Analysis of Statistical Keyword Extraction Methods for Incremental Clustering

Rafael Geraldeli Rossi\textsuperscript{1}, Ricardo Marcondes Marcacini\textsuperscript{2,1}, Solange Oliveira Rezende\textsuperscript{1}

\textsuperscript{1}Institute of Mathematics and Computer Science - University of Sao Paulo
Sao Carlos – SP – Brazil
{ragero,solange}@icmc.usp.br

\textsuperscript{2}Federal University of Mato Grosso do Sul
Tres Lagoas – MS – Brazil
ricardo.marcacini@ufms.br

Abstract. Incremental clustering is a very useful approach to organize dynamic text collections. Due to the time/space restrictions for incremental clustering, the textual documents must be preprocessed to maintain only their most important information. Statistical keyword extraction methods from single documents are useful in this scenario. However, different statistical methods have different assumptions about the properties of keywords in a text, and different methods extract different set of keywords. In this paper we analyze the different methods for keyword extraction and the impact of the number of keywords on the quality of the incremental clustering. We also define a framework for statistical keyword extraction.

1. Introduction

Incremental clustering is a very useful approach to organize dynamic text collections, in which new textual documents are constantly published. Organizing dynamic text collection is a current challenge [Aggarwal and Zhai 2012, Jain 2010]. The traditional clustering approaches, which are non-incremental, consider that the text collection is static, and the clustering process is repeated when new documents are inserted into the textual collection. This is computationally costly, mainly for situations when frequent updates in the clustering are required. Methods for incremental clustering are very useful in this context since they allow to update existing clusters instead of generating them for each new document or set of documents [Xu and Wunsch 2008].

The process of incremental clustering might become slow, due to the calculation of similarities considering the large number of different words contained in a (dynamic) text collection, and might consequently consume a large amount of memory. Besides, the documents of a domain share a lot of general words, which can affect the similarity computation and produce spurious results.

To solve or reduce the problems mentioned above, just the keywords of a document should be used instead of all the words. The keywords describe or summarize the document content concisely. Some authors, as [Liu and Wang 2007], believe that just the keywords must be considered as features of a document, discarding all the other words. Unfortunately, most of the documents do not have associated keywords. Due to this,
methods to automatically extract keywords are necessary, since the manual extraction of keywords from the documents in a collection or stream is unfeasible.

Some methods to extract keywords from documents need to analyze the entire document collection. These methods are considered “domain dependent”. Some examples of these methods are: TF-IDF (Term Frequency - Inverse document Frequency) [Salton and Buckley 1988], Mutual Information [Church and Hanks 1990] and Log-likelihood [Dunning 1993]. The domain dependent methods are not feasible for streams or large document collections, since they need to keep all the words in the memory to decide which word will be used. Besides, some methods of this type are supervised methods, i.e., they require labels for the documents, which are not provided in text clustering tasks.

On the other hand, there are keyword extraction methods that analyze just the content of each document individually. These methods do not require labeled documents and does not need to analyze the entire document collection, i.e., they are “domain independent”. Examples of these methods are: TF-ISF (Term Frequency - Inverse Sentence Frequency) [C. B. Martins and Rino 2001], CSI (Co-occurrence Statistical Information) [Matsuo and Ishizuka 2003], TextRank [Mihalcea and Tarau 2004], and Eccentricity-Based [Palshikar 2007]. The keyword extraction from single documents is very useful for i) large collections, in which the load of the entire collection in the memory to extract the keywords is sometimes impossible, and ii) incremental collections, in which the analysis of the entire collection to extract keywords for each new document is unfeasible.

The different keyword extraction methods found in literature have different assumptions about the properties of the keywords, which end up with different sets of keywords extract by the different methods. Thus, an analysis about which method is more appropriate for the incremental clustering task is necessary. Besides, a study about the number of keywords to improve or maintain the quality of the incremental clustering is also necessary, since the use of a little number of keywords might not maintain the quality of the incremental clustering and a large number of keywords might not have impact in the speed of the process. Then, this paper aims to analyze different methods for keyword extraction and analyze the impact of the different number of keywords extracted from documents in the quality of the incremental clustering.

