

Topic Categorization of RSS News Feeds

Project Team

Bhushan Pendharkar
Pooja Ambekar
Prajakta Godbole
Sanket Joshi
Srushti Abhyankar

Abstract

Plethora of information in the form of news gets produced at an incredible rate with the rapid growth of World Wide Web and electronic information services. In order to remain updated of the latest news articles many users subscribe to various RSS feeds. However, many a times this information is scattered across various news sources and spans more than one domain. Our system provides a single RSS feed that presents all the news items from various different news sources and groups them into categories. This would save a lot of user's time which he would otherwise spend in visiting various news sites and finding top news of his category of interest.

Our project aims at processing four RSS feeds (representing four news channels) and obtaining a single, well-categorized output feed. In this report, we have discussed the various implementation specific details, the algorithms, the advantages and the limitations of our project as well as the challenges that still remain to be resolved. We have also presented the user evaluation method and test results of our system. Finally, conclusions are drawn and research directions identified.

Keywords

RSS, Ontology, Stop-words, Conflation Algorithm

Introduction

A lot of people like to view and analyze news from various news sources. These people are therefore likely to subscribe to RSS feeds from many news sources. Many a times, people are interested only in the top news stories of their categories of interest. Therefore, the users have to scan through all the top news stories in order to get to read stories of his/her interest. For example, a user interested in sports related top news stories has to go through all the top news stories from various channels and their time spent in analyzing news from the multiple sources. We therefore identified the need of bringing together news from various sources and categorizing them and presenting them to the users as a single news feed. The user can then subscribe only to this news feed as against subscribing to multiple feeds. The system accepts the RSS documents of the different news sources acting as an input to the system.

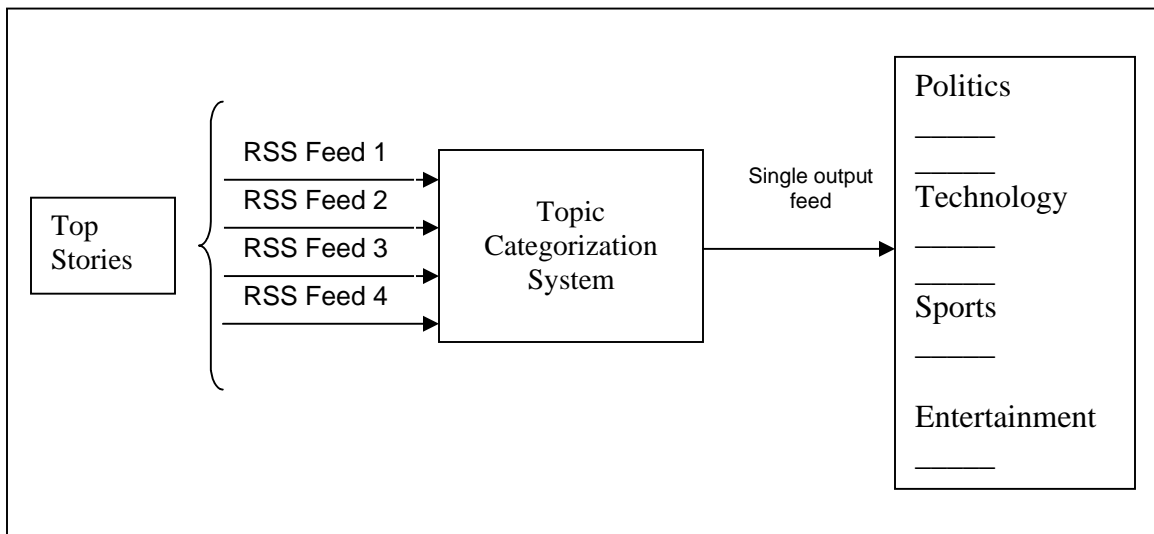


Figure 1: Project Overview

Implementation

An RSS file for news usually contains elements such as channel, item, title, description, etc. As most of the information content is present in the 'description' element of the XML document, we primarily make use of this element in our project. We have used RSSUtilities jar[1] to parse the RSS feeds in order to retrieve the contents of the description tags. The GUI allows the user to specify maximum 4 URLs in order to subscribe to the corresponding RSS feeds. In this project, the number and the scope of the URLs would be predefined as follows.

