reports are also discussed in [yGGLL01a] and [yGGLL01b].

To represent each news report in high-dimensional space, the use of pre-selected terms requires additional effort and is not feasible for us since our system processes news reports as they are read from the feeds. Therefore, to simplify matters, each news report is represented using all the terms contained within it, and those terms are weighted using the TF.IDF measure. The use of knowledge bases, as suggested in [SC01], is avoided to adhere to surface-based methods.

In our opinion, the usage of the *Single Link* policy will tend to produce clusters that form 'chains' rather than actual clusters. A news report may contain information relevant to more than one topic. Therefore, by using the *Single Link* clustering policy, one can end up with unrelated documents within the same cluster. In our system, the similarity between a news report and a cluster is quantified by the cosine similarity between the term index of that news report and the cluster centroid index. The cluster centroid index is representative of the 'average' index of the documents within that cluster.

The use of an external document collection to calculate the term weights, as suggested in [Gul05] will reduce the dependence of the Inverted Index on the current state of the document collection. Since in our case, the collection of news reports is changing continuously, such an effect would be desirable. We apply a fairly similar approach — each document is represented using term weights which are

calculated based on the current state of the document (news report) collection. A global index is maintained which keeps track of the occurrence of each term over the entire collection, and this global inverted index is updated with the processing of each document. Meanwhile, a document index is constructed for each document which stores the occurrence frequency of each term within that document. Whenever similarity between documents needs to be calculated, the term weights are calculated on the fly — thus no re-calculation is performed for terms which do not appear in either document.

4 Methodology

In this section, we describe how we implement the approaches which we discussed in the previous section. Within our system the documents (news reports) are processed — i.e. indexed and categorized — one by one, and once a document has been categorized, the system moves on to the next document. It does not perform any cluster re-organization. The justification behind this is that the incoming stream of news reports (documents) for clustering never stops.

Within this phase, the documents are first tokenized and each term is then stemmed using the Porter Stemming Algorithm [Por97]. The stemmed versions of these terms, with the exception of stop words, are placed into an individual index for each document.

The terms within each document's index are weighted using the *TF.IDF* measure, whereby the *TF* is taken to be the Term Occurrence frequency of the term within that document, and the *IDF* is taken to be the Inverse Document Frequency of that term over the entire document collection in its current state. The Occurrence Frequencies of all the terms over the entire document collection are stored in a global index, and this global index is dynamic — whenever a new document has been presented for processing and has been indexed, the global index is updated with the terms from that document.

For the case where the system has just started processing its first documents, a special procedure is performed for the weighting of those documents' terms. Before starting the categorization process, the system waits for the first 70 documents to be available for processing, and initializes a global index with the occurrence frequencies of these documents' terms within this initial collection of 70 documents. Then whilst processing these 70 documents, the global index is not re-updated.

We chose to make the system wait for the first 70 documents to initialize the global index since 70 is approximately 40% of the entire set of document downloaded within the first 24 hours of the start of the system. We found out that when the system is started (i.e. there are no downloaded reports), the system downloads approximately 180 different news reports in the first day when using 4 different sources. In our opinion, 40% of the entire set of documents downloaded in the 24 hours from 4 different sources provide ample indication of which terms are important, and which terms are common throughout the entire collection.

After the document being processed has been indexed, it is clustered. The clustering is performed by calculating the cosine similarity between the document in question and the centroids of each existing cluster. Once a cluster is found to have a similarity higher than a pre-defined threshold with that document, the document is placed within that cluster. If no cluster is found to have a similarity which exceeds the similarity threshold, a new cluster is created with that document as its first member.
The cluster centroid is represented by an index of the stemmed versions of all the terms (excluding stop-words) which appear in those documents which are members of that cluster. The weight of each term within the cluster index is set to be the average weight of that term within the documents in that cluster. More specifically:

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

where $w_{t,c}$ refers to the weight of term t within cluster c, D refers to the set of documents within the cluster, and $w_{t,d}$ refers to the weight of term t within document d.
Since news reports are being continuously received and clustered, the amount of clusters is constantly growing. Since prior to classifying a document within a cluster on its own, it must be compared to all the existing clusters, as more news reports are clustered and new clusters are created, the clustering procedure will start to take longer. Therefore, a system where the clusters are continuously being created, but never removed, would not be scalable.
To resolve this issue, we utilized the concept of "freezing" old clusters. This means that clusters which have not had new members since a period of time are "frozen", and incoming documents are not compared to them at all. They are assumed to be describing events whose "influence" has now passed and are not any more of "interest" within the world of news broadcasting. The identification of "frozen" clusters is performed by the system, which traverses the list of active (unfrozen) clusters every period of time.

