Event-Centric Clustering of News Articles

Jon Borglund
Abstract

Event-Centric Clustering of News Articles

Jon Borglund

Entertainity AB plans to build a news service to provide news to end-users in an innovative way. The service must include a way to automatically group series of news from different sources and publications, based on the stories they are covering. This thesis include three contributions: a survey of known clustering methods, an evaluation of human versus human results when grouping news articles in an event-centric manner, and last an evaluation of an incremental clustering algorithm to see if it is possible to consider a reduced input size and still get a sufficient result. The conclusions are that the result of the human evaluation indicates that users are different enough to warrant a need to take that into account when evaluating algorithms. It is also important that this difference is considered when conducting cluster analysis to avoid overfitting. The evaluation of an incremental event-centric algorithm shows it is desirable to adjust the similarity threshold, depending on what result one want. When running tests with different input sizes, the result implies that a short summary of a news article is a natural feature selection when performing cluster analysis.
Contents

1 Introduction  

2 Theory and methods  
   2.1 Clustering  
   2.2 \textit{K}-means algorithm  
   2.3 Vector space model  
   2.4 Cosine similarity  
   2.5 Word stemming  
   2.6 Term frequency-inverse document frequency  
   2.7 Cluster validity  
      2.7.1 External criteria  
   2.8 Overfitting  

3 Human evaluation  
   3.1 Corpus  
   3.2 Sorting application  
   3.3 Evaluation  
   3.4 Results and discussion  
   3.5 Conclusions  

4 Algorithm evaluation  
   4.1 Algorithm  
   4.2 Ennobler  
   4.3 Modifications to algorithm  
   4.4 Results and discussion  
   4.5 Conclusions  

5 Future work  

6 Conclusions  

References
1 Introduction

On the Internet, news articles and related news information is rapidly spread from a variety of sources. *Entertainity AB* is planning to build an innovative service to provide news to the end-user based on different criteria, such as user location, user feedback etcetera. Basically a self-learning news portal that present what each user wants.

The service must include a way to automatically group series of news texts from different publications and sources, based on the event they are covering. This also includes new event detection; when a new article arrives it needs to either be clustered with an existing group or become a new one. In this way a user of the service can easily follow the development of a news story.

*Google News* is a similar service. It is a computer-generated news site that aggregates headlines from news sources worldwide, groups similar stories together and displays them according to each reader’s personalized interests [1].

Extensive research has been done in the field of document and news clustering [2, 3, 4, 5, 6, 7]. The angle in this thesis is to cluster news articles into event-centric clusters with follow up articles included, not just similar stories in general.

For example, the event of a Swedish cleaning lady who was wrongfully accused of stealing a train on Saltsjöbanan[8, 9]. The story went that the woman stole an empty train and derailed it into a building. As the story developed, it turned out that it might have been an accident. Later the cleaner was cleared of all suspicions. A follow-up article reported that the union representing the cleaner was about to sue the train operator because they had defamed her. The line of news reports related to this event goes on. In this case a cluster should contain all these follow-up stories, but not other stories about accidents on Saltsjöbanan, other unrelated train accidents in Sweden nor cleaning ladies stealing stuff.

Research has been done before when it comes to cluster analysis of news articles. For instance J. Azzopardi created an incremental event-centric clus-
tering algorithm[2]. The result was an efficient clustering algorithm that has high recall precision against a Google News corpus; thus the algorithm is assumed to be similar to the one Google News is using. On other more generic corpuses the algorithm performed rather poorly. The conclusion is that the algorithm is event-centric because it generates highly specific clusters. The approach of Azzopardi is to aggregate a number of news sources on the web through RSS (Rich Site Summary). Each incoming news report is then represented as a Bag-of-Words[10] with tf-idf (term frequency-inverse document frequency)[11, 12] weighting, more about this in Section 2. The actual clustering is then done with a modified version of the K-means[13] clustering algorithm.