http://rss.cnn.com/rss/cnn_topstories.rss

<http://www.npr.org/rss/rss.php?id=1001>

http://hosted.ap.org/lineups/TOPHEADLINESONLY-rss_2.0.xml

<http://www.foxnews.com/xmlfeed/rss/0,4313,0,00.rss>

However, the system can function for any RSS feeds and is not restricted to the predefined URLs. The categories predefined for this project include Entertainment, Sports, Technology and Politics. We have created the ontologies[3] corresponding to each of the categories. These are pre-computed as follows:

1. Training data for each of the categories is collected from a wide variety of news channels like Reuters, CNN, US News, NPR, Fox News etc
2. The training data is cleaned using the Conflation Algorithm[4] explained above in order to get the content words representing the ontologies.
3. Thus four different ontologies representing four different news categories (Technology, Entertainment, Politics, Sports) is formed.

Vector Space Model

Each document is represented by a vector in a high-dimensional space, with as many dimensions as the number of terms. Each term has a weight, in the most general case, a frequency associated with it, which models how important the corresponding word is deemed to be to explain the content of the text. Texts whose document vectors are close to each other are considered being similar in content. These words in a document are

assumed to be a reasonable representation of the content of that document. There is no information of the order of the words.

Cosine Similarity Measure

The similarity is measured in terms of cosine similarity. The reason why we use cosine similarity, is that this measure is not affected by the size of the documents. It just considers the normalized document vectors. The similarity between two vectors is the dot product of the two vectors, ie. cosine of the angle between the two vectors.

Spherical k-means clustering

We have used k-means clustering using cosine similarity. It is also called spherical k-means, where each document is represented as a vector of word occurrences ie. Vector Space Model. Each vector is L2 normalized ie. each vector represents a point on a unit sphere in very high dimensional space (where each term is a dimension). The centroid is also on the unit sphere. The text clustering has some advantages and disadvantages:

Advantages:

- It is data-driven, so reflects results according to the collection of input documents.
- Get overview of main themes

Disadvantages:

- Variability in quality of results
- Not good at differentiating homogeneous collection
- May mismatch user's interests/perception

The time complexity of the K-Means algorithm is $O(knI)$, where k is the number of clusters, n the number of input document vectors and I the number of iterations (which is dependent on the stopping criterion).

Assumptions:

- Number of clusters is assumed to be 4.
- Random seeds are chosen for the algorithm.

Algorithm

1. User selects news sources (corresponding to rss feeds) from 4 options available on the GUI. The RSS feeds will be top stories of different news channels. Thus the feeds will consist of news from the categories like: sports, politics, entertainment and technology.
2. RSS feeds are parsed using the RSS parser[2] from the RSSUtilities library.
3. All the parsed descriptions (description is an element of an RSS file that contains a summary of the news story) are grouped together into an object map.
4. These descriptions are then input to the conflation algorithm. Following are the steps of the conflation algorithm:
 - a. Stop words or fluff words are removed from the description.
 - b. Stemming is carried out on the remaining terms to get the root words.
 - c. Frequency of every term within the description is computed. (Frequency is the number of times a particular term occurs in one description)

Let us denote the collection of terms (content words) in a description by *term-list* and the corresponding frequency values by the *frequency-list*.

5. A normalized frequency list is computed corresponding to every frequency list using the formula:

$$f1/\text{sqrt}(f1^2 + f1^2 + \dots + fn^2),$$

$$f2/\text{sqrt}(f1^2 + f1^2 + \dots + fn^2),$$

till

$$fn/\text{sqrt}(f1^2 + f1^2 + \dots + fn^2)$$

Where:

n = number of content words in one description

f1, f2,..., fn = corresponding frequencies of the n content words

6. These term-lists and the normalized frequency-lists are given as an input to the K-means algorithm:
 - a. Randomly pick 4 document vectors (term-lists) from the document vector collection and consider these to be the initial seeds of the 4 clusters.

