NSU Florida
NOVA SOUTHEASTERN UNIVERSITY

Nova Southeastern University
NSUWorks

CEC Theses and Dissertations

College of Engineering and Computing

2015

News Feeds Clustering Research Study

Haytham Abuel-Futuh
Nova Southeastern University, haytham@afutuh.com

This document is a product of extensive research conducted at the Nova Southeastern University College of Engineering and Computing. For more information on research and degree programs at the NSU College of Engineering and Computing, please click here.

Follow this and additional works at: https://nsuworks.nova.edu/gscis_etd


Part of the Computer Sciences Commons

Share Feedback About This Item

NSUWorks Citation

Haytham Abuel-Futuh. 2015. *News Feeds Clustering Research Study.* Master's thesis. Nova Southeastern University. Retrieved from NSUWorks, Graduate School of Computer and Information Sciences. (52)
https://nsuworks.nova.edu/gscis_etd/52.

This Thesis is brought to you by the College of Engineering and Computing at NSUWorks. It has been accepted for inclusion in CEC Theses and Dissertations by an authorized administrator of NSUWorks. For more information, please contact nsuworks@nova.edu.

News Feeds Clustering Research Study

By

Haytham AbuelFutuh

Supervised by

Dr. Peixiang Liu

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Computer Information Systems

Graduate School of Computer and Information Sciences

Nova Southeastern University

2015

Thesis Of

Haytham Abuel-Futuh

Submitted in partial Fulfillment of the Requirements for the Degree of

M. S. in Computer Science

News Feeds Clustering Research Study

Nova Southeastern University
Graduate School of Computer and Information Sciences

April 2015

Signed Original On File at NSU

Thesis Approved By: _____
Peixiang Liu, Ph.D.

Thesis Abstract

With over 0.25 billion web pages hosted in the World Wide Web, it is virtually impossible to navigate through the Internet. Many applications try to help users achieve this task. For example, search engines build indexes to make the entire World Wide Web searchable, and news curators allow users to browse topics of interest on different structured sites. One problem that arises for these applications and others with similar goals is identifying documents with similar contents. This helps the applications show users documents with unique contents as well as group various similar documents under similar topics. There has been a lot of effort into algorithms that can achieve that task. Prior research include Yang, Pierce & Carbonell (1998) research where they looked at the problem of identifying news events exploiting chronology order, Nallapati, et al (2004) research who built a dependency model for news events and Shah & Elbahesh (2004) research where they used Jaccard coefficient to generate a flat list of topics. This research will identify training and testing datasets, and it will train and evaluate (Pera & Ng) algorithm. The chosen algorithm is a hierarchical clustering algorithm that incorporates many of the ideas researched earlier. In evaluation phase, error will be measured in the ratio of miss-categorized documents to the total number of documents. The research will show error can be as low as 0.03 with a model built on a single node processing 1000 random distinct documents. In evaluation of the algorithm, the experiments will show that (Pera & Ng)'s fuzzy equivalence algorithm does produce acceptable results when compared to Google News as a reference. The algorithm, however, requires a huge amount of memory to hold the trained model. This renders it not suitable to run on portable devices.

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

Problem statement

With over 0.25 billion web pages hosted in the World Wide Web (World Wide Web Size, n.d.), it is virtually impossible to navigate through the Internet. Internet users resort to many applications to help with this task; Social Networks, Search Engines, news curators and customized home pages are among them. These applications' main job is to make it easier for users to find the most credible web pages relevant to what users are interested in. Social Networks rely on user's own contacts (aka. User Graph) to show them what is more likely to be of interest to them. Search Engines achieve this task by building an index of the entire browse-able web, and then they allow users to search for terms in those pages, they then display results ordered by their relevancy to search terms used.

Following we show how these applications try to deal with the navigation task from the perspective of document similarity.

Retrieving search results to users is a good example to demonstrate a fundamental issue that users face with navigating content on the Internet. Even with powerful search engines, they still fail to filter out aggregator sites accurately; those are sites that aggregates data from other known sites and present them as their own, sometimes-violating copyrights. When searching for common terms, it is common to see multiple links pointing to those sites at the top of the results page. The content is relevant to the user's search query but the engine fails to capture the fact that those links all tell the same story, tutorial or solution the user is looking for. Our hypothesis is that had search

engines took document similarity as a factor in ranking and aggregating results, more results would have shown up and only the most useful and important ones will be at the top. Other factors they take to rank web pages are outside the scope of this work.

On the other hand, news curators (e.g. Google Reader, Feedly… etc.) and customized home pages (e.g. Yahoo MyPage, MSN Home Page… etc.) work differently; they allow users to specify their interests (e.g. World News, Sports, Technical News… etc.), and then they try to show them recent web pages that are classified under these categories. They sometimes allow users to specify web sources they are interested to follow. A formal descriptive language Rich Site Summary (RSS) based on eXtensible Markup Language (XML) helps standardize the format these applications consume. Programs known as RSS Readers or RSS Aggregators help users both manage the subscriptions to feeds and download articles in the subscribed feeds. As users subscribe to more and more RSS feeds, managing them becomes a hectic task. Most RSS Readers offer users a manual way to group feeds into categories (e.g. News, Technology… etc.). This helps users read RSS posts about similar topics under same category. However, this grouping scheme fails to capture a very essential aspect of the feeds inside each group. That is they are more likely to talk about the same popular topic.

There are more specific use cases for understanding the similarity between documents on the web. For example, Amazon groups users' reviews by similarity. Then they rank them by their usefulness (defined by how much they affected users' decisions to buy the product). This allows users to browse thousands of reviews by only looking at a few distinct opinions.

Another example is Research sites group papers that discuss the same topics together. Clustering papers this way allows researchers of certain topic to view related papers.

As shown above, all of the applications meant to enable users to navigate the World Wide Web can benefit from using a document similarity algorithm.

Document clustering is the problem of grouping documents based on their similarity. Similar documents appear in the same cluster while different documents appear in different clusters. Choosing the right function to determine similarity between documents is not obvious. Many researchers looked at different methodologies for determining whether two pieces of content are similar (Text content, image contents, and other media contents are among content types researched. This research focuses on Text content). Chapter 2 below reviews previous efforts done in this area.