5 Evaluation

For the evaluation of the Document Clustering System, we perform clustering on a set of corpora of news reports and then compare our results with how *Google News*[3] clustered these same reports. In our opinion the clustering produced by Google News may be seen as the Gold Standard for news report clustering. From personal experience, the news reports clustered together in Google News are always related to each other — i.e. are always covering the same event. Therefore, we assume that the closer the clusters produced by our Document Clustering system are to the clusters produced by Google News, the more effective our Document Clustering system may be considered to be.

[3] *http://news.google.com*

A single corpus is built by downloading the reports that appear within Google News in each of the news sections that appear on the Google News front page — namely *World News*, *U.S. News*, *Business News*, *Science & Technology News*, *Sport News*, *Entertainment News* and *Health News*. This is done by downloading the RSS feeds for each of the afore-mentioned section. Each record within these RSS feeds refers to a cluster of news reports and contains links to 4 or 5 reports, as well as a link to the Google News page which displays all the related news reports for that cluster. We downloaded those reports which are referred to directly within the RSS record.

Each corpus represents a 'snap-shot' of the Google News Clusters at a particular time. We built the multiple corpora by downloading the RSS feeds every hour and building a new corpus for each time the download is performed. We built 49 corpora in this way.

Since each downloaded news report is in HTML format (as it was displayed in its original web-site), we filter each report to remove the HTML code and surrounding text which are not part of the actual report text.

When attempting to download a report which was referenced in the Google News RSS feeds, there is no guarantee that the report is available for downloading. Reasons for such cases may be that the report is not available anymore, or that the news site providing that report requires a subscription to enable readers to access its reports. Therefore, when we download the reports, inevitably we download documents which instead of containing the actual news reports contain messages such as "This report is no longer available.". To remove such documents from our corpora, we implemented a filter which calculates the average of the Inverse Document Frequencies of all the terms in the each document (excluding the stop words). Those documents whose average Inverse Document Frequency score falls below a particular threshold (in our case, this was set to 1.7), were removed from the corpora. We set this threshold to 1.7 after trial and error to see which value generates the best result — i.e. it removes as much "junk" as possible without removing good reports.

The evaluation of our Document Clustering procedure was performed separately for each of the news reports corpora. Each Google News cluster was compared to the most similar cluster produced by our Document Clustering system. Then, for each pair of such clusters the *Recall*, *Precision* and *Fallout* values were calculated.

To obtain a baseline measure, we implemented a clustering system which uses the standard *K-Means* method to cluster the reports. In this baseline system, the documents (news reports) within each corpus are indexed. The index terms are stemmed (using Porter's stemming algorithm) and weighted using the TF.IDF measure. The IDF of each term is calculated relative to the term occurrence frequency within the entire corpus. The cluster centroids are represented by an 'average' inverted index, and the cluster centroids are updated only at the end of each pass. The clustering stops either when convergence has been reached (i.e. a pass has been performed which did not modify any cluster), or otherwise when 10000 passes have been made. The clusters produced by this baseline clustering system are evaluated in the same way we evaluated our clustering system.

The results obtained both by our clustering system and by the baseline system are presented in section 6.

6 Results

This section presents the results obtained for the evaluation described in the previous section (section 5. We calculated the *Recall*, *Precision* and *Fallout* values for each data corpus for our clustering system as well as for the baseline system. Figures 1, 2 and 3 show the recall, precision and fallout results obtained.