Since the application of the keywords extracted aims to aid the incremental clustering, which can be applied in search engines, web sensors and so on, we used in this work just statistical methods for this task. The statistical methods can be applied to document written in any language and are faster than linguistic methods. In this paper we also present a framework for statistical keyword extraction.

The remainder of this paper is organized as follows. Section 2 describes the incremental clustering algorithms used in this paper. Section 3 details the statistical keyword extraction methods used in this paper. Section 4 presents the results obtained for incremental clustering obtained by different statistical keyword extraction methods with different number of extracted keywords. Finally, Section 5 presents the conclusions and future work.
2. Incremental Clustering

Incremental clustering is based on the assumption that the allocation of a new document in an existing cluster structure can be done without storing all documents in main memory [Maimon and Rokach 2005]. Formally, given a sequence of documents \( d_1, d_2, \ldots, d_n \), a partition \( P_{h+1} \) can be generated considering just a previous partition \( P_h \) and the new document \( d_i \) [Giraud 2000]. Moreover, the incremental clustering algorithms store only the cluster representatives (centroids) in main memory [Xu and Wunsch 2008].

**Leader-Follower** algorithm [Jain et al. 1999] is one of the simplest and commonly used methods for incremental clustering [Xu and Wunsch 2008, Aggarwal and Zhai 2012]. In this algorithm, the similarity of a new document with the existing clusters is calculated. If the similarity is higher than a user’s threshold \((m)\), the new document is allocated to the closest cluster. Otherwise, a new cluster is created for the new document. Usually the centroids of the clusters are used to compute the similarities. When a new document is allocated to a cluster, the respective centroid is adjusted.

Due to the time/space restrictions existents for incremental clustering, it is important to preprocess the documents to maintain only the most important information contained into them. In this way, keyword extraction methods are useful in this scenario.

3. Statistical Keyword Extraction

The statistical keyword extraction methods analyzed in this paper are based on a sentence-term matrix. In this matrix, each row corresponds to a sentence of a document and each term corresponds to a column. A term can be a single word or a set/sequence of words. We treat the set of terms as \( T = \{t_1, t_2, \ldots, t_N\} \) and the set of sentences as \( S = \{s_1, s_2, \ldots, s_M\} \) in which \( s_j \subseteq T \). If a term \( t_i \) occurs in a sentence \( s_j \), the value 1 is assigned to the corresponding cell of the matrix \((o_{t_i,s_j})\) and 0 otherwise. Table 1 shows an example of a sentence-term matrix.

<table>
<thead>
<tr>
<th></th>
<th>Term 1</th>
<th>Term 2</th>
<th>( \cdots )</th>
<th>Term N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentence 1</td>
<td>( o_{t_1,s_1} )</td>
<td>( o_{t_2,s_1} )</td>
<td>( \cdots )</td>
<td>( o_{t_N,s_1} )</td>
</tr>
<tr>
<td>Sentence 2</td>
<td>( o_{t_1,s_2} )</td>
<td>( o_{t_2,s_2} )</td>
<td>( \cdots )</td>
<td>( o_{t_N,s_2} )</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \ddots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>Sentence M</td>
<td>( o_{t_1,s_M} )</td>
<td>( o_{t_2,s_M} )</td>
<td>( \cdots )</td>
<td>( o_{t_N,s_M} )</td>
</tr>
</tbody>
</table>

In this paper, we define a framework for statistical keyword extraction. The framework is defined in 5 steps: i) preprocess the textual document, ii) generate the sentence-term matrix, iii) generate scores for each term extracted from the document, iv) sort the term scores, and v) extract the first \( k \) terms of the sorted term scores as keywords. Algorithm presents the inputs, output and the structures generated in each step of the proposed framework.