Significance of the research

Web documents (or pages) contain a verity of content types, in order for an effective document clustering technique to work; we need to extract content in an abstract form. RSS is helpful in that it provides a formal way to describe web site content. Around 30% of Internet users use RSS Feeds as shown by Barnett (2005). Most of those unknowingly do so as a study shows that only 4% of those asked whether they use RSS feeds answered ‘Yes’ (Barnett, 2005). A google survey indicated similar numbers (Survey, n.d.). We provide an abstraction for document contents and run on different types of contents (RSS, Wikimedia XML, text files… etc.).

This research explores different news clustering techniques and compares them with respect to their efficiency (accuracy and performance).

Terms and definitions

Cluster: A group or a subset of documents that are similar enough but are sufficiently dissimilar from documents in other clusters.

Clustering: Is the action of dividing a group of items into smaller groups. Items in each smaller group are similar but are dissimilar from items in other groups.

Unsupervised learning: No supervision means that there is no human expert teaching the algorithm. Clustering is one of the most common examples of unsupervised learning. In clustering, the key input to a clustering algorithm is the distance function. The distance measure suggests how close/far apart two pieces of data are, this measure is then used to put similar pieces of data into clusters.

Chapter 2 Literature review

There has been a lot of research in the field of document clustering for use in different areas of text mining and information retrieval. The basic form for using document clustering is improving the precision in information retrieval systems. Ordering search results, helping users navigate through a large base of documents are recent areas that require efficient document clustering techniques.

Many Internet services people use today have implemented clustering in some form or another (e.g. Google News (Google Inc., n.d.), Topix (Topix LLC, n.d.)…etc.). In general, clustering news articles involves identifying what articles talk about similar topics (Yang, Pierce, & Carbonell, 1998). There is a wide range of techniques to achieve this that are mentioned in the next section. There are generally two uses of clustering news based on RSS feeds:

- **Highlight the interesting bits of news within a pool of feeds**: This approach depends on the assumption that the number of appearances of a topic is directly proportional to its importance.
- **Cluster entries around topics**: This approach tries to offer the user a way to read a list of RSS feeds organized by topics.

(Nazario, 2005).

Existing techniques

We can categorize document-clustering techniques into a multiple categories (Wanner, 2004):

- Hierarchical techniques

Hierarchical algorithms learn about clusters gradually. Hierarchical algorithms produce a tree of clusters (see Figure 1 Sample hierarchical clusters).

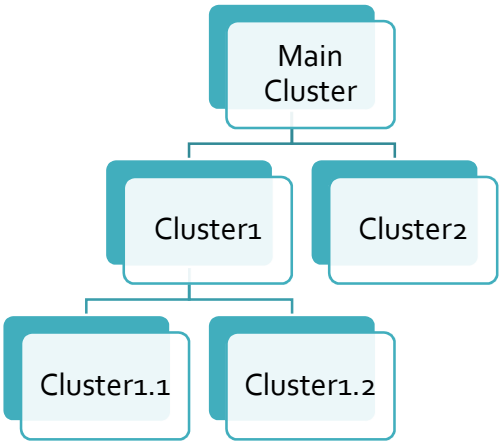

Figure 1 Sample hierarchical clusters

- Agglomerative Algorithms

Described as a bottom up approach for clustering. Starting with small clusters (one per document), the algorithms keeps merging them to produce bigger ones until the termination criterion is met (e.g. desired number of clusters).

- Divisive Algorithms

Divisive algorithms use top down approach; starting with one big cluster that contains all documents, they perform split operations until the termination criterion is met (e.g. desired number of clusters).

- Partitioning techniques

On the other hand, partitioning techniques learn clusters directly. To achieve that, they do one of two things:

1) Some algorithms define data points between clusters then iteratively relocate them to separate the clusters more accurately.

2) Other set of algorithms learn clusters by identifying areas of highly populated data points.

Examples for partitioning techniques:

- K-Means Algorithms
- Probabilistic Algorithms
- Density-Based Algorithms

- Grid-based algorithms

Grid-based algorithms work on data through summarizing data over some attribute space. They perform segmentation then aggregate appropriate segments to form final clusters.

- Co-Occurrence of categorical data-based algorithms

Categorical data has some unique characteristics that render previous algorithms inefficient.

- Constraint based algorithms
- Evolutionary algorithms
- High dimensional data algorithms
  - Subspace Clustering
  - Projection algorithms
  - Co-Clustering

Yang, Pierce, & Carbonell **(1998) research**

They researched the problem of identifying events in a continuous stream of news stories or in a retrospective manner. Specifically, they investigated events from perspective of the following:

1) Semantic and temporal properties of events
2) Document clustering based on content and temporal adjacency (rather than just content)
3) Event detection based on similarity versus novelty;
4) Evaluation methods for retrospective and on-line detection.

After obtaining 15 thousand news stories from various sources (CNN news and Reuters articles from January to February 1995), they applied their hierarchal content-based clustering algorithm, clustered the news articles into various topics and presented each cluster using the statistically significant terms in that cluster. The resulting summary table is below.

Table 1 Summary of results for the hierarchal content-based clustering algorithm proposed by Yang, Pierce, & Carbonell (1998)

Number of Documents included	Top-ranking words (stemmed)
330	Republ Clinton congress hous amend
217	Simpson 0 prosecut trial jury
98	israel palestin gaza peat arafat
97	japan kobe earthquak quak toky
93	russian chech chechny grozn Yeltsin
56	somal u mogadishu iraq marin
55	flood rain californ malibu rive
48	serb bosnian bosnia croat u
35	game leagu play basebal season
33	crash airlin flight airport passeng

28	clinic sav abort massachuset norfolk
27	shuttl spat astronaut mir discov
26	patient drug virus holtz infect
24	chin beij deng trad copyright
…	

Yang, Pierce & Carbonell evaluated multiple clustering algorithms against the document corpse. The summary of these algorithms is as follows:

1) Group Average Clustering (GAC)
    a. GAC is a bottom-up algorithm. Starting with an ordered list of documents (ordered chronologically in their case). The algorithm divides those documents into a lot of overlapping buckets. It then, iteratively, combines the buckets into higher order ones. It keeps repeating the process until it reaches the desired number of high-level buckets.
    b. GAC runtime complexity is $O(mn)$ where n is the number of documents in the input corpus, m is the bucket size, and $m \leq n$.
    c. GAC achieved best results for retrospective document clustering. It has comparable results for on-line document scanning.
2) Incremental Clustering Algorithm
    a. The incremental clustering algorithm has a simple but high cost approach. It sequentially processes each document and compares it to existing clusters. If it is sufficiently similar to any of the existing clusters, the

document joins the cluster. Otherwise, it considered the document a seed for a new cluster.