Other work has been done by H. Toda and R. Kataoka, they use named entity extraction to cluster articles in relevant labels[3]. The work by O. Alonso, collects temporal information by extracting time from the content itself[14]. He discusses how search results can be clustered according to temporal aspects.

In this thesis contributes to this line of research with the following things:

- A survey that investigate known clustering methods, and an investigation that combine unknown combinations of known clustering methods.

- Evaluate recall precision with human versus human and implementation versus human, hence an evaluation anchored in reality and not only with other computer generated event corpuses.

- Is it possible to consider better precision over run-time? One of the aspects that I have chosen to investigate is how large the size of the input needs to be to be good enough.
2 Theory and methods

2.1 Clustering

One of the most important activities in data analysis is to group data into a set of categories or clusters[15]. The grouping is based on the data objects (also known as documents), similarity or dissimilarity. Similar objects should be in the same cluster, and in different clusters than the dissimilar ones.

There is no formal definition of clustering, but given a set of inputs \( X = \{x_1, \ldots, x_j, \ldots, x_N\} \), where \( x_j = (x_{j1}, x_{j2}, \ldots, x_{jd}) \in \mathbb{R}^d \) with each \( x_{ij} \) is called a feature.

Hard partitional clustering is when the clustering seeks to create a limited \( K \) number of clusters \( X \) where \( C = \{C_1, \ldots, C_K\}(K \leq N) \), and such that

\[
C_i \neq \emptyset, \ i = 1, \ldots, K \\
\bigcup_{i=1}^{K} C_i = X \\
C_i \cap C_j = \emptyset, \ i, j = 1, \ldots, K \text{ and } i \neq j
\]

Hierarchical clustering constructs a nested structure partition of \( X \), \( H = \{H_1, \ldots, H_Q\}(Q \leq N) \), such that \( C_i \in H_m, C_j = H_l \), and \( m > 1 \) imply \( C_i \subset C_j \) or \( C_i \cap C_j = \emptyset \) for all \( i, j \neq i, m, l = 1, \ldots, Q \).

There are four major steps in clustering[15].

Feature selection or extraction is the process to determine which attributes of the data object to use to distinguish objects. The difference is that the extraction derives new features from existing feature attributes.

Algorithm design is to determine the proximity measure and construct a criterion function. The data objects are intuitively clustered into different groups if they are similar or not according to the function.

Cluster validation assessments must be completely objective and have no preferences to any algorithm. Generally there are three types of testing criteria; external testing, internal indices and relative indices. They are defined as three types of clustering partitional clustering, hierarchical clustering and
individual clusters.

The goal of the clustering is to give the user meaningful insights of the data through result interpretation.

### 2.2 K-means algorithm

One of the most common partitional clustering algorithms is the K-means algorithm[15]. This algorithm seeks the optimal partition of the data by minimizing the sum of squared error through an iterative optimization method.

The basic K-means algorithm [16] can be seen in Algorithm 1. When a point is assigned to the closest centroid with a proximity measure that defines the notation of “closest”. Euclidean distance is often used for data points in euclidian space, while cosine similarity is more appropriate for high-dimensional positive spaces e.g. text documents. Given the proximity function cosine, the objective is to maximize the sum of the cosine similarity.

#### Algorithm 1 Basic K-means algorithm

1. Select K points as initial centroids
2. Repeat
3. Form K clusters by assigning each to its closest centroid.
4. Recompute the centroid of each cluster
5. Until centroids do not change

### 2.3 Vector space model

The Bag-of-Words model (BoW) is a simplified version of a vector space model[2, 17], the difference is that the vector space model can contain phrases and not only words. In the BoW a document or a sentence is translated to a vector of unordered words. The words are often translated into a simplified representation such as an index. The dimensionality is high since each term is one dimension in the vector. Often the words are weighted with a term frequency algorithm, for instance tf-idf described in Section 2.6.