- b. The remaining term lists are assigned to one of the 4 clusters by computing their cosine similarities with all the 4 centroids. The document vector is assigned to the cluster whose centroid has maximum cosine similarity with that document vector. We compute the cosine similarity as follows:

$$\frac{(f_1 * f_1' + f_2 * f_2' + \dots + f_k * f_k')}{[\sqrt{(f_1^2 + f_2^2 + \dots + f_n^2)} * \sqrt{(f_1'^2 + f_2'^2 + \dots + f_n'^2)}]}$$

Where: $(f_x * f_x')$ in the numerator represent frequencies of the terms common in the centroid and the term list under consideration. And the set (f_1, f_2, \dots, f_n) in the denominator represents the frequencies of all the terms in the centroid and the set $(f_1', f_2', \dots, f_n')$ represents the frequencies of all the terms in the term list under consideration.

- c. Every term-list will be assigned to the cluster whose centroid has the maximum cosine similarity with it.
- d. Once we assign all the document vectors to clusters, centroids are recomputed and marks the beginning of the next iteration.

Suppose t_1, t_2, \dots, t_n are the distinct terms of all the term-lists in a cluster. Then the centroid of that cluster will contain all these distinct terms and their corresponding frequencies will be computed as follows:

$$R = [f_1(t_1) + f_2(t_1) + \dots + f_x(t_1)] / x$$

where: x is the number of terms in the cluster and

$f_k(t_1)$ is the frequency of t_1 in term-list k

R will represent the frequency of term t_1 in the centroid. Similarly the frequencies for the rest of the terms will be computed.

- e. The steps b to d are repeated until either
- i. the centroids converge or
 - ii. the number of iterations equals 3 (empirically adjusted)

7. The output of K-means will be 4 clusters.

8. The final centroids will be compared with each of the 4 ontologies (Technology, Politics, Entertainment and Sports) and the corresponding clusters will be labeled with the most similar ontology.

9. Finally, the output will be presented to the users in the form of different news categories as labeled.

The project is implemented using Java, JDK 5.0 using Eclipse SDK 3.2.1 as the Java Editor. It uses external RSSUtils.jar as an external jar file as well as the Apache Lucene[5] package to implement the porter stemmer.

Class Diagram:

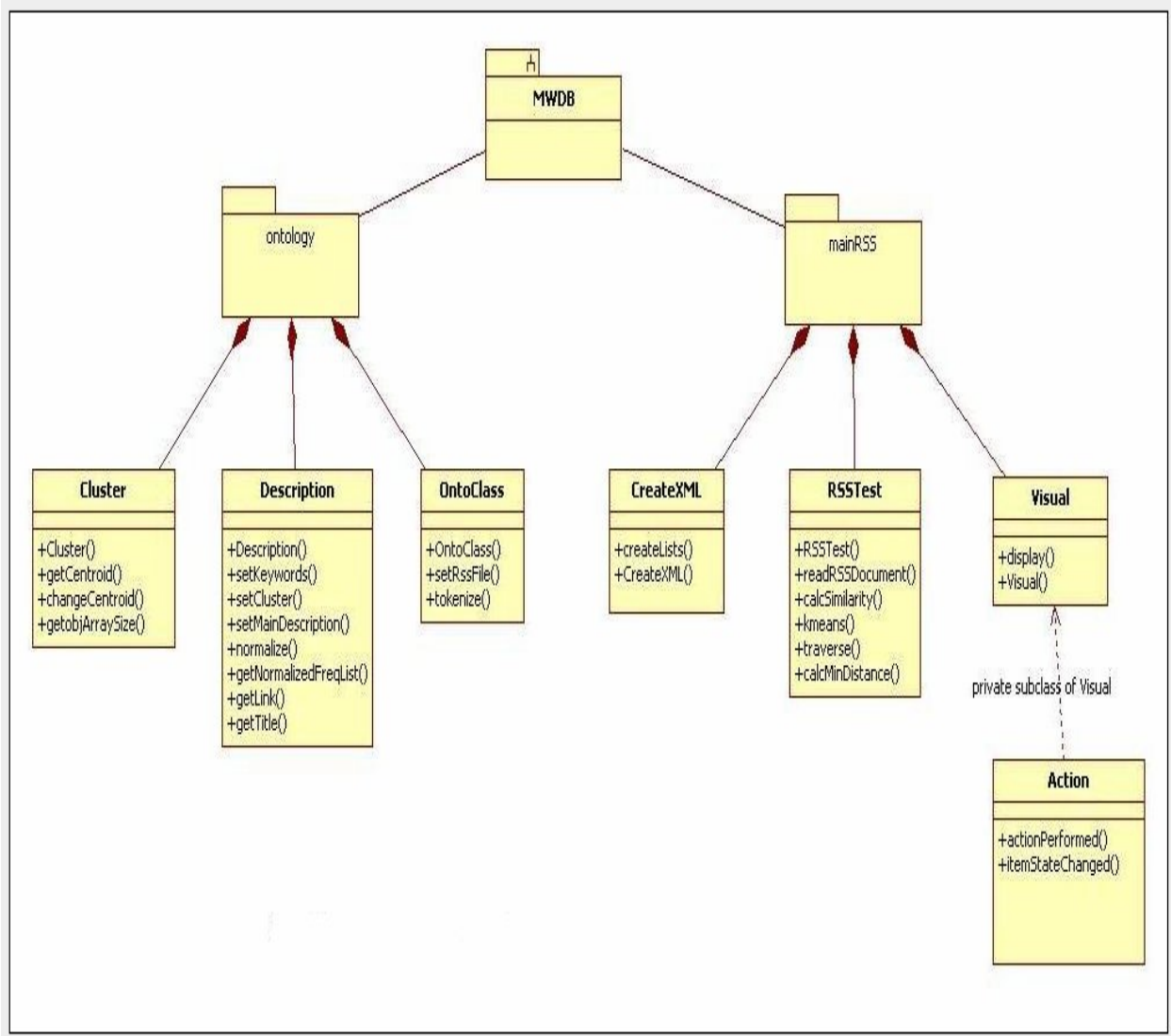


Figure 2: Class diagram

The project is implemented in java and consists of two packages; ontology and mainRSS. The classes include RSSTest, Description, Cluster, OntoClass, Visual and CreateXML.

Implementation classes detailed description:

1. Visual .java

The user screen implemented here which comes up initially is shown below:

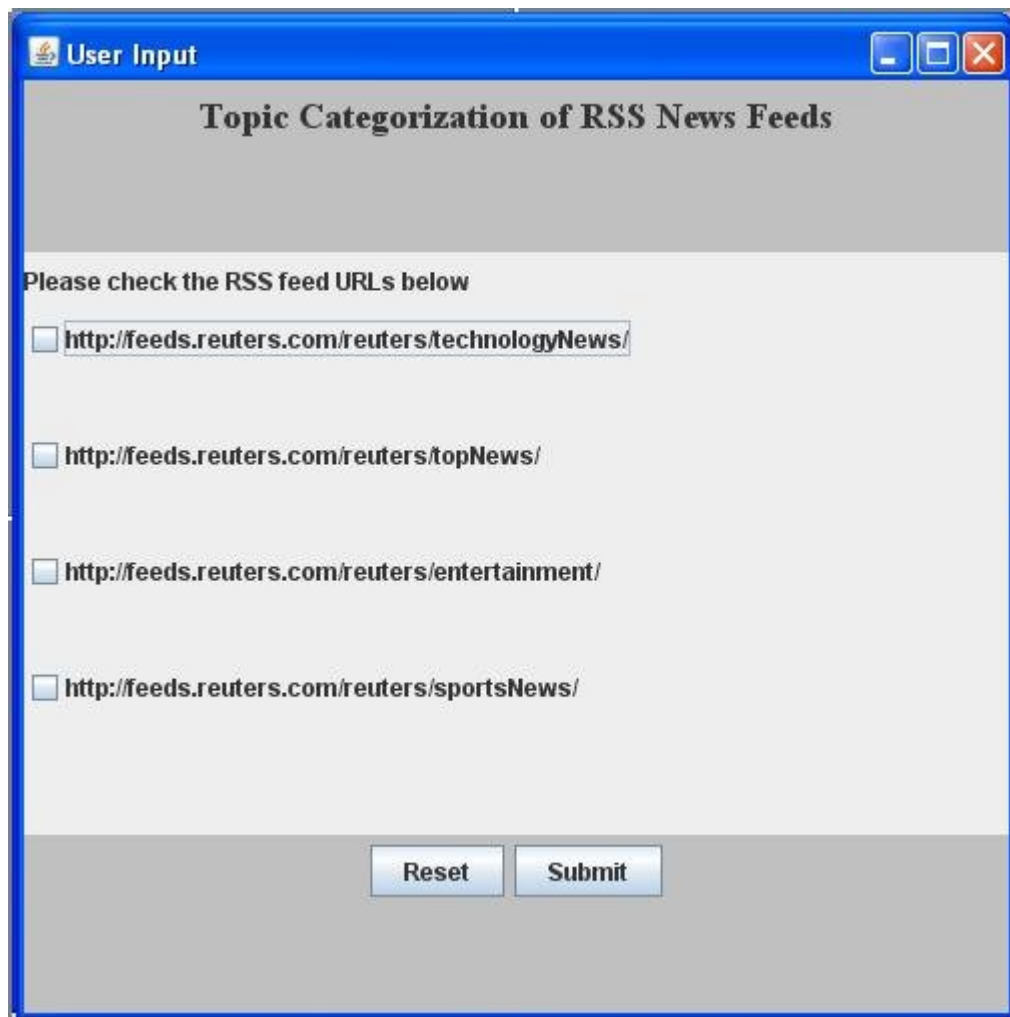


Figure 3: User Screen Java Applet

This is the class which provides the user interface for the application. It consists of a Java Swing Component which comes up as the display screen. The class has features for allowing the users to select the RSS feeds (URLs) from which the news would be fed as an input to the system. The URLs are simple strings. Four checkboxes are present on the

screen so that the user can select any of the four URLs by checking the checkboxes. As each of the checkboxes are marked, the strings are added to a String array which is then passed over to the class **RSSTest.java**.

2. Ontoclass.java

The processing of the input data starts in this class. Before the actual system is run, this class is used to create ontologies of domains related to *entertainment*, *technology*, *sports* and *politics*.

There is one major function in this class:

tokenize

For the purpose of creating an ontology, this function accepts the data from different news stories. This function uses a pre-built *stopword -file* to eliminate the stop words from the input file. Then the standard *Porter Stemmer* algorithm is applied to perform the stemming of words which is derived from the LUCENE package in the project.

The stemmed words are stored as keywords in a file which is used later as an ontology basis.

In case, when RSS files are given as input, the files undergo the same process. The exception is instead of storing stemmed words in a file they are stored in a HashMap object along with the frequency of their occurrence. The same HashMap is then used to initialize the Description object with keywords and frequency..

3. CreateXML.java

This class is used to create the RSS XML file which would be read by the RSS Reader.

The functions in this class are:

- a. createLists()
- b. CreateXML()

The createLists() function accepts the clusters formed as an input. It compares each cluster's centroid with each of the ontologies of "Entertainment", "Sports",

“Technology” and “Entertainment”. By doing so, it is possible to determine, which domain the cluster most likely belongs to and assign the appropriate label.

After the cluster’s label is determined, the function CreateXML(), creates a XML file with all the contents required for a RSS feed.(item, link, title, description). The XML file created is then uploaded onto the ASU Webspace.