To evaluate the performance of each of the algorithms, Yang, Pierce & Carbonell defined five evaluation measures; miss, false alarm (f), recall (r), precision (p) and the FI measure.

Table 2 Table comparing GAC and INCR results for retrospective news stories clustering.

	GAC	INCR
Recall (%)	75	62
Precision (%)	90	82
Miss (%)	25	38
False Alarm (%)	0.02	0.04
Micro-Avg F1	0.82	0.71
Macro-Avg F1	0.84	0.79

The results show that GAC outperformed in news stories clustering. Yang, Pierce & Carbonell concludes that the main reason for GAC's performance is its multi-level clustering approach. That comes at the cost of generating too many partitions (12K as opposed to 5K generated by INCR).

Nallapati, et al (2004) research

Their research capture the structure of on-line news events that make up different topics and the dependencies among them through different event models. The system is

efficient enough when fed the events to look for. However, the performance degrades significantly if it has to discover events dynamically.

The main strength and the focus of the research is in trying to overcome the problem resulting from organizing news pieces into a flat list of topics. They believe that approach is too restrictive and inefficient for users to browse the news. Their approach relies on threading various events together to understand news dependencies and put them in a structure that is more natural to browse.

To formally identify an event, they set certain constraints on what an event is and is not:

1) $\forall i\ \varepsilon_i \in 2^S$

   States that each event ε is an element in the power set S of news stories.

2) $\forall i, j\ s.t.\ i \neq j, \varepsilon_i \cap \varepsilon_j = \emptyset$

   States that each story can belong to at most one event.

3) $\forall s_i \exists \varepsilon_k \epsilon \varepsilon\ s.t.\ s_i \epsilon \varepsilon_k$

   States that each story S_i belongs to one of the events in the events set.

4) $f(s_i) = \varepsilon_k iff\ s_i \epsilon \varepsilon_k$

   From 1-3, they concludes this mapping function from story S_i to event ε_k.

They further define a set of directed edges $E = \{(\varepsilon_i, \varepsilon_j)\}$ that describe dependency or temporal ordering of events.

To test their approach, Nallapati, et al picked 53 news pieces from TDT corpus, hired an annotator to create truth data, the annotation included identifying events and

their dependencies. Then they have split the data randomly into 26 topics for training set and 27 for testing set. Table 3 shows that the training and test sets have similar statistics:

Table 3 Statistics of annotated data

Feature	Training set	Test set
Num. topics	26	27
Avg. Num. Stories/Topics	28.69	26.74
Avg. Doc. Len	64.60	64.04
Avg. Num. Stories/Event	5.65	6.22
Avg. Num. Events/Topic	5.07	4.29
Avg. Num. Dependencies/Topic	3.07	2.92
Avg. Num. Dependencies/Event	0.61	0.68
Avg. Num. Days/Topic	30.65	34.48

A summary of the models trained/tested follows:

1) Nearest Parent Model

   The assumption in this model is that each event can have at most one parent. Moreover, events must occur within a short time range to be considered for dependency relationship.

2) Best Similarity Model

   The assumption in this model also assumes that each event can have at most one parent. In this model, an event is a parent of another only if it happened earlier and has scored the highest similarity score.

3) Maximum Spanning Tree Model (MST)

This model allows events to have more than one parent. It assumes a fully connected undirected graph between events. Calculates the minimum spanning tree on all edges based on average similarity of events and their temporal ordering (chronological ordering).

4) Simple threshold Model

This model defines a threshold for similarity. If two events score higher than the threshold, it creates a dependency edge between them.

Results and conclusion

After training the various model algorithms, they ran them on the test data. Table 4 shows summary of the test results.

Table 4 Testing results for various models

Model	DP	DR	DF
Nearest Parent	0.61	0.60	0.60
Best Similarity	0.71	0.74	0.72
MST	0.70	0.68	0.69
Simple Thresh	0.57	0.75	0.64
Baseline (Complete-link)	0.50	0.94	0.63

Legend:

- Dependency Precision (DP): It is the probability that there is a dependency between the events of two randomly selected stories S_i and S_j in the true model M given that they have a dependency in the system model M'.
- Dependency Recall (DR): It is the probability that there is a dependency between the events of two randomly selected stories S_i and S_j in the true model M' given that they have a dependency in the system model M.
- Dependency F-measure (DF): The average between DP and DR.

In conclusion, Nallapati, et al (2004) developed a time decay based clustering approach that takes advantage of temporal localization of news stories on the same event and showed that it performs significantly better than the baseline approach based on cosine similarity.

Shah & ElBahesh (2004)

In their research, they used Jaccard coefficient and stemming algorithms to perform text mining over news articles and find similarities.

Unlike Nallapati, et al, Shah & Elbahesh decided to group news articles into a flat list of topics by similarity. They have obtained records the UAB Reporter's archive. The algorithms used do not build a model hence do not a training set. They have tested their proposed algorithm (as well as others for comparison) on a 1500 set of stories obtained from the same archive in random.

Before running through the similarity algorithm, they have done a few steps on the data set:

1) Cleansing: removing stop words
2) Word Stemming: Converting each article into its stemmed equivalent.
3) Similarity measure: They have computed the similarity factor between articles as:

$$Sim(A_i, A_j) = \frac{|Words(A_i) \cap Words(A_j)|}{|Words(A_i) + Words(A_j)|} \; \forall i, j \in NewsSet$$

This states that the more common words two news articles have, the more similar they are.

4) Clustering: They have used three algorithms summarized below
    a. K-Nearest Neighbor:

    In this algorithm, after defining k, it finds the k closest articles to every article in the set.

    After it terminates, articles with links among each of them form a cluster.

    b. Single Link Clustering Algorithm:

    This algorithm defines a similarity threshold, if two articles' similarity measure is more than or equals to the threshold, they belong to the same cluster, otherwise a new cluster forms.

    c. Hybrid Algorithm:

    This algorithm is very similar to K-Nearest Neighbor with one major difference. It only draws an edge between two articles if the similarity measure is more than or equals to a user-defined distance.