To filter insignificant words, it is possible to use a list of stop words[18]. The list contains common and function words like the, you, an etcetera. Stop
words are language specific, but sometimes a list may even contain words to work better in specific topics. Using a list of stop words can reduce the complexity and improve performance.

### 2.4 Cosine similarity

As the dimensionality increases in the data, as it does with text mining and the BoW model, the cosine similarity is a commonly used similarity measure[16, 17]. Given two feature vectors $A$ and $B$ of same length, the cosine similarity between them is the same as their dot-product with magnitude[17]. The cosine similarity $cos(\theta)$ represented as dot product:

$$A \cdot B = \sum_{i=1}^{n} A_i \times B_i$$

and the magnitude:

$$||A|| ||B|| = \sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}$$

which gives the cosine similarity:

$$Sim(A, B) = cos(\theta) = \frac{A \cdot B}{||A|| ||B||}$$

The angle between the vectors is the divergence, and can be used as the numeric similarity. Cosine is 1.0 when the vectors are identical and 0.0 for orthogonal.

### 2.5 Word stemming

Word stemming reduces word to their so called root or base. The advantage of doing this in information retrieval is to reduce the data complexity and size. Word stemming is often used as query broadening in search systems[17]. Martin Porter is one of the pioneers in this field. Porter’s suffix stemming routine was released in 1980, and it became the standard algorithm for suffix stemming in the English language[19]. The algorithm was however often
interpreted in ways that introduced errors. Porters solution to this was to release his own implementation which he later developed into the Snowball framework.

Suffix stemming of words is the procedure to stem words to their root, base or stem form. Example given the words[19]:

CONNECT
CONNECTED
CONNECTING
CONNECTION
CONNECTIONS

A stemmer automatically removes the various suffixes -ED, -ING, -ION, IONS to leave the base term CONNECT. Porter’s algorithm works by removing simple suffixes in a number of steps, in this way even complex suffixes can be removed effectively.

2.6 Term frequency-inverse document frequency

Term frequency-inverse document frequency (tf-idf) is one of the most commonly used term weighting schemes in information retrieval systems[11]. Tf-idf is used to determine the importance of terms for a document relative to the document collection it belongs to. This can for example be used to produce tag-clouds. A tag-cloud is where the words of a text are listed in random order, but the font size of the words depends on the relevance, for example tf-idf weights.

The tf-idf product is calculated with the term frequency (tf) and the inverse document frequency (idf)[12, 11]. The term frequency function $tf(t, d)$ where $t$ is the term and the $d$ is the document. The $tf$ can be represented as the raw frequency $f(t, d)$, that is the number of times $t$ is in document $d$. Some of the other ways to represent the the frequency are[20];

Boolean where:

$$tf(t, d) = \begin{cases} 
\text{true} & \text{if } t \subseteq d \\
\text{false} & \text{if } t \notin d 
\end{cases}$$
Logarithmically scaled where:

\[
    tf(t, d) = \begin{cases} 
        1 + \log(f(t, d)) & \text{if } f(t, d) > 0 \\
        0 & \text{if } f(t, d) = 0 
    \end{cases}
\]

Augmented frequency which prevent longer documents to get a higher weighting:

\[
    tf(t, d) = \frac{1}{2} + \frac{f(t, d)}{2 \times \max \{f(w, d) : w \in d\}}
\]

The idf function \( idf(t, D) \) is a measure of how common the term \( t \) is across all the documents in the collection \( D \).

\[
    idf(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}
\]

The \( tf-idf \) product is then calculated with:

\[
    tfidf(t, d, D) = tf(t, d) \times idf(t, D)
\]

When tf-idf is high it means that the term \( t \) is present in the document \( d \), it also implies that either \( t \) is a very uncommon term in the collection \( D \) or that the term frequency \( tf \) is high in the document \( d \).

### 2.7 Cluster validity

One of the greatest problems with clustering is to objectively and quantitatively evaluate the result, another problem is to determine if the clusters derived are meaningful. The determination of such underlying features is called cluster validation[3].

