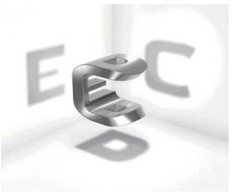
ECD Master Thesis Report



Textual Data Clustering and Cluster Naming

Marian-Andrei RIZOIU

05/06/2009

Supervision:

Julien VELCIN, laboratory ERIC, Univ. Lumière Lyon2, France
Jean-Huges CHAUCHAT, laboratory ERIC, Univ. Lumière Lyon2, France
Ştefan TRĂUŞAN-MATU, Univ. Politehnica Bucureşti, România

Location: Laboratory ERIC, Univ. Lumière Lyon2

Abstract:

In this paper we present the research a way of clustering textual data based on the thematics approached in the texts and a manner of finding a suitable, humanly readable name for each group. Previous research done on the field of data clustering and thematic extraction is briefly presented, along with observations of their suitability for the intended purpose, and then we propose an approach to combine the ones that we consider that maximize the effectiveness of the process. Our work is intended for general text files (newspaper articles, forums, chat logs) and takes into account the fact that a text can naturally have multiple thematics, so the clustering must be done in such a fashion that this condition is respected (a text can be part of more than one group).

The main idea is to regroup the textual documents using different term weighting schemes (a comparison of which will be presented later in the paper) and from each cluster extract the frequent keyphrases and associate them to the cluster's centroid.

A practical implementation of the algorithm has also been prepared and an expert evaluation was performed to assess the results.

Résumé :

Dans ce mémoire, nous présentons la recherche que nous avons mené dans le domaine du regroupement de données textuelles selon la thématique développée dans les textes en question et dans celui de la caractérisation de ces groupes d'une façon convenable et humainement lisible. Nous présentons brièvement les recherches des antérieures dans le domaine du regroupement de données et dans celui de l'extraction thématiques. Nous expliquons comment ajuster ces résultats à notre problématique, et nous proposons une facon de combiner les deux approches afin de maximiser l'efficacité de la procédure. Nous travaillons avec des fichiers textuels généraux (articles de journaux, forums, logs de chat), tout en considérant le fait qu'un texte peut naturellement avoir plusieurs thématiques (un texte est susceptible de faire partie de plusieurs groupes).

Au-delà de ce travail, nous utilisons plusieurs mesures pour regrouper les documents textuels (une comparaison est présentée plus tard dans ce dossier), et les motifs fréquents sont extraits de chaque groupe puis associés aux centres des groupes.

On a également préparé une implémentation pratique de l'algorithme, et une évaluation basée sur des experts a été utilisée pour juger des résultats obtenus.

Contents

1 Hosting Institution

This paper presents the conclusions of the research internship, which represents the final step towards completing Master 2 ECD studies. (Extraction de Connaissances à partir de Données). The Master is the result of an international collaboration of Universities in France with partner Universities in Romania, Vietnam, Canada. The internship is part of a cooperation between the University Politehnica Bucharest and the University Lumière Lyon2, ERIC laboratory, the supervision team being composed by two French professors and one Romanian. While the actual work was performed in the laboratory ERIC at Lyon, close contacts were kept with the Romanian part also.

Being an ERASMUS exchange student, doing a double diploma, means that the work will be presented and evaluated by committees both in France and in Romania.

The work is related to the research work performed by Mathilde Forestier which deals with the analysis of thematics in forums and can also be used in project of the enterprise Between and the thesis of Anna Stavrianou: the analysis of online chats. That created the premises of a very tight cooperation between the author of the paper and the PhD students, as well as with the representative of the Between Project Enterprise.

2 Acknowledgments

During my staying in Lyon and my work on the internship I had the great pleasure of working and interacting with the people from the ERIC laboratory, all of whom have actively supported and encouraged me during the work and adaptation to a totally new and different society and way of thinking and living.

First of all I would like to give my sincere appreciations to the professors from Master ECD with whom I had the pleasure to study, for introducing me to this new and fascinating domain of Data Mining and Artificial Intelligence.

I would like to give special thanks to Mr. Julien VELCIN and Mr. Jean-Hugues CHAUCHAT for their continuous guidance, patience and good will without which I could hardly finish the work. I also would like to mention their heroic efforts in understanding my terrible French accent. Also PhD students Mathilde FORESTIER and Anna STAVRIANOU for their continuous flow of ideas and help.

And last, but not least, to Mr. Djamel ZIGHED, head of the ERIC laboratory, for his continuous interest and prompt help to all the problems I was faced.

3 Introduction

The world today is faced with a challenge. It is an old acquaintance, but conditions have changed since we first encountered it: "**information**". Since humanity always felt the hunger for discovering the world around it, information can be found through out our history, from the beginning of the most ancient cultures. It has always been stored in libraries, carefully indexed and sorted by librarians. Its support (paper, tablets, papyrus) has always limited the amount of stored information so that a team of trained specialist could always deal with it.

But all changed in the last decades. The new storing technologies, the most important of all being the INTERNET and magnetic drives, made information storing notably easier, and the sharing capabilities of Internet encouraged people to create amount of information never seen before. And the thematics approached by the new information are more diverse: if at the beginning information would be philosophy, religion, history or art, lately huge quantities of personal experience, debates, social networks emerged. So it is safe to say that information increases at an **exponential rate** each year.

The new reality made impossible for human expert to manually sort out and inventor every single piece of knowledge. So this was the birth time of **unstructured information**, as opposed to **structured** one, which is manually maintained. This unstructured information is in the form of natural language text (*texts that have no predefined structure* - as opposed to tables for example) and usually found on various pages on the Internet. Unstructured information can be contained in forums, chat logs, blogs.

As the economical importance of the new type of information rose and in ten years (1996 - 2006) surpassed the information manually indexed - through the new emergence of profits by publicity on social networks, etc - so did the need of finding a way to automatize the process of Information Retrieval from Natural Language texts.