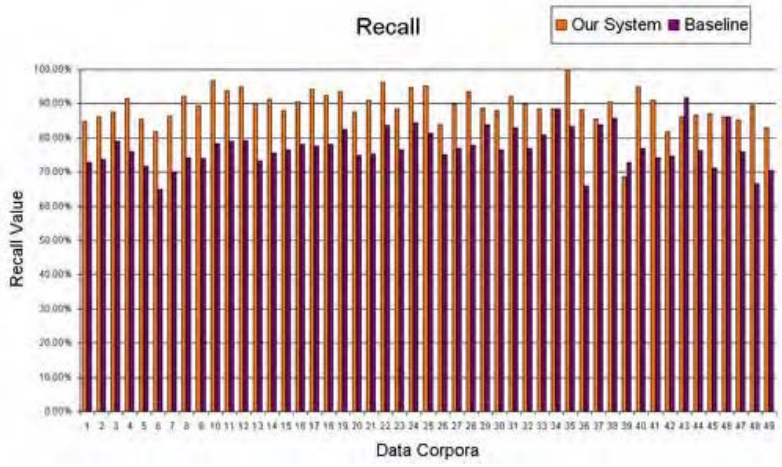

Fig. 1: Recall values

Our system obtained 89.21% average recall whilst the baseline system obtained 77.26%. This means that our system obtained 11.95% better recall. As regards precision, our system obtained 90.17% average precision whilst the baseline system obtained 61.49% — an improvement of 28.68% of precision from our system's side. Our system obtained 1.02% fallout rate whilst the baseline system obtained 1.91% — a difference of 0.89%.

7 Conclusion

The results presented in section 6 show that our clustering system performs significantly better than the standard K-Means clustering system. With a couple of exceptions, our system obtained better recall and precision in all corpora, and had lower fallout as well. One has to bear in mind that the baseline K-Means clustering system was provided with the number of news clusters that should be created beforehand whereas our system did not possess and use this information.