#### 2.7.1 External criteria

External criteria is used to evaluate two given clustering sets which are independent of each other. Given \( P \) as a partition of data set \( D \) with \( N \) data objects which is independent of the clustering result \( C \). The evaluation of \( C \)
by external criteria can be conducted by comparing $C$ to $P$. Take a pair of data objects $\{d_i, d_j\} \subset D$ which are present in both $C$ and $P$ [15, 21]:

**Case1:** $d_i$ and $d_j$ are in the same cluster of $C$ and the same group of $P$.

**Case2:** $d_i$ and $d_j$ are in the same cluster of $C$ and the different groups of $P$.

**Case3:** $d_i$ and $d_j$ are in the different clusters of $C$ and the same group of $P$.

**Case4:** $d_i$ and $d_j$ are in the different clusters of $C$ and the different groups of $P$.

The sum of each of these cases represents a fallout, namely:

**Case1:** *True Positives (TP)*

**Case2:** *False Negatives (FN)*

**Case3:** *False Positives (FP)*

**Case3:** *True Negatives (TN)*

Rijsbergen/22/ writes about information retrieval and describes recall and precision during queries as;

- recall as the proportion of relevant material actually retrieved.
- precision as the proportion of retrieved material that is actually relevant.

The *precision* ($P$) and *recall* ($R$) can be defined as followed:

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

To get a weighted average the *recall* and *precision* measurements are combined in the *F-score* also known as *F-measure*/22/, the formula is given by:

8
\[ F_1 = 2 \times \frac{P \times R}{P + R} \]

Rijsbergen also uses the \( \beta \) factor to weight the relative importance of recall and precision in the formula.

\[ F_{\beta} = \frac{(\beta^2 + 1) \cdot P \cdot R}{\beta^2 \cdot P + R} \]

The Jaccard Index, also known as the Jaccard coefficient is a statistic used for comparing the similarity and diversity of sample sets[21]. The measurement between two sample sets is defined as the size of the intersection divided by the size of the union of the sample sets \( A \) and \( B \), this is formulated as:

\[ J(A, B) = \frac{|A \cap B|}{|A \cup B|} \]

The dissimilarity of two sample sets can be measured with the Jaccard distance, hence it is the complementary to the Jaccard coefficient. The distance is calculated by subtracting the Jaccard index from 1.

\[ J_\delta = 1 - J(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} \]

The Jaccard similarity coefficient \( J \) is then given by:

\[ J = \frac{TP}{TP + FP + FN} \]

The Jaccard distance \( J' \) is then given by:

\[ J' = \frac{FP + FN}{FP + FN + TP} \]

2.8 Overfitting

Overfitting is a problem of great importance especially in the machine learning and data mining fields [16]. When overfitting becomes apparent the algorithm perfectly fits the training data, but has lost the ability to generalize, hence to successfully recognize new data[23]. This can also happen
when fine tuning the input parameters in unsupervised algorithms such as K-means, by choosing too many predefined clusters \( K \).

To avoid overfitting in training cases it is possible to use methods like cross-validation or early stopping. Cross-validation is often used when the training data is very limited\cite{23}. The data is randomly split into \( n \)-folds where one of the sets is used as evaluation set and the other \( n - 1 \) as training sets.

In the case of early stopping the data set is initially split in two sets, one training set and a validation set. The algorithm perform the training on the new training set, then it validates on the validation set. It repeats the splitting and training until the result start to degenerate, then the algorithm halts.

3 Human evaluation

The idea to conduct a human evaluation is to compare how similar or dissimilar human users are when sorting news articles in an even-centric manner. The articles in one cluster should describe the same event, and there should be only one specific partition for each event. The result is measured with external validation criteria against the default clustering partition. The evaluation can answer the following questions

- If the predefined cluster set which is used for evaluation is event-centric or not?
- If there is a universal truth and how it is defined?
- If the deviance between individuals should be considered when performing cluster analysis?