The purpose of this paper is to study the ways in which information can be retrieved from unstructured texts (natural language texts). Basically, by means of Unsupervised Machine Learning, we are searching for a way of dividing a set of texts into groups that are similar in terms of their thematics, meaning that all the texts in a group approach the same domain and there is a visible distinction between them and the texts from the other groups.

After this division is done, we are looking for a way of assigning to each group a meaningful name that best describes the content. In this way, the user can be presented not only with a partition of the texts, but also can easily identify the ones he is interested in. The work can also be used to identify the thematics approached in a group of texts: for example, in a forum, it is well known that multiple threads appear, resulting in parallel discussions about different subjects. Through the usage of our work, the thematics approached can be identified and the lines corresponding to each of them could be isolated.

There are several problems that present themselves. First of all, there is the pertinent observation in [4] that states that natural language texts can approach multiple domains. For example, a newspaper article talking about the economical consequences of a political decision can be considered both in the economical group, as well as in the politics group. Translated into our terms, that means that a text can be part of more than one group, the clustering algorithm must allow **overlapping**.

The second problem encountered was naming the groups: what makes a good name for groups? In [18] the problem is presented in detail. One of the first things that must be taken into consideration is that words have the propriety of polysemy, meaning that the same word can have different meaning in different contexts. For example each of the words "data" and "mining" have different words then the expression "data mining". Incompletely extracted phrases can cause ambiguity. Our thematics extraction algorithm must be able to take this observation into consideration.

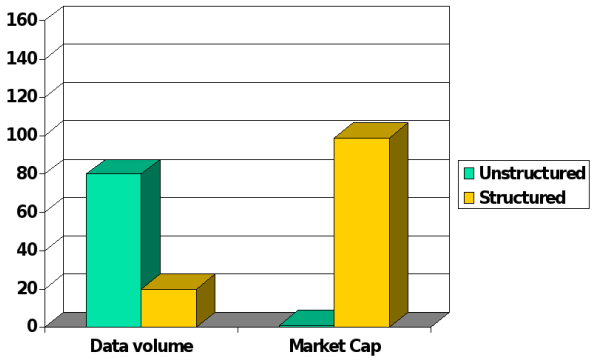


Figure 1: Market share of structured and unstructured information in 1996.

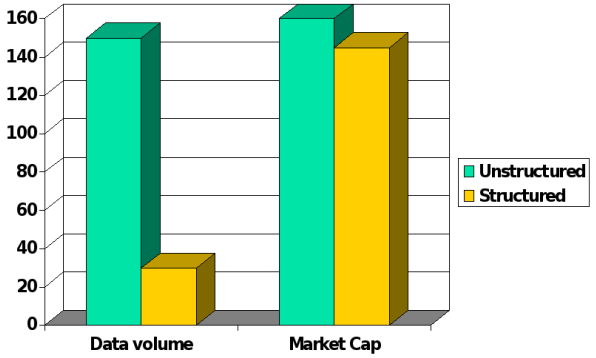


Figure 2: Market share of structured and unstructured information in 2006.

Also, prepositions and articles have a fundamental role in human comprehension of names, a correctly formulated phrase has more chances of being properly interpreted than a list of words. Example: "tales of war" vs. "tales, war"

This work was performed having in mind a couple of restrictions:

- to have an automatic algorithm, one which requires as little intervention as possible from the user;
- to be able to adapt to other languages of the text with minimal intervention and change;
- to be able to approach any thematic or field - not to be specialized on a certain domain;
- to be able to work with any kind of texts: scientific articles, newspaper articles, log chats, forums, etc. That implies that the chosen algorithm must be robust and noise resistant, as it is known that in non-formal discussions and writing people have the tendency of not respecting all the rules of the language. Writing mistakes often occur (eg. misspelling, not using the accent in French language, etc).

There are many ways to extract keyphrases from text, some of which will be presented in the next chapters, some linguistic, other statistical. In our work we chose a statistical one because they are less dependent on the language (only the pretreatment still requires language selection) and are more stable with informal texts than the linguistic ones. [2]

What is new in our work:

- **an overlapping clustering solution for the topic extraction**. While the literature presents solutions for topic extraction from a set of documents, they usually create *crisp clusters* [1], [9],

[10]. **LINGO** [16] outputs overlapping groups, but it is rather specialized for *Search Engine result* clustering.

- **the experimentations**. The application was designed in such a manner that would allow us to experiments over several data sets in two languages (French and English) using different term weighting schemes (see Section 5).
- **comparing with other algorithms**. The application design was created bearing in mind an interoperability with other Data Mining platforms (eg. WEKA). This will allow us to use along with our **OKM** [4], implementations of other clustering algorithms, in order to be able to compare performances of both the regrouping phase and that of the cluster naming.
- **an expert based keyphrase evaluation**.We propose an evaluation scheme that, based on grades given by the expert to each extracted keyphrase, calculate a general score of the obtained partition. This way we can compare the performances of different term weighting schemes and clustering algorithms and their influence on the naming process.

Outline of the clustering algorithm. The process of clustering and the one of naming are both crucial operations, but different authors propose different balances between the two. For example in [1], [10] and [15] the accent falls on the keyphrase discovery stage, while in [9] a bigger importance is given to the clustering (regrouping) phase. We have chosen the second one. As result of the discussion presented later in the document, we have chosen to first do the clustering of the documents, after a pretreatment phase, using the algorithm OKM, presented in [4]. OKM stands for Overlapping K-Means and is similar to the well known K-Means algorithm, the major difference being that it allows a document to belong to more than one group. This feature makes it more suitable to use with textual data.

Outline of the thematic extraction. Thematic extraction from Natural Language Texts is a very active research domain, a lot of work being done on it in the last two decades. The results of this work are very well synthesized in [18]. Linguistic, statistical and hybrid solutions exist for the problem. Some of the are completely automated, like **LocalMaxs** [7] and [8], **CorePhrase** [10], **Armil** [9], **XTRACT** [19], while other use expert intervention to validate at each step the extracted collocations, **ESATEC** [2] and **EXIT** [18]. We have chose a Suffix Tree based [15] [16], automated , statistical approach to extract frequent keyphrases from the already partitioned clusters. This way of extracting keyphrases is based on the propriety of *completeness*.