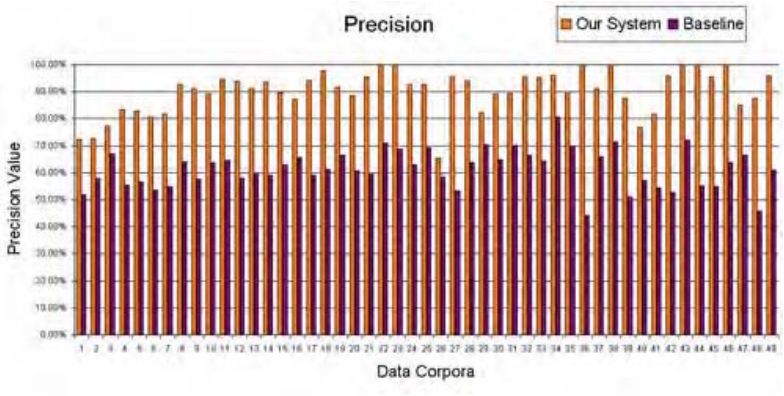

Fig. 2: Precision values

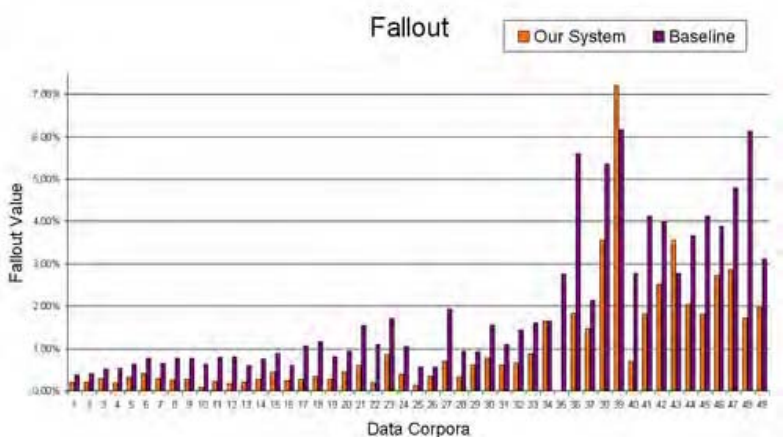

Fig. 3: Fallout values

Nevertheless, our system produced better results. This shows that our system is able to detect the cluster set very well. Moreover, our clustering system is also more efficient than the standard K-Means system since our system does not perform any cluster re-organization, and each report is only processed once.
Besides producing better results than the baseline clustering system, our Document Clustering system also compares pretty well to the Google News clustering system which we consider to be the gold standard of news clustering. There is an instance — corpus 35 — where the recall is 1.00 for this entire corpus of data. This means that for this data corpus all the reports which should have been clustered together were in fact clustered together. There are also various cases where precision is 1.00 — corpora 22, 23, 36, 38, 43, 44 and 46. This means that for these corpora, our system produces clusters which are equivalent to, or are sub-sets of the clusters produced by Google News.
The results also show instances where our system did not perform so well — for example corpus 39 has 0.688 recall, corpus 1 has 0.725 precision. The main reason behind such instances is the presence of documents which do not contain an actual news report (due to erroneous download, or the report not being available anymore). The report filter (described in section 5) does not manage to remove all such reports. When processing such documents, our Document Clustering system places such document into clusters of their own. Obviously, Google News does not have the equivalent of such clusters.
Another reason for the occurrence of some low recall and precision values is that some news report documents contain more than 1 news item in them. For example, in the case of breaking news, a single document may contain 5 different news items where each item is covering an event totally different from the events covered by the other news items within that same document. In Google News, each news item is considered seperately, and the same news document may be forming part of different clusters. Our Document Clustering system assumes that each document contains only one news item. Therefore it performs poorly when it encounters such documents.
The results obtained in our evaluation show that our News Report Clustering system produces news report clusters which are very similar to the clusters produced by Google News, and that it performs significantly better than a standard K-Means clustering system. When one considers that our system is able to work on a dynamic collection — i.e. it reads reports from a news stream and clusters them on the fly — it shows that our News Report Clustering system performs a satisfactory job, and it can be used as a reliable component in a News Web Portal similar to Google News, or as a part of a more complex system.

References

[APR99] Javed Aslam, Katya Pelekhov, and Daniela Rus. A practical clustering algorithm for static and dynamic information organization. In *SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 51–60, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.

[BB63] Harold Borko and Myrna Bernick. Automatic document classification. *J. ACM*, 10(2):151–162, 1963.

[Gul05] A. Gulli. The anatomy of a news search engine. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 880–881, New York, NY, USA, 2005. ACM Press.

[HP96] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 76–84, New York, NY, USA, 1996. ACM Press.

[LA99] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22, New York, NY, USA, 1999. ACM Press.

[Por97] M. F. Porter. An algorithm for suffix stripping. In *Readings in information retrieval*, pages 313–316, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[Sal71] G. Salton. *The SMART Retrieval System — Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[Sal72] Gerard Salton. Dynamic document processing. *Commun. ACM*, 15(7):658–668, 1972.

[Sal97] G. Salton. A blueprint for automatic indexing. *SIGIR Forum*, 31(1):23–36, 1997.

[SC01] Nicola Stokes and Joe Carthy. First story detection using a composite document representation. In *HLT '01: Proceedings of the first international conference on Human language technology research*, pages 1–8, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[SKK00] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[TK05] Hiroyuki Toda and Ryoji Kataoka. A clustering method for news articles retrieval system. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 988–989, New York, NY, USA, 2005. ACM Press.

[VF95] Charles L. Viles and James C. French. On the update of term weights in dynamic information retrieval systems. In *CIKM '95: Proceedings of the fourth international conference on Information and knowledge management*, pages 167–174, New York, NY, USA, 1995. ACM Press.

[yGGLL01a] Manuel Montes y Gomez, Alexander F. Gelbukh, and Aurelio Lopez-Lopez. Discovering ephemeral associations among news topics. In *IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*, pages 25–30, 2001.

[yGGLL01b] Manuel Montes y Gomez, Alexander F. Gelbukh, and Aurelio Lopez-Lopez. A statistical approach to the discovery of ephemeral associations among news topics. In *DEXA '01: Proceedings of the 12th International Conference on Database and Expert Systems Applications*, pages 491–500, London, UK, 2001. Springer-Verlag.