The result can also reveal if the initial cluster partition used is good enough.

3.1 Corpus

The data for the human evaluation is a corpus gathered from the Google News RSS-feed during one week, the period between 2013-03-18 and 2013-
03-24, the statistics of the corpus is seen in Table 1. A similar corpus was used by Azzopardi[2] to evaluate his algorithm.

<table>
<thead>
<tr>
<th>Documents</th>
<th>Clusters</th>
<th>Period in days</th>
</tr>
</thead>
<tbody>
<tr>
<td>4440</td>
<td>960</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 1: *Google news* corpus used for human evaluation

### 3.2 Sorting application

To compare the dissimilarity between humans, each one needs to construct their own set of clusters. The ideal solution would be to have all articles unsorted, which then each human sorts completely manually. Such a sorting task would however take too long time to conduct. The approach used is instead to use the *Google News* corpus described in Section 3.1 as a default starting point. The similarity between the predefined clusters are computed with the BoW model weighted with tf-idf and the cosine similarity, hence a incidence list is created for each cluster.

The participants was asked to check if each cluster was deemed event-centric, hence each cluster was about the same news event. To aid the humans in their manual labor of sorting news articles into clusters, a web application was implemented, the interface for this can be seen in Figure 1.
To the left all the clusters are shown, in the middle the documents from the current selected cluster is shown, to the right it is possible to see the content of the documents. The procedure of the sorting is then done in the following steps:

1. The user loads the predefined clusters.
2. The user selects one of the clusters.
3. The user determines if all the documents within the cluster are about the same news event.
   
   (a) If there are one or more irrelevant documents in the cluster, the user has the option to move them to another similar cluster or isolate them in a single document cluster. To find other similar clusters there is a possibility to filter visible groups on the left by a similarity threshold.
   
   (b) Else if all the documents are deemed relevant then the user can highlight the cluster as “human”
4. Repeat step 2 to 3
The default filter threshold is set to 0.1, which is relatively low. If the threshold is too high it will show too few similar ones, which would make the application too biased. This approach is still probably biased toward the default corpus, but it does not matter. It is not the users that are close to the original clustering that is relevant, the importance is to check if there is a difference, even if the application is biased.

The benefit of using a web application is that several users can conduct the survey simultaneous, the disadvantage is that it is hard to supervise the procedure.

### 3.3 Evaluation

The dissimilarity between each users’ cluster sets and the Google news clusters are compared with the external evaluation methods *Jaccard index* and *F*-measure.

To compute the similarity of user $A$ and user $B$, the documents used in the measurement is defined as the intersection of both users documents, $D_{AB} = D_A \cap D_B$, this is to be sure that both users have evaluated the documents included in the estimate. Document pairs are constructed for each user, e.g. for user $A$, where each pair is defined as $\{d_i, d_j\} \subset D_{AB}$, meaning that the user $A$ has clustered the document $d_i$ in the same cluster as document $d_j$, where $d_i < d_j$.

### 3.4 Results and discussion

The total number of clustered documents and clusters for each user can be seen in Figure 2. The variation seen between users’ number of documents, is due to each participating user did as much as they had time to do, meaning that even with the aid of the sorting application described in Section 3.2 the sorting was time consuming.
Figure 2: User statistics
In Figures 3 the similarity is measured with increasing number of documents in order of document id (primary key). In correlation with each users document count in Figure 4, it is possible to deduce several possibilities.

Some of the users are relatively close to the default clustering, this is, as mentioned, most likely because the sorting application makes the result biased towards it. One example is that some clusters can be perceived as more complex than others, if a subject then skips it and continues with another cluster which is virtually “ok” without adjustments. In the end such a user would be a perfect match with the default clustering.

The dip near the beginning of Figure 3 is caused by of one cluster that almost all users have adjusted. The great impact on the lines is because there is not much data available at this point. It is also possible to see that some users correlate when the similarity decrease in the same pattern. On the other hand some users diverse from the default clustering others stay close to it.