Outline of the name associating phase. Once the clustering of the document is complete and frequent keyphrases are extracted from each group, forming the candidate set, we must now chose one of them to be the name of the cluster. In ARMIL [9], *Information Gain* is used to associate the names. In our work we decided to consider the candidates as pseudo-documents (using the same measure as the one in the clustering process) and calculate the distance (document similarity) between them and the center of the group (the centroid). The one that is most similar to the center is chosen as name. In this manner, candidates comprised only from words that do not bring any information (“analyst said”, “two months”) are filtered out from the rest.

Outline of the paper. The reminder of the document is structured as follows: Section 4 presents the bibliographical study on the existing solutions, both on overlapping clustering algorithms and keyphrase extraction, Section 5 contains the presentation of our approach. In Section 6 the evaluation and practical results of our work are discussed and in Section 7 we present the conclusions of our work and some future perspectives.

4 State of the Art

In the last two decades, the domain of automatic text clustering based on the thematics and thematic extraction has seen a lot of work. Many authors proposed different ways of achieving the goal. In preparation for our thesis we concentrated on three types of work:

- Papers that deal with clustering - we were mainly interested in overlapping clustering algorithms [4], [6], [5], [3] ;
- Work in the keyphrase extraction field [2], [7], [8], [14], [19] ;
- Algorithms that dealt with the clustering part as well as thematic extraction, this work being the closest to the one approached in this thesis [15]. [16], [1], [9], [10].

The authors of [9] observed that the problem is naturally composed of two subproblems: the clustering of the documents and the keyphrases extraction from the corpus. In the light of that remark, our bibliographical research was not directed only toward papers that approach a similar thematic as ours, but also on work performed on both of the subproblems.

4.1 Clustering of the documents

K-Means [12] is one of the most well-known clustering algorithm. Extensive work was done in the subject and it is proven that good result are obtained with it. It consists in constructing a collection of disjointed classes forming a partition of the dataset, by optimizing an objective criteria. Though it was successfully used for textual data, one particularity of **Natural Language Texts** rules it out of our candidates list: in these kind of text, usually, more than one thematic is approached (the same text can talk about more subjects, ex: politics and economics). Therefore, we are looking for ways of obtaining non-disjointed classes - ***Overlapping Clustering***.

Most of the related work [10], [2] performs ***crisp clustering*** on the textual data, therefore we consider that finding overlapping classes would improve the user experience and results quality.

In the reminder of this sub-chapter we present three algorithms that are candidates for our clustering phase. Still, we consider to necessary to make a distinction between algorithms that output an overlapping partition and those whose result are more of a "fuzzy" approach.

In fuzzy clustering, each document has a degree of belonging to all clusters, as in fuzzy logic, rather than belonging completely to just one cluster. Thus, documents on the edge of a cluster, may be in the cluster to a lesser degree than documents in the center of cluster. For each document x, we have a coefficient giving the degree (probability) of being in the kth cluster $u_k(x)$.

Still, fuzzy logic clustering algorithms, can be adapted to output an overlapping partition by choosing a threshold and considering that if $u_k(x) > threshold$ than the document is in the kth cluster.

4.1.1 Singular Value Decomposition

Singular Value Decomposition is the underlying mathematical ground behind Latent Semantic Analysis (described in [15]). Although LSA (Latent Semantic Analysis) can be used as a statistical topic discovery algorithm [18], in LINGO [15] it is used also for the clustering purpose.

The main idea of the algorithm is to decompose the term/document matrix (all the documents translated into Space Vector Model) in a product of three matrices: $A = USV^T$. U and V are orthogonal matrices, containing the left and right singular vector of A, and S a diagonal matrix, having the singular values of A ordered decreasingly. Figure 3 shows the well-known schema of matrix decomposition, with r being

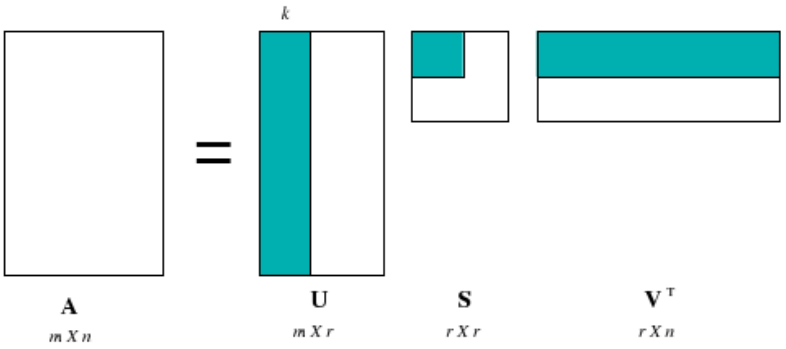


Figure 3: Singular Value Decomposition. A is the original corpus matrix.

the rank of matrix A [18]

If we keep only the k highest ranking singular values and eliminate the rest, along with the corresponding columns in U and lines in V, the product $A_k = USV^T$ is also know as the k-approximation of A.

It is well-known that most clustering algorithms take the number of clusters as a parameters, which is arbitrarily set by expert. The SVD approach allows an automatic determination of the number of clusters, based on the value of singular values of the original matrix. In LINGO, the Frobenius norm of the A and A_k matrices is used to calculate the percentage distance between the original term / document matrix and its approximation. This way an aproximation for k can be found.

Once the number of classes has been chosen, the corresponding columns in U create an orthogonal basis for the document space. According to mathematical vectorial space theory, every component of the space, in our case every document, can be expressed as a weighted sum of the elements of the base.

$$d_i = \alpha_1 e_1 + \alpha_2 e_2 + \ ... \ + \alpha_k e_k$$

The elements $e_l, l \in \{1..k\}$ of the base can be considered as the centers of the classes and the formula above resembles a fuzzy approach, the document d_i having the probability α_j of belonging to the j^{th} cluster.