Results and conclusion

Table 5 shows results of running those algorithms against the same set of news articles. The Hybrid algorithm outperforms the rest.

Table 5 Summarized results of clustering 1500 articles (n=10, k=1)

Algorithm	ε	No. of non-singleton clusters	Avg. No. of elements in non-singletons
Single-link	0.7	31	42.16
Single-link	0.5	132	2.5
Nearest Nbr.	n/a	277	5.42
Hybrid	0.7	277	4.72
Hybrid	0.5	139	2.37

Li, et al (2007)

Li, et al specifically considers RSS news-clustering problem have proposed using K-nearest neighbor algorithm to find the k nearest stories to any given story S. There has been many techniques built on top of K-mean algorithm; however, Li, et al (2007) seemed to provide the best performance of them.

They have architecture their system into three main modules: a crawler module, a clustering module and a topic extraction module.

Their system addresses one of the main concerns on the practicality of any clustering algorithm: the cost of doing online clustering. To deal with this, they have defined a

flexible half-bounded sliding time window, which dynamically extends its ending point during a period and periodically re-cluster all stories.

Results and conclusion

Table 6 Summarizes results for RCS algorithm

Algorithm	Clustering Purity
Content Based K-Means	0.78
RCS	0.71
Link-based	0.31

They ran their system on 200,000 news stories from 3000 RSS feeds. Table 6 summarizes their results.

Content Based K-Means does score slightly better than the proposed RCS algorithm. However, its performance is exponentially worse. Li, et al claim that RCS performance is constant without giving much details on how they have achieved that.

Chapter 3 Methodology

We have chosen (Pera & Ng) algorithm for evaluation. The algorithm incorporates many of the ideas discussed in previous techniques and improves on them using Fuzzy equivalence making it arguably closer to human interpretation to equivalency than a one-to-one word matching.

In this chapter, we summarize the steps involved in the algorithm, we detail our choices for external algorithms dependencies and then we describe the training and runtime algorithms in details. We also detail the hardware specifications used in evaluating the algorithm.

In the next chapter, we explain inputs and outputs used in evaluating the algorithm and its accuracy.

Algorithms

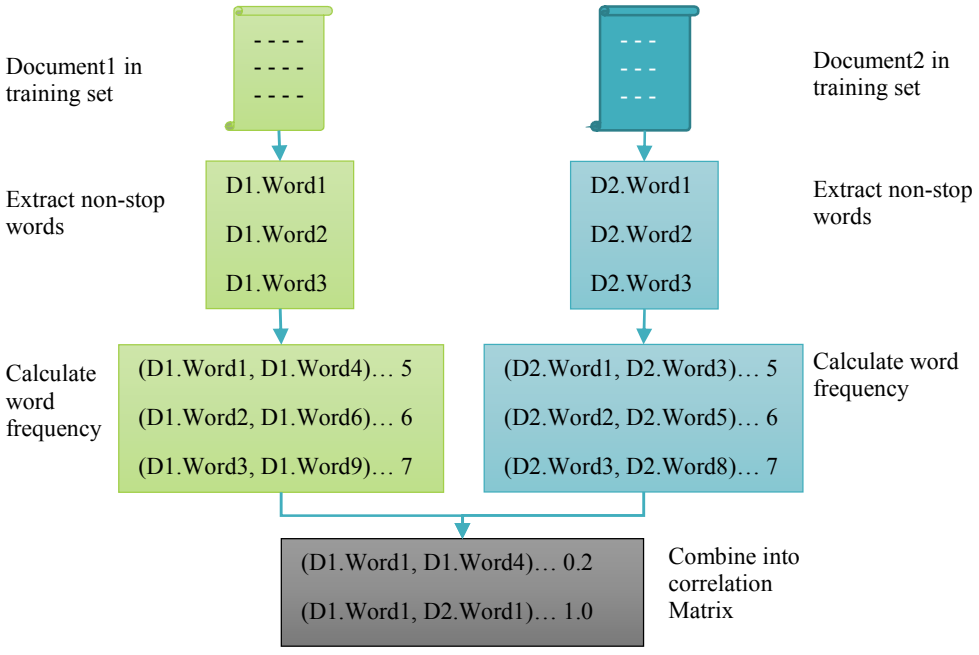

Figure 2 Training Algorithm.

Following (Pera & Ng) algorithm, we have implemented the original algorithm as well as a distributed version on Hadoop's Map Reduce implementation.

Algorithm summary

Training algorithm

The goal of the algorithm is to build a training model. The model describes the likelihood of two words appearing in the same statement (Figure 2 Training Algorithm.

1) Run Word Breaker algorithm to extract words from each document

There are multiple algorithms for implementing statement segmentation. We categorize them into two categories:

i. Ad-Hoc approaches

- Build heuristics that minimizes false positives/negatives when segmenting statements. E.g. a sample set of rules:
  a. If the parser encounters a period, it ends a sentence.
  b. If the preceding token is on a pre-defined list of abbreviations, then it does not end a sentence (e.g. U.S.A… etc.).
  c. If the next token is capitalized, then it ends a sentence.

(Gillick, 2009)

ii. Machine Learning approaches

- Using supervised learning, we can feed the algorithm pre-annotated segmented statements to build a model. We can then use the model to detect sentence boundaries.
- Maximum Entropy Model