![Figure 3: Users compared to the default cluster set, Jaccard index similarity versus number of increasing documents in order of document id.](image-url)
All the users’ document count increase in the beginning of the corpus, this can be seen in Figure 4. This is implies that the users have clustered the same documents.

![Figure 4: Users document count increase over the corpus.](image)

In Figure 5 we can see the differences between precision, recall and f-measure. The documents measure is the fraction of documents processed in relation to the user whom have sorted the most documents. Several of the users have high recall, this indicates that they have the same pairs as the default partitioning, meaning that they have not split many clusters to smaller ones. A drop of precision can mean that the user have merged clusters, such an act would not affect the recall, or that the user have moved documents to another cluster, but then the recall is affected also.
The similarity matrix in Figure 6 shows the result of the user versus user evaluation. In this figure the Google news partition is included as User 1. Each element in the matrix represents the similarity between a pair of users. The number in each element represents the size of the document intersection. The light colored parts are where the similarity is high, darker parts means that the dissimilarity increases.

The shade of the matrix means that there are differences between users, the lighter parts also indicates that some users are similar. The number of documents in some of the intersections are low, this is most likely because the users have done different areas of the corpus, or if one of the users has organized an insufficient number of documents.
Figure 6: User versus user similarity matrix Jaccard index.

3.5 Conclusions

From the results it is possible to draw the conclusion that human users are in fact different. To be able to derive a measurement that is useable, this needs to be further investigated.

The result indicates that it is not possible to find one universal clustering result that fit all users perfectly. This should be considered when performing clustering analysis to avoid overfitting, e.g. when it is sane to stop the optimization of a clustering algorithm.

One thing that should have been considered before conducting the user survey would have been to randomly distort the default clustering, for ex-
ample split clusters, move documents to similar clusters and so on. Then it would have been easy to see if a participant is “lazy” or not. A “lazy” user would probably have stayed at the random line. It would also have been possible to see if the default clusters are nearest to the universal truth. Even though this was not considered here, the result still indicates that users are different enough to warrant a need to take that into account.

4 Algorithm evaluation

The cluster algorithms are evaluated with the same methodology as in the human evaluation in Section 3.3 and the same corpus. Other commonly used corpuses (like Reuters) are not investigated since they are not clustered by the events its articles describe. The evaluation consists of measurements between the recall and precision with varied threshold, and short versus long texts. A comparison between the online incremental clustering algorithm and a modified one that re-calculates the whole cluster set when documents are added.

4.1 Algorithm

The baseline algorithm that is used in this thesis work is like Azzopardi’s incremental clustering algorithm[2]. The solution aggregate a number of news sources on the web through RSS, with a Bag-of-Words approach, filtered with stop words and Porter’s suffix stemming routine. The terms are weighted with tf-idf measure. The news reports are indexed as term vectors. Each cluster is represented as a centroid vector which is an average of news reports’ terms included in the cluster. The similarity between a report and the clusters are calculated with the cosine similarity measure. Combinations like these are widely used in information retrieval[15, 17].

The clustering technique used is the strict partitioning clustering, and it is adaptive, derived from the k-means clustering algorithm, meaning that the number of clusters do not need to be known before starting the clustering. A similarity threshold is set to determine if an article belongs to a cluster, each
article can only belong to one cluster. If a similar cluster is not found a new one is created. To keep resources free and avoid clustering with obsolete news events, old clusters are frozen after a certain time. A cluster is considered old or dormant when no new articles has been added to it within a specific time period. When the clusters are frozen they are removed from the clustering.

The incremental solution does not perform well on general corpuses. It performs very well on a Google News corpus, therefore it is deemed to be very event-centric. The corpus consists of 1561 news reports downloaded in January 2011 that are covering 205 different news events (clusters). The result of the algorithm reaches a recall of 0.7931, precision of 0.9518 and a f-measure of 0.8370. Compared to the human evaluation in Section 3, this result is good.