4.1.2 Latent Dirichlet allocation

Latent Dirichlet Allocation [3] is a constructive model. It considers documents as collections of words and models each word in a document as a sample from a mixture model, where the mixture components are representations of "topics". Thus each word is generated from a single topic, and different words in a document may be generated from different topics. Each document is represented as a list of mixing proportions for these mixture components and thereby reduced to a probability distribution on a fixed set of topics.

LDA is similar to **probabilistic latent semantic analysis** (pLSA), except that in LDA the topic distribution is assumed to have a *Dirichlet* prior. In practice, this results in more reasonable mixtures of topics in a document.

An LDA model starts with a set of topics. Each of this topics has probabilities of generating various words. Words without special relevance, like articles and prepositions, will have roughly even probability between classes (or can be placed into a separate category). A document is generated by picking a distribution over topics and given this distribution, picking the topic of each specific word. Then words are generated given their topics.

Once the distribution has been established for each document in the collection, the ones that share one common topic can be placed in the same group. As each document is a mixture of different topics, in the clustering process, the same document can be place in more that one group, though resulting in a **Overlapping Clustering Process**.

Viewed from **our work's point of view**, this method presents some disadvantages:

- It is a complex mathematical model, that consideres each document a mixture of many topics. While this could be good for explaining the documents, in the case of clustering, it would mean that each document belongs, in a certain percentage, to many clusters (fuzzy approach). A suitable method should be devised that could pick only part of the clusters for each document;
- This method does not present a center for each cluster, but a distribution of the document over the topics. This is not very readable for the user and makes associating a name to a cluster harder;
- Its complex mathematical model makes implementation difficult and error prone. Also, if used in the above manner, the clustering process would not make use of all the informations that it provides (mainly the probabilistic distribution of the topics and the fact that it can identify words in the document according to topics).

Due to this observations, the authors of this paper considered other solutions for the clustering phase.

4.1.3 OKM (Overlapping K-Means)

As the name suggests, **OKM** [4] is an extension of the well-known **K-Means**. It shares the general outline of the algorithm, trying to minimize an objective function. It does so by initially randomly choosing k centroids (centers) from the data set, and then iterating these two steps:

1. Assigning the documents to the clusters;
2. Recalculating the clusters, based on the new configuration;

until the objective value reaches a local minimum.

The main difference in the OKM algorithm compared to K-Means is that a document can be assigned to multiple clusters. If in K-Means each document was assigned to the centroid that was closest to him, in terms of cosine distance (detailed in subsection 5.1), OKM calculates an image of the centroids, adding the document to clusters so that the distance between it and its image is minimal. This image is the *Gravity Center* of the assigned centroids.

Therefore, the function that OKM tries to minimize is the distortion in the dataset:

$$distorsion\,(\Pi) = \frac{1}{NK}\sum_{i=1}^{N} d\left(X^{(i)}, Z^{(i)}\right)^2$$

where $Z_{(i)}$ represents its image.

While OKM inherits from K-Means its powerful dependence on the initialization - the algorithm goes towards a local maximum - and the number of clusters must be arbitrary specified by the expert, its linear execution time, its performances [6] and simplicity make it a perfect candidate for our clustering phase.

In [5] is presented wOKM, a weighted version of OKM, that uses weights internally and achieves even better performances in terms of **precision**, **recall** and **FScore**. Updating the clustering algorithm to wOKM is a future perspective of this work.

4.2 Extracting the keyphrases

From the clustering techniques, the hierarchical clustering is ideal for cluster content presentation tools, mainly interactive visualization and browsing [17]. But often, users need more than that, the need a description of the groups, so that they can see from one look what it is about and decide if it is important or not. That is what emerged the need of adding meaningful descriptions to the clusters.

A meaningful, human readable name is a complete phrase, that contains all the words that have a special meaning together (like "data mining") and all the prepositions and articles that make sens to the human reader ("of" in "Ministry of Internal Affairs"). A **keyphrase** is "a sequence of one or more words that is considered highly relevant as a whole", while a **keyword** is "a single word that is highly relevant" [10].

In our search for ways of extracting such a name, we directed our research towards topic extraction algorithms, because most of them concentrate on presenting the used with a coherent, readable expression.

In [10], keyphrase extraction algorithms are divided into two categories:

- Those that **construct** the keyphrases from a single document, which is usually a supervised learning task, often regarded as a more intelligent way of summarizing the text, but more difficult in implementing. Examples: *ESATEC* [2], *EXIT* [18], *XTRACT* [19]
- Those that **extract** them from a set of documents, which is an unsupervised techniques, trying to discover the topics, rather that learn from examples. While this type of algorithms can not find keyphrases that are not in the texts of the documents, they can be used in conjuncture with the clustering algorithm in order to improve their performances. Examples: *CorePhrase* [10], *Armil* [9], *SuffixTree Extraction* [15].

In [18], topic extraction algorithms are divided into 3 categories, based on their approach: linguistic, numeric and hybrid.

4.2.1 Linguistic approaches

In [18], 3 linguistic systems are presented: TERMINO, LEXTER and INTEX & FASTR.

All these systems make use of morphological and syntactic informations about the words in the texts. The POS tagger (Part-Of-Speech) tries to recognize whether the word is a noun, adjective, verb or adverb, and tries to characterize it morphologically (number, person, mode, time etc). Based on this information, the lematisation process extract the radix of the word (masculine single - for nouns, infinitive - for verbs).

With the texts tagged, each system has each own approach toward discovering the keyprases. In TERMINO, a lexical-syntactic analyzer is used to describe the sentences, and then certain patterns are used to uncover the keyphrases (ex: <Head> <Prepositional Group> <Adjectival Group>). LEXTER uses the morphological information to extract from the text nominal groups and then searches for dis-ambiguous maximal nominal groups.