- Neural Networks

2) Discard Stop Words

3) Calculate Keyword Co-Occurrence Matrix

4) Calculate Normalized Keyword Correlation Matrix

Runtime Algorithm

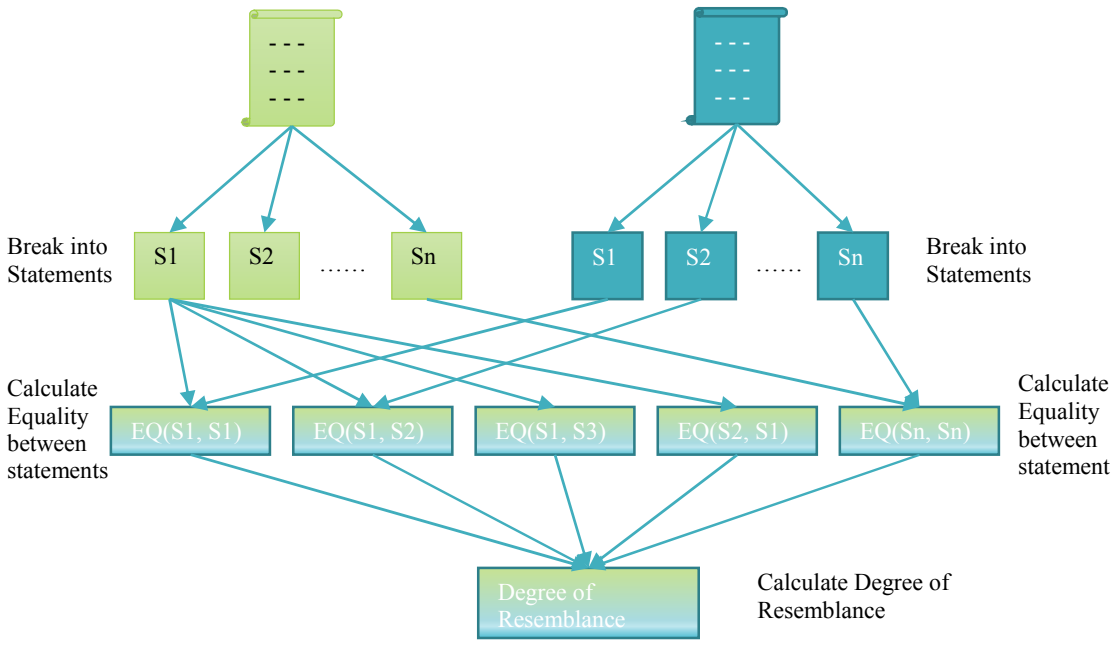

Figure 3 Runtime Algorithm

The goal of the algorithm is to assign a score that describes the similarity between any two documents. We then use that score to cluster documents into groups of similar topics. The summary of the steps for the runtime algorithm are as follows:

1) Calculate Odds Ratio between two documents

This involves calculating the number of equivalent statements between every two documents

2) Create a new cluster if we cannot find a document that matches the new document in any existing cluster.

Algorithm details

Training algorithm

1) Break statements into words.
2) Run word-stemming algorithm to remove stop words (e.g. and, or, I, he… etc.) from the article. We got this (https://skydrive.live.com/?cid=057cfc26026be037&id=&action=Share#cid=057CFC26026BE037&id=57CFC26026BE037%21105) stop words file and we are using The Porter Stemming Algorithm (2006).
3) We calculate Keyword Co-occurrence for each document in the training set as follows:

```
FOR ALL w1 IN doc
   WordCount[w1]++
   FOR ALL w2 IN doc
      Frequency[w1_w2] += 1 / (d(w1, w2) + 1)
FOR ALL item IN Frequency
   item.Value = item.Value / (WordCount[item.W1] *
WordCount[item.W2])
```

4) We calculate keyword correlation factor as follows:

```
FOR ALL frequencyModel IN frequencyModels (calculated in step 2)
   FOR ALL item IN frequencyModel
```

correlationFactor[item.Key].Value += item.Value

correlationFactor[item.Key].Count++

Now we can get correlationFactor["hi_bye"] and will return Value/Count calculated above...

5) Force the correlation value for pairs of the same word (e.g. correlationFactor[“hello_hello”] to be equal to 1)

6) Calculate a normalized correlation factor matrix as follows:

$$C_{i,j} = \frac{P_{i,j}}{N_i \times N_j}$$

Where $P_{i,j}$ is the calculated correlationFactor value in (4) above. N_i and N_j are the occurrences count of word i & j respectively.

R*untime algorithm*

The algorithm defines a few relations to calculate similarity between documents.

1) Degree of similarity between a word and a sentence:

Defined as the cumulative value of correlation factors between word i and each word j in statement S.

$$\mu_{i,S} = 1 - \prod_{j \in S} (1 - C_{i,j})$$

2) Degree of similarity between two statements

Defined as the normalized value of summation of similarities between each word i in statement $S1$ and statement $S2$.

$$Sim(S_1, S_2) = \sum_{i \in S_1} \frac{\mu_{i,S_2}}{|S_1|}$$

3) Equality of two statements

Defined as either 0 or 1. This value is driven by pThresh and vThresh value. The process to drive these values is discussed in next section.

$$EQ(S_1, S_2) = \begin{cases} 1 \; if \; MIN(Sim(S_1, S_2), Sim(S_2, S_1)) \geq pThresh \\ \quad \text{^} |Sim(S_1, S_2) - Sim(S_2, S_1)| \leq vThresh \\ 0 \qquad\qquad\qquad\qquad\qquad otherwise \end{cases}$$

4) Degree of resemblance between documents

Defined as the number of statements in document D_1 that are equivalent to (based on 3) above) statements in document D_2 over the total number of statements in document D_1.

$$RS(D_1, D_2) = \frac{\sum_{i \in D_1} \left(1 - \prod_{j \in D_2} \left(1 - EQ(S_i, S_j)\right)\right)}{|D_1|}$$

5) Odds Ratio

$RS(D_1, D_2) \neq RS(D_2, D_1)$ This does not make it easy to determine which sets of documents are equivalent. In order to combine these two values to produce a single value that indicates the resemblance, the author used Dempster-Shafer combination rule (Sentz & Ferson, 2002).

The odds ratio is defined as the ratio for the probability that an event occurs to the probability that it does not.

$$ODDS(D_1, D_2) = \frac{\left(RS(D_1, D_2) \times RS(D_2, D_1)\right)}{1 - \left(RS(D_1, D_2) \times RS(D_2, D_1)\right)}$$

To determine if two documents are equivalent, a threshold value $dThresh$ has to be set such as:

$$EQ(D_1, D_2) = \begin{cases} TRUE \; if \; ODDS(D_1, D_2) \leq dThresh \\ FALSE \qquad\qquad\qquad\qquad Othewise \end{cases}$$

Requirements and specifications

The methodology described here has been evaluated on a single node with the following relevant hardware specifications:

1) CPU: Intel Core i7 2.49GHz
2) Memory: 8.00 GB

We ran one of the experiments on a Hadoop cluster with four nodes.

Chapter 4 Results

Experiment 1 Devising algorithm for determining pThresh and vThresh

To determine pThresh and vThresh, we run the algorithm on a set of documents with known clusters. We run the algorithm on the same set of documents multiple times using different values for pThresh and vThresh. We then calculate the error value to determine the optimal values for pThresh and vThresh.