There are a few input parameters that can be adjusted in this approach;

- The time that should have passed before a cluster is considered to be idle and therefore inactivated. Remember that a cluster in this case is a news event.

- How many documents that should be processed by the tf-idf, before starting the actual clustering. If the clustering starts directly the term weights provided by tf-idf would be more or less invalid.

- The similarity threshold, that is at which similarity a document should be placed in a cluster.

4.2 Ennobler

The news article corpus that is used in this evaluation is collected from Google News RSS feed, as described in Section 3.1. In the RSS-feed the text content is short. To retrieve more data from the original source an “ennobler” is needed.

The idea is to use the link provided in the RSS to get the html-code from the original source, parse it and try to find relevant information. The best would be to write a specific parser for each news source, but since this would take too much time the solution needed to be generalized.
Most pages are built using the div-tag, therefore the parsing is done by parsing each div-tag, and finding the inner most div that contains most of the terms of the RSS-content. If the div contains paragraphs they are collected, otherwise the entire text contents of the div is collected.

To determine if the contents $C$ of a div contain the relevant text, each present term $t_i$ from snippet $S$, is counted. The length of $S$ is defined as $n$.

$$\text{TermIsPresent}(t, C) = \begin{cases} 1 & \text{if } t \in C \\ 0 & \text{if } t \notin C \end{cases}$$

$$\text{Sim} = \frac{\sum_{i=1}^{n} \text{TermIsPresent}(t_i, C)}{n}$$

The similarity fraction $\text{Sim}$ can then be used to determine if the data is relevant or not.

### 4.3 Modifications to algorithm

To evaluate improvements to the baseline algorithm the following modifications are made;

- If the incremental features of the baseline algorithm are not used. This could be the case if the clustering is suppose to run on a central server with enough hardware to compute the whole cluster set each time a new article is added. That would mean that all documents are processed with tf-idf before the clustering starts. The difference is that newly added documents will also be included in the overall term weighting.

- During the clustering of an article, the similarity is compared to existing clusters' centroids until a fixed similarity threshold is reached, or else it is put in a new cluster. This is a first come first serve methodology, which means that when the similarity threshold is reached there may still exist better matching clusters. Cluster may become dormant, this is when no new articles are added to the cluster for a specific period of time, then it will be frozen, removed from the further updates. One possible solution to this is to on a regular basis compare the sim-
ilarities between existing clusters. If two clusters are deemed similar then they should be merged.

4.4 Results and discussion

The number of documents needed before starting the clustering is set to 140, this to be sure that the tf-idf has sufficient data to be able to weight terms correctly. The number of active clusters is set to 150 (news events). The terms are weighted with tf-idf and the augmented term frequency. In Figure 7 the recall and precision is measured against the Google corpus as the initial partition. The **snipped** source is based on the original RSS content and title, while the **text** is based on the ennobled versions of the text. The result is then measured with the cosine similarity threshold varied between 0.5 and 0.01 with the interval of 0.01.

A high threshold results in a high precision and low recall, meaning that almost every document is put in their own cluster, but the few documents that are grouped is in the same clusters as the original partition. As the similarity threshold decreases the precision decreases and the recall increases. When the threshold approaches 0.01 all documents are distributed in fewer and fewer clusters. The low threshold makes the articles end up in irrelevant clusters, since it takes the first available cluster in the list, which leads to that recall decreases slightly before it increases asymptotically to 1.

The precision of the large text is slightly lower than the snippet version, this is most likely because of the sometimes poor quality of the text that the automatic ennobler fetches. The snippet describes the core of the article, and with the ennobler distortions (like the fact that user comments can be parsed too) introduces information that might not be relevant.