4.2.2 Numerical approaches

These algorithms make use only of numerical (statistical) informations in order to discover the topics. For each couple of words in the text, the Mutual Information is calculated. This allows to measure the dependence between the two words in the binary collocation, also called bigram. The Mutual Information is given the formula: [18]

$$IM(x,y) = \frac{P(x,y)}{P(x)P(y)}$$

where $P(x)$ and $P(y)$ are the probabilities that the word x and, respectively, y appear in the text, while $P(x, y)$ represents the probability of the words x and y appearing together as neighbors. This allow us to calculate the dependence between 2 words that are one next to the other or in a window of specified dimensions. In [1] a windows of dimension 11 is considered around a word (5 words before + word + 5 words after).

Once we have the tool for extracting bigrams from the text, some authors (**EXIT** [18], **ESATEC** [2]) propose ways of constructing ngrams, by combining iteratively the bigrams or adding the an existing (n-1)gram another word, trying to obtain longer collocations that have a high *Mutual Information* score.

There were many statistical measure proposed to calculate the strength of the relationship between 2 words. In [1] the algorithms first finds a set of terms that are frequent (over a minimum threshold). Than a set of pair of these terms is created, retaining only the ones that score a minimum frequency. Only for these pairs, the β-similarity is calculated and the set of documents for which the pair is representative is constructed. This algorithm uses the topic extraction phase for the clustering phase.

In [7], [8] , the authors consider that a special **"glue"** exists between words that make them have a sens together. LocalMaxs is used in conjuncture with the **Symmetric Conditional Probability** (SCP) measure to extract **Continuous Multiple Word Units** and with **Mutual Expectation** (ME) measure for extracting **Non-Continuous Multiple Word Units**.

[14] starts from the idea that all ngrams can be constructed from bigrams and it is, therefore, essential to better understanding the impact of the measure used on the algorithm's performance. Experiments are performed on some of the most known and used measure's performances , judging by their ability to identify lexically associated bigrams. The measures compared are: **t-score**, **Pearson's χ-square test**, **log-likelihood ratio**, **pointwise mutual information** and **mutual dependency**.

There are other approaches that do not rely on bigram detection and ngram construction. In **CorePhrase** [10] keyphrases are considered to naturally lie at the intersection of the document cluster. The CorePhrase algorithm compares every pair of documents to extract matching phrases. It employs a document phrase indexing graph structure, known as the **Document Index Graph** (DIG). It keep a cumulative graph representing currently processed documents. Upon introducing a new document, its subgraph is matched with the existing cumulative graph to extract the matching phrases between the new document and all previous documents. The graph maintains complete phrase structure identifying the containing document and phrase location, so cycles can be uniquely identified. Simultaneously, it calculates some predefined phrase featured that are used for later ranking.

In **LINGO** [15], [16], a **Suffix Tree** based keyphrase discovery is used. The algorithm tries to avoid extracting incomplete phrases (like "President Nicolas" instead of "President Nicolas Sarkozy") which are often meaningless, it uses the notion of *phrase completeness*.

A phrase is complete if and only if all its components appear together in all occurrences of the phrase. For example, is the phrase "President Nicolas" is followed in all occurrences by the term "Sarkozy", than is it not a complete phrase. Starting from this definition, right and left completeness can be defined (the example above is left complete, but not right complete). Using a Suffix Tree data structure, the complete phrases can be detected and the ones that occur a minimum number of times (frequent keyphrases) create the candidate set for the topics. A more detailed explanation of this approach is presented in Chapter 5.

4.2.3 Hybrid approaches

An hybrid system is usually adding linguistic information to an essentially numerical system or adding numeric (statistical) information to a essentially linguistic system. This process usually ameliorates the results.

It is well-known that statistical systems (like those based on Bayesian networks) produce noisy results in the field of Information Retrieval [2], meaning that among the extracted candidates, most of them pass the frequency threshold and get good scores, but they are uninteresting from the topics point of view. Such expressions can be comprised of common words (articles, prepositions, certain verbs, etc) like "he responded that" or "the biggest part of the", and they bring no new information. Such phrases should be eliminated. For that, linguistic filters are very useful.

When revising the linguistic methods, we have seen that some of them rely of certain keyphrase formats (like <Subject> <Verb> or <Verb> <Adverb>) to construct the result. A morphological and syntactic tagger could be used as a final phase to filter out the noise from the candidates set resulted from statistical extraction.

From such a filter benefits the system **XTRACT** [19], [18] which is comprised of 3 phases. In the first phase, bigrams are extracted from a grammatically tagger corpus, using an eleven words window. The next phase consists in extracting longer phrases if they are frequent in the text. These phrases are called *rigid noun phrases*.

The third phase is the linguistic phase. It consists in associating a syntactic etiquette to the extracted bigrams (<Noun> <Verb>, <Adjective> <Noun>), and for each bigram associate longer phrases containing the ngrams obtained at the second phase.

4.3 Approaches that deal both with clustering and topic extraction

Some of the systems presented so far deal both with clustering textual data and extracting topics. While both phases are of crucial importance, different authors treat them differently. Some consider the thematic extraction more important and start with keyprase discovery and from these candidates perform the actual data clustering (seen in [1], [10], [15], [16]), while others start with the clustering process and from the resulted clusters proceed with extracting the keyphrases [9].

In LINGO [15], [16], the algorithm starts by extracting the complete phrases and then uses a Singular Value Decomposition process to identify the basis of the vectorial space (which can be considered as centers of the groups). Using this vectors, the candidates for group names are chosen and finally documents are assigned to the groups, based on their similarity to the chosen names.

A somewhat similar approach is presented in [1]. Here, bigrams are extracted from the text, with a 5 term window size. After calculating the β-similarity, the set of documents for which the bigram is representative is constructed. The downside of this method is that only bigrams (two words collocations) are extracted.

CorePhrase [10] uses a phrase-based indexing model to describe the documents. Keyphrases are obtained by intersecting documents and a graph of processed documents is maintained. Also, phrase features are calculated for later ranking.