The preprocessing of a textual document corresponds to the basic steps of preprocessing carried out in Text Mining as the removal of stopwords and unnecessary characters, and word simplification. The difference among the keyword extraction methods is in the way that they compute the scores for the terms considering the
Algorithm 1: Framework for statistical keyword extraction.

```plaintext
input:
D - textual document; k - number of desired keywords.
output:
K - list of keywords.

/* Step 1: Preprocessing textual document */
1 D_preprocessed = Preprocessing(D)

/* Step 2: Generating sentence-term matrix */
2 sent_term[][] = GeneratingSentTermMatrix(D_preprocessed)

/* Step 3: Generating a score for each term of the sentence-term matrix */
3 scores[] = GeneratingScores(sent_term[][])

/* Step 4: Sorting the scores */
4 scores_sorted[] = SortingScores(scores[])

/* Step 5: Extracting the terms related to the first k positions of the vector scores_sorted[] */
5 K = ExtractingKeywords(scores_sorted[], k)
```

We evaluated 5 statistical methods to compute the scores of the terms: i) Most Frequent (MF), ii) Term Frequency - Inverse Sentence Frequency (TF-ISF) [C. B. Martins and Rino 2001], iii) Co-occurrence Statistical Information (CSI) [Matsuo and Ishizuka 2003], iv) Eccentricity-Based [Palshikar 2007] and v) TextRank [Mihalcea and Tarau 2004]. The first three methods consider solely the sentence-term matrix and the last two methods are based on graphs, which are generated through the sentence-term matrix.

A graph is defined as \( G = \langle V, E, W \rangle \) in which \( V \) represents the set of vertices, \( E \) represents the set of edges among the vertices and \( W \) represents the weights of the edges. The terms of the sentence-term matrix are the vertices of the graph, i.e., \( V = T \), and an edge between the terms \( t_i \) and \( t_j \) \((e_{t_i,t_j})\) is generated if they co-occur at least once. The weight of the edge is defined according to the characteristic of the algorithm. Some algorithms consider that the edge weight must be small for terms with high co-occurrence. In this case, Equation 1 must be used to set the edge weight. On the other hand, some algorithms consider that the edge weight must be high for terms with high co-occurrence. In this case, Equation 2 must be used.

\[
\begin{align*}
    w_{t_i,t_j} &= \begin{cases} 
    \frac{1}{\text{co-occurs}(t_i,t_j)} & \text{if } e_{t_i,t_j} \in E \\
    0 & \text{otherwise}
    \end{cases} \quad \text{(1)} \\
    w_{t_i,t_j} &= \begin{cases} 
    1 - \frac{1}{\text{co-occurs}(t_i,t_j)} & \text{if } e_{t_i,t_j} \in E \\
    1 & \text{otherwise}
    \end{cases} \quad \text{(2)}
\end{align*}
\]

The different assumptions of each method will extract different keywords. To illustrate this, we consider an example text composed by some paragraphs about data mining extracted from [Wikipedia 2013]. Table 2 presents the example text. We extract 10 keywords composed by single words considering the 5 different methods analyzed in this paper. The ranking of the extracted keywords is presented on Table 3. We notice that for most methods there are several common keywords. Only the method CSI presented a low number of keywords in comparison with the other used methods. We can also notice that some important words for Data Mining domain as “kdd”, “machin” and “learn” do not appear for all methods. On the other hand, important words as “data”,...
“mine”, “analysis”, “process”, “database” and “pattern” appear for almost all the methods. The next subsections will present the details of these methods.

### Table 2. Example text.

Data mining (the analysis step of the “Knowledge Discovery in Databases” process, or KDD), an interdisciplinary subfield of computer science, is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. Aside from the raw analysis step, it involves database and data management aspects, data pre-processing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating.

The actual data mining task is the automatic or semi-automatic analysis of large quantities of data to extract previously unknown interesting patterns such as groups of data records (cluster analysis), unusual records (anomaly detection) and dependencies (association rule mining). This usually involves using database techniques such as spatial indices. These patterns can then be seen as a kind of summary of the input data, and may be used in further analysis or, for example, in machine learning and predictive analytics. For example, the data mining step might identify multiple groups in the data, which can then be used to obtain more accurate prediction results by a decision support system. Neither the data collection, data preparation, nor result interpretation and reporting are part of the data mining step, but do belong to the overall KDD process as additional steps.