We have compiled a list of pairs of equivalent statements. The list we created by downloading public articles from news.google.com (Google news auto categorizes "similar" news pieces from different sources).

```
INITIALIZE Articles[] FROM DownloadedArticles
INITIALIZE ORIGINAL_CLUSTERS FROM Articles
FOR ALL pThresh = 0 TO 1
   FOR ALL vThresh = 0 TO 1
      CLUSTERS ← CATEGORIZE(Articles, pThresh, vThresh)
      ERROR ← CALCULATE_ERROR(ORIGINAL_CLUSTERS, CLUSTERS)
      OUTPUT (pThresh, vThresh, AVG(ERROR))
   vThresh ← vThresh + 0.01
   END FOR
   pThresh ← pThresh + 0.01
END FOR
```

Figure 4 Pseudo-Code for training algorithm

For this experiment:

- We have downloaded 66 articles from nine topics from google news. For each topic, google news API offers related subjects links. Reference clusters are computed based on these related links.
- Figure 4 lists the pseudo code for the training algorithm, details are as follows:
  - The algorithm first loads the articles previously downloaded from google news into ORIGINAL_CLUSTERS.
  - It then tries pairs of pThresh and vThresh from 0 → 1 with a step of 0.01.
  - For each pair, it runs the categorization algorithm and stores the results into CLUSTERS.
  - Then calculates the average error between ORIGINAL_CLUSTERS and CLUSTERS and outputs the triple (pThresh, vThresh, error).
- We then analyze the output to find the minimum error and the corresponding pThresh and vThresh.

```
FUNCTION CALCULATE_ERROR (ORIGINAL_CLUSTERS, CLUSTERS)
INITIALIZE SCORE ← 0
FOREACH ORIGINAL_CLUSTER IN ORIGINAL_CLUSTERS
  FOREACH ORIGINAL_DOCUMENT IN ORIGINAL_CLUSTER
    INDEX ← FIND_CLUSTER_INDEX(CLUSTERS, ORIGINAL_DOCUMENT)
    IF INDEX == CLUSTER_INDEX THEN
      SCORE ← SCORE + 1
    ELSE
      SCORE ← SCORE – 1
    END IF
  END FOR
  SCORE ← SCORE / NUM_OF_DOCUMENTS(ORIGINAL_CLUSTERS)
END FOR
```

Figure 5 Pseudo-Code for Error Evaluation Algorithm

How error is calculated

The average number of miss-labeled articles defines error for each run of the algorithm. Figure 5 lists the pseudo code for the error evaluation algorithm used.

The details of the algorithm (Figure 5) are as follows:

- The algorithm loops through all reference clusters.
- It checks to see if all documents in a reference cluster got clustered into the same result cluster
- For each correctly classified document it adds score one and for each wrongly classified one it subtracts 1 from score.
- Finally, it calculates the average score by dividing the score on the total number of documents from all clusters.

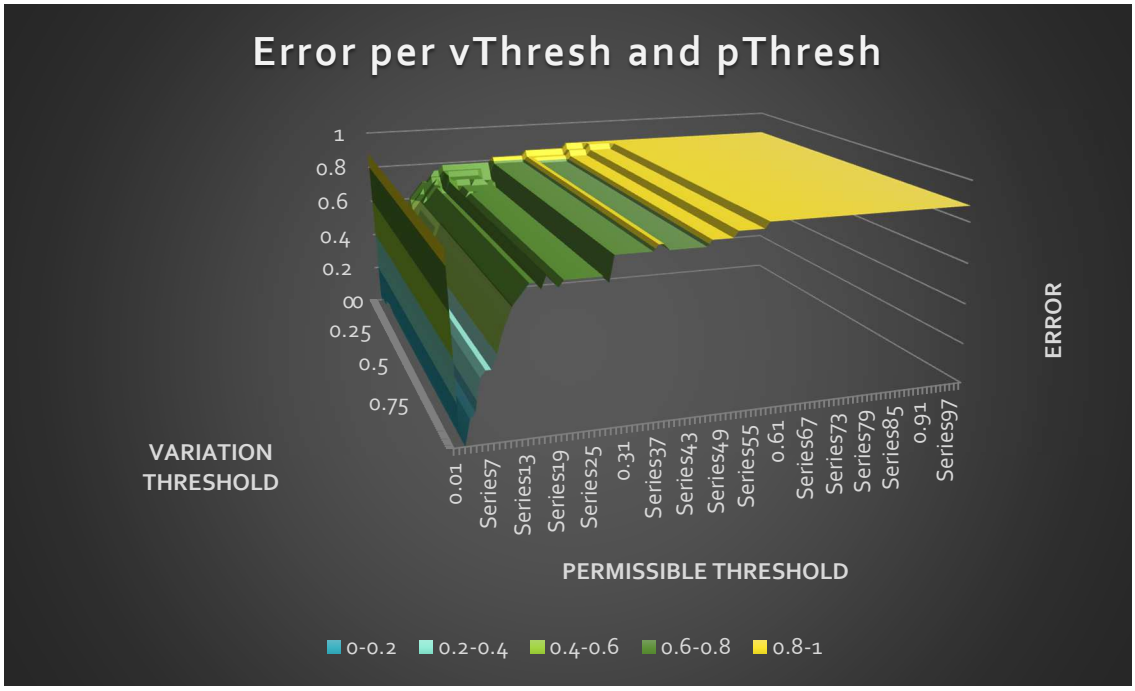

Figure 6 Average error for each pThresh/vThresh value in range 0-1 for input of size 66 documents.

Results and conclusion

In this graph (Figure 6), the x-axis (horizontal) represents permissible threshold values, z-axis (depth) represents variation threshold values and y-axis (vertical) represents the calculated average error for each pair.

Using the resulting excel sheet, the minimum error calculated is 0.03030303 that corresponds to pThresh and vThresh values of 0.11 and 0.02 respectively.

Experiment 2 Building the model

To build the full model, we had to obtain a comprehensive training set. The steps we followed are:

1) Obtain document training set

   We have looked at different sources for training data. The characteristics that we decided to look for are:

   1) **Diverse topics**: An important aspect of the data set is to include wide variety of topics. This enriches the resultant model. This also increases the chance that any pair of words in the test data set will have a score in the trained model.
   2) **Rich vocabulary**: It is also important for the training model to include as many known words as possible.