Armil [9] starts by clustering the documents using a M-FPF (Modified Furthest-Point-First) algorithm, which resembles K-Means method, but in which no centroids are calculated. Centroids tend to be dense vectors and working with them can be computationally costly. M-FPF avoids that by calculating the intra-cluster variance using the distance between each pair of documents, From the extracted clusters, thematics are extracted, using the Information Gain, taking into consideration both the intra and the inter-cluster variance.

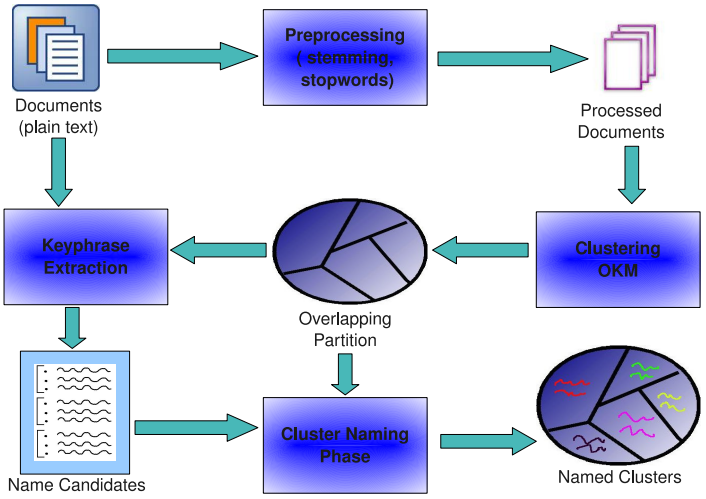


Figure 4: Streamlined schema of our algorithm.

5 Our approach

While keyphrase extraction based on linguistic approaches do succeed in obtaining less noisy output, they are also vulnerable to multilingual corpora and neologisms. They also have the tendency of being adapted to stereotypical texts (texts from a specified narrow field) [2].

In [18] another two reasons that advantage the numerical approaches are presented:

- Although they have the tendency of omitting rare terms (mainly those that appear in the corpus only once), experience has shown that more often the rare terms are in fact incorrect terms. So not taking into account phrases that contain these terms actually improves the overall quality of the extracted keyphrases. This fact privileges the precision in the detriment of recall.
- The use of linguistic methods lead to almost exponential explosion of the numbers of collocations extracted when the size of the corpus increases. That is why usage of method based only on linguistic information could prove prohibitive.

To this argumentation, we dare to add another reason. Complex approaches, based on constructive methods and/or linguistic information are usually designed to extract the thematics from one document. They try to reject meaningless phrases (those that bring no new information) based on special phrase layout. But they do not make use of the result of the clustering process.

We believe that once the clustering has been performed and the centers of the classes obtained, we could use this information to filter out the noise from the keyphrase candidate set. While centroids (center of classes) are the essence of the documents in those classes, choosing the candidates that are closest to them naturally eliminates phrases that are too general.

For example: in a document group that talks mainly about politics, the most important terms (measured with a *term weighting scheme*) should naturally be "parliament", "govern", "president", "party", "politics" etc. When calculating the similarity (cosine similarity) between this centroid and phrase candidates, is is natural that a candidate that contains as many of those words would be favored. "Presidential elections" phrase would clearly scored higher than "as a matter of fact the" phrase.

In the light of this observation, we chosen to give the clustering process a higher importance and perform it first. Once the clusters are created, starting from each one of them, through statistical methods, we create a candidate set of frequent keyphrases. And finally, by comparison with the center of the class, we

chose the name, as the highest rated.

For the clustering phase, we have chosen **OKM (Overlapping K-Means)** [4] algorithm because it was specially designed for overlapping clustering (one of our conditions) and because its proven effectiveness when working with textual data. As for the topic extraction phase, from the statistical methods presented, we chose the **Suffix Tree based** [15] approach, for its ability to extract the phrases from raw (untreated text), language independence, linear execution time and the power to extract humanly readable phrases.

5.1 Vector Space Model

Various models have been proposed for modeling the information in Information Retrieval systems: *Boolean Model* compares True / False query statement with the word set that describes a document, *Probabilistic Model* calculates the relevance probabilities for the documents in the set. But the model that is most widely used in modern clustering algorithms is *Vector Space Model*.

In this model, each document is represented as multidimensional vector. Each dimension is a keyword or a term and its value associated to a document is directly proportional on the degree of relationship between them. There are 4 major ways of measuring this relationship degree, also known as **term weighting schemes**.

1. **Presence / Absence.** It is also known as **binary weighting** [16] and it is the simplest way to measure the appartenance of a word to a document. Its mathematical formula is:

$$a_{i,j} = \begin{cases} 1 & if\, term\, i\, is\, found\, in\, document\, j; \\ 0 & otherwise. \end{cases}$$

   In [15] is shown that this scheme can only show if a word is related to a document, but does not measure the *strength* of the relationship.

2. **Term Frequency.** Or **term count**. It is the number of times a given term appears in a document. While this is a better measure of the relationship between the term (word) and the document, this scheme has the tendency of favoring longer documents. In order to prevent that, normalization is usually used.

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

   where $n_{i,j}$ is the number of occurrences of the considered term (t_i) in document d_j, and the denominator is the sum of number of occurrences of all terms in document d_j.

3. **Inverse Document Frequency.** Is a measure of the general importance of a term in the whole corpus. It expresses the idea that a word should be less important if it appears in many documents. In this way very common words, as prepositions, articles and certain verbs and adjectives could be filtered out or, at least, given less importance.

$$IDF_i = \log \frac{|D|}{|\{d \,|\, t_i \in d\}|}$$

   where $|D|$ is the total number of documents in the collection and $|\{d \,|\, t_i \in d\}|$ is the total number of documents where the term t_i appers. Special attention should be given to situations where the word doen't appear in any documents (division by zero). The logarithm is used in order to reduce the influence of IDF in TfxIDF scheme (next), so that a balance is achieved.