Data mining uses information from past data to analyze the outcome of a particular problem or situation that may arise. Data mining works to analyze data stored in data warehouses that are used to store that data that is being analyzed. That particular data may come from all parts of business, from the production to the management. Managers also use data mining to decide upon marketing strategies for their product. They can use data to compare and contrast among competitors. Data mining interprets its data into real-time analysis that can be used to increase sales, promote new product, or delete product that is not value-added to the company.

### Table 3. Keywords extract by different methods

<table>
<thead>
<tr>
<th>Rank</th>
<th>MF</th>
<th>TF-ISF</th>
<th>CSI</th>
<th>Excentricity</th>
<th>TextRank</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>data</td>
<td>data</td>
<td>extract</td>
<td>data</td>
<td>data</td>
</tr>
<tr>
<td>2nd</td>
<td>mine</td>
<td>mine</td>
<td>result</td>
<td>analysi</td>
<td>analysi</td>
</tr>
<tr>
<td>3rd</td>
<td>analysi</td>
<td>analysi</td>
<td>group</td>
<td>step</td>
<td>process</td>
</tr>
<tr>
<td>4th</td>
<td>process</td>
<td>step</td>
<td>analyz</td>
<td>process</td>
<td>step</td>
</tr>
<tr>
<td>5th</td>
<td>step</td>
<td>databas</td>
<td>product</td>
<td>databas</td>
<td>databas</td>
</tr>
<tr>
<td>6th</td>
<td>databas</td>
<td>process</td>
<td>part</td>
<td>pattern</td>
<td>involv</td>
</tr>
<tr>
<td>7th</td>
<td>product</td>
<td>kdd</td>
<td>automat</td>
<td>involv</td>
<td>patter</td>
</tr>
<tr>
<td>8th</td>
<td>pattern</td>
<td>comput</td>
<td>record</td>
<td>manage</td>
<td>discov</td>
</tr>
<tr>
<td>9th</td>
<td>involv</td>
<td>discover</td>
<td>inform</td>
<td>product</td>
<td>machine</td>
</tr>
<tr>
<td>10th</td>
<td>manag</td>
<td>pattern</td>
<td>interpret</td>
<td>kdd</td>
<td>learn</td>
</tr>
</tbody>
</table>

#### 3.1. Most Frequent

A simple measure to automatically extract keywords is to consider the most frequent terms as keywords. The score of the term $t_i$ is obtained by counting the number of occurrences of the term in the sentence-term matrix, i.e.:

$$score(t_i) = freq(t_i) = \sum_{s_j \in S} a_{i,j},$$

(3)

#### 3.2. Term Frequency - Inverse Sentence Frequency

The basic idea of TF-ISF (Term Frequency - Inverse Sentence Frequency) measure [C. B. Martins and Rino 2001] is to determine the score of a term according to its frequency and its distribution through the sentences of the document. The score of a term decreases if a term occurs in a large number of sentences in the document, since this can be a common term and do not characterize the content of the document. The TF-ISF score for a term $t_i$ is given by:

$$score(t_i) = TF - IDF(t_i) = freq(t_i) \ast \log \left( \frac{|S|}{freq(t_i)} \right).$$

(4)
3.3. Co-occurrence Statistical Information

CSI (Co-occurrence Statistical Information) measure [Matsuo and Ishizuka 2003] obtain scores for words using $\chi^2$ measure [Greenwood and Nikulin 1996]. $\chi^2$ measures how much the observed frequencies are different from the expected frequencies. For the keyword extraction problem, the $\chi^2$ measure for a term $t_i$ is given by:

$$\text{score}(t_i) = \chi^2(t_i) = \sum_{t_j \in \mathcal{T}} \frac{(\text{co-occ} (t_i, t_j) - \text{co-occ}(t_i)p(t_j))^2}{\text{co-occ}(t_i)p(t_j)} .$$

(5)

in which $p(t_j)$ is the probability of term $t_j$ occurs in the sentence-term matrix and $\text{co-occ}(t_i)$ is the total number of co-occurrences of the term $t_i$ with terms $t_j \in \mathcal{T}$. In this case, $\text{co-occ}(t_i, t_j)$ corresponds to the observed frequency and $\text{co-occ}(t_i)p(t_j)$ corresponds to the expected frequency.