(link:<http://www.public.asu.edu/~bpendhar/output.xml>)

We are using a RSS reader "Sharpreader", in which the user can type in the name of the link given above and receive the news from our RSS feed. The contents i.e the descriptions of each of the cluster are wrapped within these elements. The RSS file is created in such a way that all the descriptions in a single cluster are grouped together and displayed in the XML format. Each cluster is represented by a corresponding label assigned in createLists(). The RSS file created can be read in any RSS reader.

4. RSSTest.java:

- a. readRSSDocument : This function accepts a url from the Visual class and uses rss parser to parse the urls available in the rssutils.jar [reference]. For every new description in the urls, a separate object of Description class is created and stored in a map allDescriptions. Routines from Ontoclass are called to perform the stemming on these descriptions. Thus a preprocessed list of descriptions with the term list and the corresponding frequency list is prepared.
- b. getRandom : This function generates random numbers to pick the initial random centroids.
- c. kmeans : This function implements the main clustering algorithm as mentioned in Section [mention the section where the Algorithm is defined]. After the kmean algorithm has converged, CreateXML() function is called to generate the output xml file.

- d. `traverse` : For every iteration of `kmeans`, this function calculates the distance of every description to each of the clusters and assigns the description to the cluster for which it has the minimum distance.
- e. `calcSimilarity` : This function calculates the cosine similarity between two descriptions.

5. Description.java:

- a. `setKeywords` : This function adds the stemmed term list and the corresponding frequency list of a particular description to the `objTermList` and `objFreqList`, attributes maintained for every description.
- b. `normalize` : This normalizes the frequency list of the description.

6. Cluster.java:

`changeCentroid` : For every iteration of `kmeans`, this function calculates a new value for the centroid as discussed in section [describe the function in algorithm section].

User Study Evaluation

It is generally very hard to evaluate a system dealing extensively with text documents. This is because; it works closely with the concept of relevance, which is inherently a subjective matter. To do better evaluation, time consuming questionnaires need to be carried out. But then again, it would depend mainly on the user's subjective notion of relevance. The user study is thus considered critical to the success or failure of our system.

The users would be selecting the URLs from our User Interface (a Java applet). The URLs present news from different sources. The users must view the contents of these URLs on the browser and decide the category to which each of these descriptions might belong. The results that are obtained from our system are then compared with what the user had guessed. This gives a fair indicator of our system's performance.

We then plan to use some metrics to evaluate the system's quantitative performance.

- Recall =

the number of relevant descriptions grouped in the category

total number of relevant descriptions

- Precision =

number of relevant descriptions grouped into the category

total number of descriptions in the category

User Evaluation Results

User Name: Hardik Doshi (Group: 2)

Run 1-----Labeling

Sr. No.	Cluster Name	Precision	Recall
1	Sports	0.1071	0.75
2	Entertainment	0.3333	0.0625
3	Politics	0.3529	0.4615
4	Technology	0.2	0.25

Run 1-----Clustering

Sr. No.	Cluster Name	Precision	Recall
1	Cluster 1	0.3571	0.625
2	Cluster 2	0.3333	0.0625
3	Cluster 3	0.3529	0.4615
4	Cluster 4	0.8	0.25

Run 2-----Labeling

Sr. No.	Cluster Name	Precision	Recall
1	Sports	0.1818	0.1818
2	Entertainment	0/9	0/9
3	Politics	0/10	0/6
4	Technology	1	0.909

Run 2 -----Clustering

Sr. No.	Cluster Name	Precision	Recall
1	Cluster 1	0.5454	1
2	Cluster 2	1	0.8181

3	Cluster 3	0.8	0.7272
4	Cluster 4	1	0.909

User: Kavita (Non Technical User)***Run 1-----Labeling***

Sr. No.	Cluster Name	Precision	Recall
1	Sports	0.0476	0.5
2	Entertainment	0.1428	0.1
3	Politics	-	-
4	Technology	-	-

Run 1-----Clustering

Sr. No.	Cluster Name	Precision	Recall
1	Cluster 1	0.4285	0.9
2	Cluster 2	0.4285	0.3333
3	Cluster 3	-	-
4	Cluster 4	-	-