4. **TFxIDF.** Is the most used scheme in Information Retrieval. It is the product of **Term Frequency** and **Inverse Document Frequency.**

$$TFxIDF_{i,j} = TF_{i,j} * IDF_i$$

This scheme aims at balancing local and global occurrences. A high weight in TFxIDF is reached by a high term frequency (in the given document) and a low frequency of the term in the whole collection of documents. This weighting scheme hence tends to filter out common terms.

Once that documents were translated into **Space Vector Model**, using one of the schemes presented above, the similarity between two documents is usually calculated using the **cosine distance**.

$$similarity(a,b) = cos(\vec{a},\vec{b}) = \frac{\sum_{i=1}^{t} a_{i,j} b_{i,j}}{\sqrt{\sum_{i=1}^{t} a_{i,j}} \sqrt{\sum_{i=1}^{t} b_{i,j}}} \quad (1)$$

which can be interpreted as the geometrical angle between the vectors in the multidimensional space.

5.2 Pretreatement

Pretreatment is an important part of the algorithm. It is comprised of two elements: *stemming* and *stopwords removal*.

Stemming is the process through which inflection, prefixes and suffixes are remove from each term in the collection. It is extremely useful especially for languages that are are heavily inflected (like the verbs in French) and reduces words to their stems. This guarantees that all inflected forms of a term are treated as one single term, which increases their descriptive power. At the same time, bare stems may be more difficult for the users to understand, but since the stemmed version of the terms are never presented to the user, it will not hinder their usage. For English, a free Java version of Porter's stemmer was used.

Stopwords (articles, prepositions etc) do not present any descriptive value, so they are of no use for the clustering process. Even more, they only make the corpus dictionary bigger, so that computation is slower. Some term weighting schemes (such as Term Frequency) are especially vulnerable to stopwords, so there elimination is compulsory. It is done using stopword lists for each language (English, French).

As the topic discovery phase requires the texts in their natural form (stemmed words are hard to read and stopwords improve the overall quality of the cluster names), an untreated version of the documents is also kept to be used for that phase.

Pretreatment is the only part of our algorithm that is language dependent. The modular design of the application (subsection 5.6) makes adding languages easy, by adding new stemming algorithms and stopwords lists.

5.3 Clustering

Clustering the text documents is done using the **OKM algorithm** (see subsection 4.1.4).

The documents are translated into the **Space Vector Model** using one the the four measures presented in subsection 5.1. Following the idea that a compound measure could be created as the product of other measures [13], the algorithm was created in such a way that any of the term weighting schemes presented in this paper could be used. In this manner, we could experiment and compare the performances obtained by different schemes (see Chapter 6).

Our implementation of the **OKM algorithm** respects the original indications [4]. The only change that we made was the stopping condition. In the original form, the iteration come to an end when the partition composition do not change - which means that a local minimum has been reached. While from the clustering's point of view, the final result has been found, it does not necessarily mean that centroids do not evaluate over the next iterations.

In **K-Means** [12], the centroid is computed depending only on cluster's composition. Meaning that if the clusters do not change between 2 iteration, neither do the centroids. In **OKM** the centroid update process is a little more complicated, each of the centroid being dependent on both the documents in their own group, but also on the other centroids resulted from the last iteration. The update formula for the centroids is:

$$c_{j,v} = \frac{1}{\sum_{x_i \in R_j} \frac{1}{\delta_i^2}} x \sum_{x_i \in R_j} \frac{1}{\delta_i^2} . \hat{x}_{i_v}^j \tag{2}$$

where $\hat{x}_{i_v}^j$ in formula 2 has the following expression $\hat{x}_{i_v}^j = \delta_i x_{i_v} - (\delta_i - 1)\bar{x}_{i,v}^{A \setminus \{c_j\}}$

- A is the set of centroids to which document x_i is assigned;
- $\bar{x}_{i,v}^{A \setminus \{c_j\}}$ is vth component of the center of gravity of the centroids to which x_i is assigned, except centroid j
- $c_{j,v}$ is the vth component of the centroid to be updated.

This dependency means that centroids continue to change even if the classes do not. For us, it is very important to have exact centroids - the process of name assignment is dependent on the centroid quality. That is why we have chosen not to stop when the clusters stop changing, but rather use a threshold for the variance of the objective function between iterations ($\epsilon < 0,001$).

5.4 Keyphrase Extraction. Name candidates

In [15], there are four conditions specified that must be met by a collocation (or term) in order to be considered a name candidate:

- appear in the text with a specified frequency. This is based on the assumption that the keyphrases that occur often in the text have the strongest descriptive power. Also, isolated appearances have high chances of being incorrect words [18].
- they do not cross sentence boundary. Usually, meaningful keyphrases are contained into a sentence, because sentences represent markers of topical shift.
- be a complete phrase. Complete phrases make more sense than incomplete ones ("President Nicolas" vs "President Nicolas Sarkozy").
- not begin or end with a stopword. Cluster name candidates will be stripped of leading or trailing stopwords, as that is likely to increase readability. Stopwords in the middle of the keyphrase will be regarded.

We have chosen to use a Suffix Tree based approach to extract name candidates. It makes use of the propriety of completeness (see subsection 4.2.2). The heyphrase discovery algorithm works in two phases: in the first one left and right complete expressions are found and in the second one the two sets are intersected to obtain the set of complete expressions.

5.4.1 Suffix Tree Construction

The algorithm of discovering right complete expressions relies on the usage of Suffix Tree. A Suffix Tree is an alphabetically ordered array of all suffixes of a string. We note here that in our case, the fundamental unit is not the letter (as in the case of classical strings), but the term / word. For example, having the phrase "we are having a reunion", the suffix tree for it would be:

No	Suffix	Start Pos
1	a reunion	4
2	are having a reunion	2
3	having a reunion	3
4	reunion	5
5	we are having a reunion	1

One of the most important problems in the construction of the Suffix Tree is the space-time and time-efficient sorting of the suffixes. In [11], two approaches are presented: "Manber and Myers" and "Sadakane's algorithm". As the paper presents also a comparison of the two, from both the theoretical and practical performance point of view, we have chosen to use the second approach, which gives better results (in terms of efficiency).