Since a document is composed by sentences of varied length, terms that appear in long sentences tend to co-occur with more terms. Then, the authors of the CSI measure redefined $p(t_j)$ as the sum of the total number of terms in sentences where $t_j$ appears divided by the total number of terms in the document, and $\text{co-occ}(t_i)$ as the total number of terms in sentences where $t_i$ appears. Moreover, the value of the $\chi^2$ measure can be influenced by non important but adjunct terms. For instance, the term “step” co-occurs almost selectively with the frequent term “data mine” in the example text present on Table 2. The $\chi^2$ value for the term “step” can be high since it has a high dependence with the term “data mine”. To make the method more robust to this type of situation, the authors of the CSI measure subtracts from the $\chi^2(t_i)$ the maximum $\chi^2$ value for any $t_j \in \mathcal{T}$, i.e.:

$$\text{score}(t_i) = \chi^2(t_i) - \arg \max_{t_j \in \mathcal{T}} \left\{ \frac{(\text{freq}(t_i, t_j) - n_{t_i,t_j})^2}{n_{t_i}p_{t_j}} \right\} .$$

(6)

Using Equation 6, a low $\chi^2(t_i)$ value is generated if $t_i$ co-occurs selectively with only one term. The measure presents a high value if $t_i$ co-occurs selectively with more than one term, i.e., the term is relatively important in the document.

3.4. Eccentricity-Based

Eccentricity is a centrality measure, i.e., a measure which determines the importance of a node in a graph [Palshikar 2007]. According to eccentricity measure, a node is central if its distance to the most distance node is small. The distance between a term $t_i$ and term $t_j$ ($d(t_i, t_j)$) is given by the sum of the edge weights on the shortest path from $t_i$ to $t_j$ in $G$. Thus, the eccentricity of a term $t_i$ is given by:

$$\text{score}(t_i) = \epsilon(t_i) = \arg \max_{t_j \in \mathcal{T}} \{d(t_i, t_j)\} .$$

(7)

We highlight that any other centrality measure can be used in place of eccentricity. For instance, [Matsuo et al. 2001] used the vertex contribution to maintain a term network with a small world characteristic as centrality measure.
3.5. TextRank

TextRank algorithm [Mihalcea and Tarau 2004] is based on PageRank algorithm [Brin and Page 1998], which defines the importance of a vertex in the graph considering the importance of its connected objects. Thus, the score of a term \( t_i \) using TextRank algorithm is given by:

\[
\text{score}(t_i) = TR(t_i) = (1 - p) + p \times \sum_{e_{t_i,t_j} \in E} \frac{w_{t_i,t_j}}{\sum_{e_{t_j,t_k} \in E} w_{t_j,t_k}} TR(t_j).
\] (8)

in which \( p \) is a constant usually defined as 0.85 [Mihalcea and Tarau 2004]. Equation 8 is applied iteratively until convergence, that is, until the scores of the terms do not change too much or until a fixed number of iterations.

4. Experimental Evaluation

To evaluate the impact of the use of keywords in the incremental clustering results, we applied the 5 methods for statistical keyword extraction presented in Section 3 in 8 collections of computer science articles. The description of the textual document collections, the experiment configuration, evaluation criteria, results and discussion are presented in the next subsections.

4.1. Document Collections

As all the methods for statistical keyword extraction used in this paper are based on the sentence-term matrix, we choose 8 collections composed by proceedings of the ACM Digital Library\(^1\). These collections are composed by documents with a large number of sentences, allowing the extraction of more reliable results. Table 4 presents the categories and the number of document per category of the used textual document collections.