User: Krupa (Non Technical User)***Run 1-----Labeling***

Sr. No.	Cluster Name	Precision	Recall
1	Sports	0.5	0.7692
2	Entertainment	0.1	0.0909
3	Politics	-	-
4	Technology	0/9	0/11

Run 1-----Clustering

Sr. No.	Cluster Name	Precision	Recall
1	Cluster 1	0.5	0.7692
2	Cluster 2	0.2	0.2
3	Cluster 3	-	-
4	Cluster 4	1	0.8181

User Name: Deepa (Computer Science Student)***Run 1-----Labeling***

Sr. No.	Cluster Name	Precision	Recall
1	Sports	0.0645	0.5

2	Entertainment	-	-
3	Politics	-	-
4	Technology	(0/8)	0/0

Run 1-----Clustering

Sr. No.	Cluster Name	Precision	Recall
1	Cluster 1	0.3870	0.9230
2	Cluster 2	-	-
3	Cluster 3	-	-
4	Cluster 4	0.375	0.2

User Name: Naman (Computer Science Student)***Run 1-----Labeling***

Sr. No.	Cluster Name	Precision	Recall
1	Sports	0.5882	0.7692
2	Entertainment	0/1	0/0
3	Politics	0.2592	0.4074
4	Technology	-	-

Run 1-----Clustering

Sr. No.	Cluster Name	Precision	Recall
1	Cluster 1	0.5882	0.7692
2	Cluster 2	0/1	0/0
3	Cluster 3	0.2592	0.4074
4	Cluster 4	-	-

Limitations

1. We have created our ontologies based on limited training data. Our training data consisted of entire news stories from the 4 domains: politics, technology, entertainment and sports. We created the training data from 320 news stories for every domain. We removed stop words, applied stemming and computed the term frequencies of the remaining words. We empirically observed that words with frequencies 4 or more were important in representing the domain. Thus we retained such words for every domain as the content words representing that domain. Our final ontology consists of 3587 content words of sports domain, 623

- of technology, 216 of entertainment and 630 of politics. Therefore, our ontologies may not be all inclusive for these categories and may have some false hits and misses. This eventually would deteriorate the quality of clustering and the final results.
2. Our system does not take care of redundant news from various news sources. A similarity detection module may be added to remove redundant information.
 3. Some descriptions belong to categories which we have not considered. So we deal with such data by labeling it as “Other”.

Challenges

1. One of the major challenges is building the concept lexicons with domain specific terms. We use some training data to build the concept lexicons, and thus there is a possibility that we miss some terms.
2. Another challenge is to build a scalable system that would work reasonably well even when the input data considered is huge.
3. The converging conditions in text clustering can be tricky because of the vastness of data.
4. Finding good RSS feeds is cumbersome. Some RSS feeds has description like “Read full story for detail”.

System Requirements:

Operating System: All 32- bit MS Windows (95/98/NT/2000/XP).

Browser: Netscape 7.0 and above/ Mozilla Firefox.

Java Runtime Environment Version 5.0

Internet Connection : Cable or DSL.

Related work

Gmeans[6] is a algorithm for text clustering of a given dataset and it uses four different similarity measures, six various initialization methods and a powerful local search strategy called first variation. A frequent item based approach[9] provides a natural way

of reducing the dimensionality of the vector space model. These frequent item sets are discovered using association rule mining. This paper describes two algorithms for frequent term-based text clustering, called FTC (flat clustering) and HFTC (hierarchical clustering). A lot of different text clustering algorithms are proposed in the literature, including Scatter/Gather, SuffixTree Clustering and bisecting k-means. [10] discusses the use of association rule mining in building text categorization system and proposes a new fast algorithm for building a text classifier. [11] discusses the use of ontology and preprocessing in form of concept selection and aggregation. Scatter-Gather algorithm [12] is a document clustering method that uses the table of contents metaphor and can be used as an effective document retrieval tool. [13] details the text clustering and evaluation methods. [14] discusses training documents with known categories and then using probabilistic clustering algorithm to find the category of the test document.