In Figure 8 it is possible to see that the full text version has slightly better F-score. The impact on running time is however more significant than this nudge in performance. A benefit of the full text is that the precision is much more stable than the snippet. The span of the threshold where the F-score is high, is much longer.
Figure 7: Variable cosine similarity threshold from 0.5 to 0.01, recall vs precision
The result seen here has a F-score 0.14 lower than the one seen in previous work[2]. The cause of this can be the cluster ratio of the corpus used before was $1561 \div 205 \approx 7.61$ articles per cluster, while in this evaluation the ratio is $4440 \div 960 \approx 4.63$ articles per cluster.

To investigate if the algorithm behaves differently when running on only a set of 205 preselected news events, the following test was evaluated. Selecting the documents from 205 clusters in the mid range cluster sizes, that is ignoring the extremes (skipping the top 20 largest clusters), the ratio becomes $2352 \div 205 \approx 11.5$. Running the algorithm on this set of reports yields in a F-measure of 0.77 with the snippet version, that is not far from 83.4. But in the real world it is not possible to preselect a number of news events
The results of the modified none incremental algorithm can be seen in Figure 9 and 10. This means that the benefit of clustering the entire corpus is not rendering any tremendous increase in performance, but the snippet is a bit more stable over precision like the text version was in the previous test (Figure 8).

Figure 9: Pre-calculated tf-idf, variable cosine similarity threshold from 0.5 to 0.01, recall vs precision
To investigate if merging of clusters yields better performance, the algorithm is modified to do the merging before freezing idle clusters. The merging is done by comparing the two cluster centroids. The cosine similarity threshold for merging is set to 0.2 higher than the variable clustering threshold. The Figures 11 and 12 show the result of this. The result is a bit higher at its peak, but overall it seems more unstable and random.
Figure 11: Merge clusters, variable cosine similarity threshold from 0.5 to 0.1, recall vs precision
Figure 12: Merge clusters, F-Score versus cosine similarity threshold, 0.1 to 0.5

4.5 Conclusions

The run time complexity of the algorithm is dependent on the input size, that is the number of term in each article. Figure 8 shows that increasing the input size has little impact on the performance of the algorithm. The main causes of this are likely the poor quality of the data the ennobler fetches, and that the short text from the RSS-feed is a naturally good feature selection for the news reports. This means that it is probably better to stay with the snippet size in the majority of cases.

Figure 7 shows that it is possible to set the threshold to benefit either
precision or recall. High precision will result in clean clusters but perhaps with relevant articles missing. High recall will on the other hand give more news reports in each cluster, but with articles present which do not belong there. This can be utilized in real world implementations depending on what kind of result one wants.

5 Future work

- Conduct a human survey which is for the subjects both less complex to comprehend and less time consuming. Make it available online to meet a greater and broader population. Use randomness if there is an initial set of clusters, this should make it easy to filter “lazy” users. From this a measurement that can be used to evaluate the clustering sets can be derived.

- To bridge the difference between users dissimilarities, future experiments with machine learning could be conducted to explore if it is possible to personalize the clustering to each users preferences.

- Construct specific crawlers for each news source, in this way it could also be possible to evaluate an potential automatic ennobler. A specific crawler could also have the option to follow related links to collect even more information.

- One point that has become apparent, is that there are articles that are overlapping between events. Instead of just strict partitioning clustering the analysis should also investigate overlapping clustering.

6 Conclusions

The human evaluation seen in Section 3 shows that some users organizes news reports differently when asked to sort them in an event-centric manner. Therefore it is probably wise to consider personalization when clustering news reports, in the future. The result also show that none of the users
abruptly diverges from the default cluster partition, making it useful for external cluster validation in an event-centric clustering algorithm.

The evaluation of the clustering algorithm (Section 4) shows that the baseline clustering algorithm performs rather well on the default data partitions. The result also imply that a small snippet that introduces an article, is a natural feature selection for the clustering. The result reveals that if utilizing the algorithm in the real world, it is desirable to adjust the similarity threshold, to get either small but pure clusters or clusters with high recall.
References


33