<table>
<thead>
<tr>
<th>Collect.</th>
<th>Class</th>
<th># Docs.</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM-1</td>
<td>3D Technologies</td>
<td>91</td>
<td>401</td>
</tr>
<tr>
<td></td>
<td>Visualization</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wireless Mobile Multimedia</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Solid and Physical Modeling</td>
<td>74</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software Engineering</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td>ACM-2</td>
<td>Rationality And Knowledge</td>
<td>86</td>
<td>411</td>
</tr>
<tr>
<td></td>
<td>Simulation</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software Reliability</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Visual Reality</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Web Intelligence</td>
<td>86</td>
<td></td>
</tr>
<tr>
<td>ACM-3</td>
<td>Computer Architecture Education</td>
<td>78</td>
<td>424</td>
</tr>
<tr>
<td></td>
<td>Networking And Communications Systems</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Privacy in the Electronic Society</td>
<td>98</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software and Performance</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Web Information and Data Management</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>ACM-4</td>
<td>Embedded Networked Sensor Systems</td>
<td>50</td>
<td>394</td>
</tr>
<tr>
<td></td>
<td>Information Retrieval</td>
<td>74</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Parallel Algorithms and Architectures</td>
<td>98</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Volume Visualization</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Web Accessibility</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>ACM-5</td>
<td>Tangible and Embedded Interaction</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Management of Data</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td></td>
<td>User Interface Software and Technology</td>
<td>103</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Information Technology Education</td>
<td>87</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Theory of Computing</td>
<td>103</td>
<td></td>
</tr>
<tr>
<td>ACM-6</td>
<td>Computational Geometry</td>
<td>89</td>
<td>439</td>
</tr>
<tr>
<td></td>
<td>Access Control Models and Technologies</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Computational Molecular Biology</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Parallel Programming</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Integrated Circuits and System Design</td>
<td>93</td>
<td></td>
</tr>
<tr>
<td>ACM-7</td>
<td>Database Systems</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Declarative Programming</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Parallel and Distributed Simulation</td>
<td>98</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mobile Systems Applications and Services</td>
<td>93</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Network and System Support for Games</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>ACM-8</td>
<td>Mobile Ad Hoc Networking &amp; Computing</td>
<td>90</td>
<td>495</td>
</tr>
<tr>
<td></td>
<td>Knowledge Discovery and Data Mining</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Embedded Systems</td>
<td>102</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HyperText and Hypermedia</td>
<td>93</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Microarchitecture</td>
<td>103</td>
<td></td>
</tr>
</tbody>
</table>

\(^1\)http://dl.acm.org/
4.2. Experiment Configuration and Evaluation Criteria

To extract the keywords we developed the Statistical Keyword Extraction Tool\(^2\) (SKET). All the preprocessing steps and the algorithms used in this paper are implemented in this tool. To build the sentence-term matrix we consider as sentence the set of words separated by a stop mark ("."", "?" or "!") and as terms the single words contained in the document. We removed from the sentence-term matrix the stopwords contained in the stoplist of the SKET and stemmed the terms using the Porter’s algorithm [Porter 1980].

The edge weights of the graphs used in Keyword and Eccentricity methods were obtained using Equation 1. The edge weights of the graph used in TextRank method were obtained using Equation 2. The value of the constant \(p\) for the TextRank method was set to 0.85 according to [Brin and Page 1998].

The minimum similarity threshold \(m\) of the incremental clustering was set for each method of keyword extraction. The used range of \(m\) values was \([0.05, 0.5]\) with 0.05 of increment. Since the results of the incremental clustering depend on the insertion order of the documents, the process were repeated with different insertion orders. The clustering process was repeated several times to attenuate random fluctuations of the evaluation results. We generate approximately 144700 partitions in the experiments. We used the cosine measure to compute the similarities of the documents since this is a recommended measure for this type of data [Tan et al. 2005].

An adaptation of the \textit{F-Measure} index [Manning et al. 2008] was used to evaluate the quality of the partitions generated by the incremental clustering algorithm. To compute the F-measure index, consider that:

- \(P\) is a partition;
- \(L_r\) represents a document set of a class \(r\); and
- \(G_i\) represents a document set of a cluster \(i\) that belongs to the partition \(P\).