Conclusions

In this report, we have discussed the method to build ontologies and perform text clustering based on the given RSS feeds. Further, we are also labeling each of the formed clusters with predefined category names. The experimental results for the “Clustering” part were better than that of the “Labelling” part. Thus, we are able to infer that the objective of clustering descriptions is well achieved. Moreover, the reasoning for lower quality results of “Labelling” part can be attributed to the ontology which we have built. We need to have an ever evolving ontology which would help us in correctly labeling the clusters.

Future Work

The project could be extended by including more domains. Moreover, we must use extensive data from large number of news sources. This would ensure that we have a scalable system. The system can also be made web based such that online access will be facilitated and system can be extended to huge dataset and concept based lexicons. Redundant information from different news sources can be removed by using similarity detection techniques.

Bibliography

[1] Description of RSSUtilities

http://java.sun.com/developer/technicalArticles/javaserverpages/rss_utilities/

[2] “Reading the News with Sun's RSS Utilities”- Chris Hardin

<http://today.java.net/lpt/a/275>

[3] Domain specific Ontology

[http://en.wikipedia.org/wiki/Ontology_\(computer_science\)](http://en.wikipedia.org/wiki/Ontology_(computer_science))

[4] “Conflation Algorithm” <http://www.dcs.gla.ac.uk/Keith/Chapter.2/Ch.2.html>

[5] “Apache Lucene” <http://lucene.apache.org/java/docs/>

[6] “Gmeans: clustering in ping-pong style”-

<http://www.cs.utexas.edu/users/yguan/datamining/gmeans.html>

[7] “RSS Tutorial” <http://www.mnot.net/rss/tutorial/>

[8] http://www.java.com/en/download/windows_manual.jsp

[9] “Frequent Term-Based Text Clustering” - Florian Beil, Martin Ester, Xiaowei Xu

[10] “Classifying Text Documents by Associating Terms with Text Categories”

Osmar Zaiane Maria-Luiza Antonie

[11] “Ontology-based Text Clustering”- A. Hotho and S. Staab, A. Maedche

[12] “Scatter/Gather: a cluster based approach to browsing large document collections”

[13] “Introduction to Information Retrieval and Text Clustering” - Magnus Rosell

[14] Cluster-Based Text Categorization: A Comparison of Category Search Strategies

[15] <http://www.sharpreader.net/>

Further Reading

This section enumerates the references that we have used while working on our project, though eventually we dropped few ideas.

- “How to parse RSS feeds with PHP”

http://www.softarea51.com/tutorials/parse_rss_with_php.html

- “Latent Semantic Indexing”

http://www.knowledgesearch.org/lsi/lisa_definition.htm

- “Text Clustering for Topic Detection”, Y. Seo and K. Sycara, tech. report CMU-RI-TR-04-03, Robotics Institute, Carnegie Mellon University, January, 2004.
- <http://eivind.imm.dtu.dk/thor/projects/multimedia/textmining/node5.html>
- “What is Text Clustering?” <http://www2.parc.com/istl/projects/ia/sg-clustering.html>
- <http://flock.sourceforge.net/>
- “Strength and similarity of affix removal stemming algorithms”- William B. Frakes and Christopher J. Fox.
- “RSS Creator” <http://www.webreference.com/cgi-bin/perl/makerss.pl>

Appendix

Specific roles of the group members:

1. Conceptualization of the algorithm: Bhushan, Pooja, Prajakta, Sanket, Srushti
2. Creating module for processing documents to derive content words and building the ontology: Bhushan , Pooja, and Srushti
3. Parsing of RSS feeds through RSSutils package: Prajakta and Sanket
4. Design and implementation of framework classes: Sanket and Srushti
5. Implementation of Algorithms: Bhushan, Pooja and Prajakta
6. Evaluation of the System: Bhushan, Pooja, Prajakta, Sanket, Srushti
7. Documentation: Bhushan, Pooja, Prajakta, Sanket, Srushti