Wikipedia seemed a good fit for both of these requirements. The nature of Wikipedia (encyclopedia about everything, multiple different editors) makes it a very good representative of the set of all possible documents.

We have obtained a copy of English Wikipedia articles (42.4 GB). This includes only the latest version of all available articles in English as of 13th of February 2014. It includes around 750,000 distinct articles.

2) Obtain a comprehensive dictionary

Using stemming and stop word elimination only have proved not to be sufficient. After analyzing only 400 random articles, the algorithm discovered 56,000 unique words. These alone generated over 60million connections. It became evident that we need a white list of words to accept.

We have then obtained stemmed words from (12dicts) and (Ispell) dictionaries. After combining these dictionaries, we obtained a comprehensive list of 400,000 stemmed English words. Using the same 400 random sample articles, the algorithm discovered only 29,443 unique words and hence generated only 47,471,777 relationships between those words allowing the process to proceed further on a single computer node.

3) Obtain a test document set

We needed a clustered test document set to be able to calculate error value and observe the changes as we add more documents to the training set.

To obtain such a set, we acquired documents from Google News from nine hot topics at the time.

Table 7 Topics picked for testing dataset

Topics
barrack Obama

Philippines
incognito
satellite hit
Rouhani's past
kidnap
expulsion
church of government
doctor kill wife

This resulted into 100 document-set spanning nine clusters.

Results and conclusion

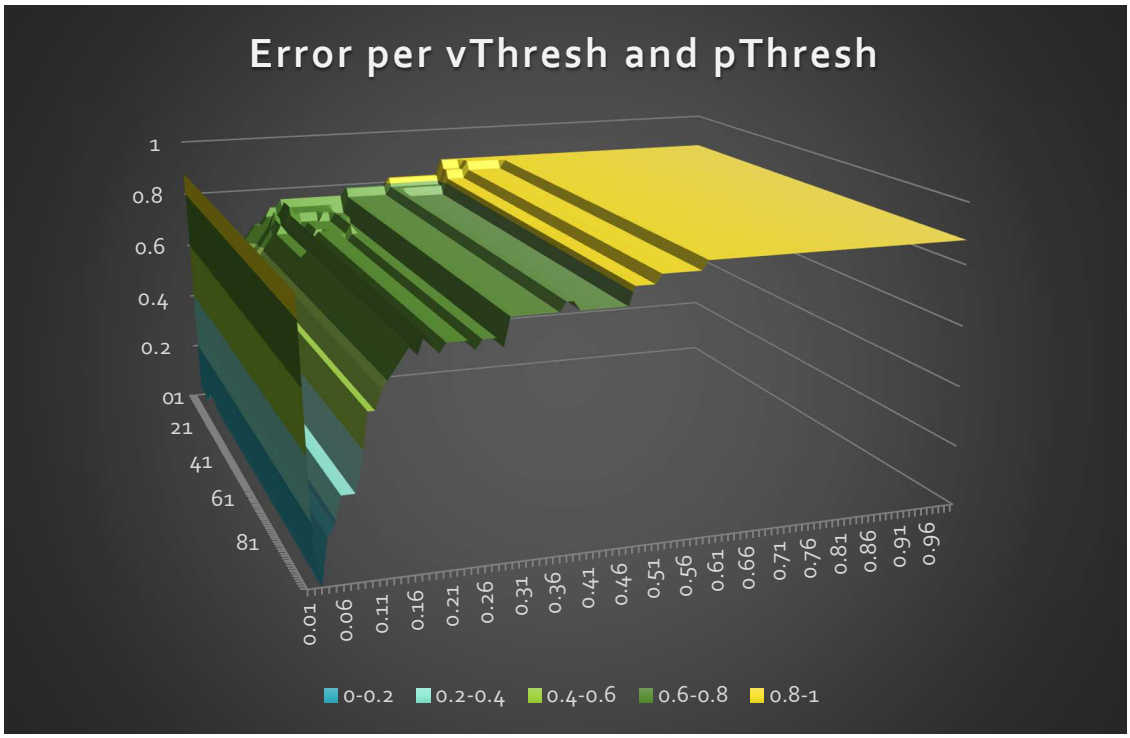

Figure 7 Average error for each pThresh/vThresh value in range 0-1 for input of size 1000 documents.

On a single processing node with Core i7 x64 process, we managed to process around 1000 articles before we hit memory allocation issues. The model generated consumed well over 4GB of memory.

Using the same pThresh and vThresh values obtained earlier, the resulting error was the same (i.e. 0.030303) (Figure 7).

Discussions

In this chapter we have run two experiments to train the algorithm. We ran the first experiment against a pre-categorized dataset in order to find reasonable values for the variation and permissible thresholds. Following the experiment, we ran a second experiment to build the model against the training dataset.

In running these experiments, we have made some observation and concluded the limitations of the algorithms used. These can be summarized below.

Observations

1) A very small pThresh (Permissible Threshold) value lead to a big Error value because it meant the algorithm is too strict (i.e. creates a separate cluster for each document).
2) A very large value for pThresh (Permissible Threshold) value also lead to a big Error value because it meant the algorithm is too permissive (i.e. ended up creating less clusters than needed).
3) Choosing a random set of documents from Wikipedia achieved the same accuracy as an over trained dataset (i.e. using input dataset as testing dataset).
4) Building the model using this algorithm is a memory consuming process.

5) The minimum achieved error value is 0.030303 that corresponds to pThresh and vThresh values of 0.11 and 0.02 respectively.

Limitations of the algorithms used

1) Training algorithm does not consider variation in dThresh value. We have determined dThresh value based on manual observation.
2) Error evaluation algorithm misses the case when a single result cluster contains all documents from all original clusters (e.g. if the clustering algorithm is too permissible).
3) Error evaluation algorithm should use set intersect to find the result cluster that share the most number of documents from original cluster.

The goal of these experiments was to devise and evaluate the algorithm for calculating pThresh and vThresh. Therefore, we have run the algorithm on a fraction of the document set prepared. We repeated the experiment after generating the full model.

Chapter 5 Conclusions, Implications, and Recommendations

The internet is growing at a massive scale and so is the amount of information an average person is exposed to. This dictates the need for an efficient algorithm to group documents into meaningful clusters in order to make the process of exploring them easier.