Given a class \(L_r\) and a cluster \(G_i\), Precision (\(P(L_r, G_i)\)), Equation 9, Recall (\(R(L_r, G_i)\)), Equation 10, and \(F\) (\(F(L_r, G_i)\)), Equation 11, can be computed.

\[
P(L_r, G_i) = \frac{|L_r \cap G_i|}{|G_i|} \quad (9)\]
\[
R(L_r, G_i) = \frac{|L_r \cap G_i|}{|L_r|} \quad (10)\]
\[
F(L_r, G_i) = \frac{2 \times P(L_r, G_i) \times R(L_r, G_i)}{P(L_r, G_i) + R(L_r, G_i)} \quad (11)\]

\textit{F-Measure} selects for a certain class \(L_r\) the highest value obtained by some cluster of the partition \(P\), as presented in Equation 12. Then, the \textit{F-measure} of the partition is the sum of the \textit{F} value for the \(c\) classes of the collection divided by the number of documents \(n\), as presented in Equation 13. The \textit{F-Measure} value is 1 if the partition separates correctly the documents according to their class and 0 otherwise.

\[
F(L_r) = \max_{G_i \in H} F(L_r, G_i) \quad (12)\]
\[
\text{F-Measure} = \sum_{r=1}^{c} \frac{|L_r|}{n} F(L_r) \quad (13)\]

\(^2\)Statistical Keyword Extraction Tool is available at \url{http://sites.labic.icmc.usp.br/sket/sket.zip}. 


4.3. Results

The results are presented and discussed considering two aspects: i) the quality of the keywords extracted for each method and ii) the adequate number of keywords to represent the texts. The main results of the experimental evaluation are presented in Figure 1. For each collection we present the $F$-Measure value of the partition obtained by the different keyword extraction methods with the different number of keywords used. The $F$-Measure value using all the terms of the collections (All) is also presented.

![Figure 1](image1.png)

Figure 1. F-Measure values for the ACM collections obtained by the different keyword extraction methods and the different number of keywords used in this paper.
The CSI method obtained the worst results for the document collections used in this paper. We observe that CSI method tends to eliminate keywords that has high frequency in the text. The most frequent keywords of a document tend to occur in other documents of the same domain. Thus, eliminating these keywords spoils the similarity computation and consequently the quality of the partitions. Analogously, the same idea explains the fact that the simple MF method is capable to generate good keyword sets, specially for a reduced number of keyword (between 5 and 30). Eccentricity, TextRank and TF-ISF methods present a similar behavior for most of the cases.

Figure 2. Critical difference diagram for the different number of keywords used in this paper.
Figure 2 presents the critical difference diagrams among the different keyword extraction methods for the different number of keywords analyzed in this paper. In this diagrams, each method is sorted according to the rank of the Friedman’s Statistical Significant Test [Demsar 2006]. The methods connected by a line do not present statistical significant difference among them. We observe that the measures Eccentricity, TextRank and TF-ISF present competitive results in comparison with the MF method as the number of keywords increases. We also observe that 35 keywords are sufficient to maintain/improve the quality of the partitions for most of the collections. When using more than 35 keywords, the keyword extraction methods have a better ranking than the use of all the terms. We highlight that the Most Frequent measure is suitable to the analyzed scenarios due to the low computational cost to extract the keywords.

5. Conclusions and Future Work

In this paper we analyze the impact of the different statistical keyword extraction methods and the different number of keywords extracted per document in the incremental clustering task. In general, the keyword extraction methods improved/maintained the quality of incremental clustering when more than 35 keywords per document are used. Using more than 35 keywords, the statistical keyword extraction methods presented equivalent results, except the method Co-occurrence Statistical Information, which presented inferior results. In general, Most Frequent and TextRank methods obtained the first positions in the ranking of the statistical significance test for most of the analyzed scenarios.

As future works we intend to build a term graph using different correlation measures and apply other centrality measures to extract keywords. We also intend to use sliding windows instead of sentences to build an input matrix for the keywords extraction methods. With this, the methods can be applied to texts with a small number of sentences. An ensemble of the statistical keyword extraction methods will be also evaluated.

Acknowledgements

Grant 2011/12823-6 and 2011/19850-9, Sao Paulo Research Foundation (FAPESP).

References