Conclusions

We have reviewed various algorithms with a wide range of approaches to solve this problem. Yang, Pierce & Carbonell (1998) arrived at using Group Average Clustering as the best algorithm among the ones they reviewed. Nallapati, et al (2004) exploited the chronological order to identify similar news events. Shah & Elbahesh (2004) used a hybrid approach between nearest K and single link to achieve better results. Li, et al (2007) ran their experiments on a large number of documents and showed that their devised content-based K means achieves best results.

We have then implemented a hierarchical clustering algorithm that incorporates these ideas. We built a model that can identify similar documents based on the frequency of pairs of words. We have achieved error value of 0.030303 (measured in the ratio of miss-categorized documents to the total number of documents). We used a single processing node to process 1000 documents from the training set consisting of 30 thousand unique stemmed words.

Implications

In conclusion, (Pera & Ng)'s fuzzy equivalence algorithm does produce acceptable results when compared to Google News as a reference. The algorithm, however, requires

a huge amount of memory to hold the trained model. This renders it not suitable to run on portable devices (e.g. phones) but very suitable to run on a server farm. Moreover, during the training phase, the algorithm builds and the model in memory and that makes it hard to process the full training dataset on a single processing node.

Recommendations

In order to overcome the memory constraints of the training algorithm, we have redesigned the algorithm to work on MapReduce pattern in order to run on Hadoop (Figure 8).

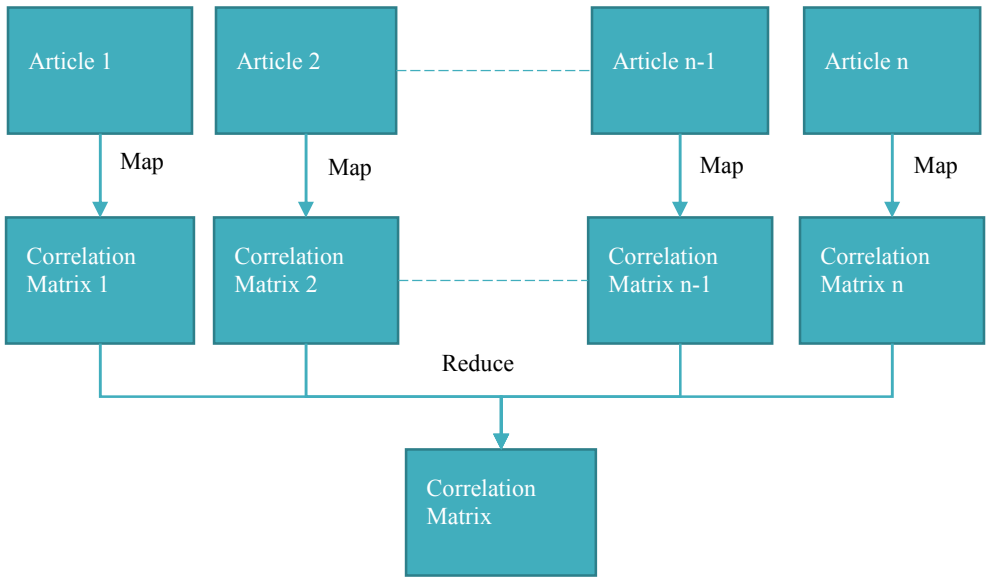

Figure 8 MapRedue architecture for training algorithm

To process all 750K articles in the training dataset, we will have to divide those 400 at a time (since that's the maximum achieved per single node), this leads to 1875 jobs, and each runs in around 10minutes. On a big Azure HD Insight cluster (48 nodes), this will take roughly 6.5 hours to complete.

We have successfully run a prototype job on Hadoop cluster to prove the viability of this experiment.

Appendix I. Bibliography

12dicts. (n.d.). Retrieved from http://prdownloads.sourceforge.net/wordlist/12dicts-4.0.zip

Barnett, A. (2005). Retrieved from http://blogs.msdn.com/b/alexbarn/archive/2005/10/08/478598.aspx

Ghwanmeh, S. H. (2005). Applying Clustering of Hierarchical K-means-like Algorithm on Arabic Language. *International Journal of Information Technology, 3*.

Gillick, D. (2009). Sentence Boundary Detection and the Problem with the U.S. *NAACL HLT* (pp. 241-244). Boulder, Colorado: Association for Computational Linguistics.

Google Inc. (n.d.). *Home Page*. Retrieved from Google News: http://news.google.com

Ispell. (n.d.). Retrieved from http://fmg-www.cs.ucla.edu/geoff/ispell.html

Li, X., Yan, J., Deng, Z., Fan, W., Zhang, B., & Chen, Z. (2007). A Novel Clustering-Based RSS Aggregator. *World Wide Web*, (pp. 1309-1310).

Nallapati, R., Feng, A., Peng, F., & Allan, J. (2004). Event Threading within News Topics. *CIKM*, (pp. 446-453).

Nazario, J. (2005, July 8). *RSS Clustering: A Unique Approach for Managing Your RSS Feeds.* Retrieved from Pearson InformIT: http://www.informit.com/articles/article.aspx?p=398884

Pera, M. S., & Ng, Y.-K. (n.d.). Synthesizing Correlated RSS News Articles Based on a Fuzzy Equivalence.

Sentz, K., & Ferson, S. (2002). Combination of Evidence in Dempster-Shafer Theory. *SAND*.

Shah, N. A., & ElBahesh, E. M. (2004). Topic-based clustering of news articles. *Proceedings of the 42nd annual Southeast regional conference*, (pp. 412 - 413). New York.

Survey, G. (n.d.). *Google Survey*. Retrieved from Google Survey: https://www.google.com/insights/consumersurveys/view?survey=lmyvojz46ogho&question=4&filter=&rw=1

The Porter Stemming Algorithm. (2006, January). Retrieved from http://tartarus.org/~martin/PorterStemmer/index.html

Topix LLC. (n.d.). *Home Page*. Retrieved from Topix: http://www.topix.net

Wanner, L. (2004, July 1). *Introduction to Clustering Techniques.* Retrieved from http://www.iula.upf.edu/materials/040701wanner.pdf

World Wide Web Size. (n.d.). Retrieved from http://www.worldwidewebsize.com/

Yang, Y., Pierce, T., & Carbonell, J. (1998). A Study on Retrospective and On-Line Event Detection. *ACM SIGIR*, (pp. 28-36).