The only thing required for the algorithm is that the terms have a lexicographic order, so they can be compared. If in the example above, for the sakes of clarity, we have used the alphabetical order, in real-case implementation, the criteria used is not important. The order of term arrival into the collection can also be used.

The "Sadakane's sorting algorithm" is a modified bucket sorting, which takes into consideration the unequal dimensions of the suffixes. In [11], it is shown that the sorting complexity is $O(n \log n)$, with n the number of suffixes.

Because a keyphrase can not pass the boundary of a sentence, we have modified the algorithm proposed in [15] by constructing the Suffix Tree on a sentence based approach, rather than the whole document approach that was suggested in the paper. Therefore a suffix identification is given not only by the beginning of the suffix, but also on the index of the sentence.

5.4.2 Complete Phrase Discovery

The general idea behind the right complete keyphrase discovery algorithm is to linearly scan the suffix array in search of frequent prefixes, counting their occurrences meanwhile. Once such a prefix is identified, information about its position and frequency (initially the frequency is 2) is stored along with it.

Once the right complete phrases have been discovered, we also need to discover the left complete phrases. This can be achieved by applying the same algorithm as before to the inverse of the document - meaning that we create another version of the document, having the words in reverse order. While the algorithm finds the right complete phrases in lexicographic order, the left complete set still need work. Because they were extracted from the inverted version of the documents, the phrases are in inverse order, so they need to be inverted once again and sorted.

With both sets in lexicographic order, we can intersect the two in linear time. Name candidates are returned along with their frequency. We must note here that the extracted candidates can also be single terms, as sometimes a single word can be enough for explaining the content of the cluster [15].

The last phase is filtering the candidate set. First, only phrases that have a minimum frequency are kept, the rest being eliminated. In [15], the value of this threshold is suggested to be between 2 and 5. The relatively low value for it can be explained by the fact that the most frequent expressions are not necessarily the most expressive, but usually they are meaningless expressions - noise in the output. We referred to them in subsection 4.2.3. Throughout our experimentations we used a threshold equal to 3.

The last filtering of the candidates that we do is based on one of the conditions that we enumerated at the beginning of this subsection: *not to begin or to end with a stopword*. Using the same pretreatment module as for the clustering, we recursively eliminated leading and trailing **stopwords** from the phrases.

As a result some of the candidates disappeared completely (they were composed only from stopwords), while others reduced their form to another one (example: "the president" and "president of" become both "president"). After pruning the empty and identical candidates from the set, our experimentations show a decrease of the number of phrases by aproximately one third.

5.5 Associating names to clusters

As mentioned at the beginning of this section, we propose to use the results of the clustering process to select the name of each cluster. Our approach consists in two phases:

- **candidates extraction from the documents in the cluster**. We feed into the keyphrase discovery algorithm only the documents that are part of the cluster that want to name. That way only frequent collocations and terms from those documents will appear as candidates, though speeding up the calculations.
- **phrase scoring based on the similarity with the centroid** - cluster center.

As the centroid is the "essence" of the cluster - with the most relevant terms having high weights, we believe that, by use of the same measure as in the clustering process, we can properly chose a name for the group. Basically, we take all the name candidates candidates and reintroduce them into the document collection as "pseudo-documents". After applying the same pretreatment as to original documents (because the keyphrases were extracted from natural language texts, they contain inflected words and stopwords), we translate them into the Space Vector Model, using the same term weighting scheme as for the other documents.

The last step is to calculate the similarity between each of these "pseudo-documents" and the centroid of the class. The one that scores highest is considered to be the cluster name.

Our experimentations showed that this approach gives good results, but the fact that only the distance between the candidate and the cluster's center is taken into consideration can result in similar names for different clusters. As a future perspective, we plan to enhance the name selection scoring scheme by also taking into account the other centroids. This way, we would find not only the keyphrase that is closest to the center, but also the most distant to the other centers.

5.6 Application Design

Throughout the application design process, we had in mind two things: *modularity* and *interoperability*.

As figure 4 presents, our application has 4 major modules. The first three are separated one from another: Preprocessing (stemming and stopword elimination), Clustering algorithm and Keyphrase Extractor. Each of them was developed independently and only after separate testing they were merged into the main application as modules. The forth part - The Cluster Naming Phase - was integrated into the main program, as it is dependent on all the three previous modules (see Subsection 5.5).

By modular design two objectives were able to be fulfilled:

- **reduced dependency on the language.** As the pretreatment is the only language dependent phase of the program, it was given special attention. It was designed so that stemming algorithms for different languages could be easily added and switched between them. Stopword elimination is done using stopwords lists (which are external to the program). Adding a new language is as easy as adding a new file and making the module aware of it.
- **possibility of comparing different algorithms.** Implementations of other clustering algorithms could replace our OKM module, while keeping the same keyphrase extraction algorithm, which allows us to compare the performances of different clustering approaches in the field of topic extraction. Same goes for the keyphrase extraction algorithm.

We developed the application in the Java programming language, so that it could be compatible with other Data Mining platforms (ex. WEKA). As WEKA contains implementations of several clustering algorithms (Fuzzy KMeans, EM), this could prove to be an advantage.

6 Evaluation

In this section we will present our attempts to evaluate the result of our algorithm. We will briefly summarize popular evaluation approaches that are used in similar work throughout the literature and some of the results we obtained.

Our evaluation comports two aspects: the evaluating of the **clustering algorithm** and that of the **clustering naming process**. For each we have chosen different means of scoring the results.

[15] points out three major approaches for evaluating the resulted partition:

- **Standard IR metrics: precision and recall.** This evaluation method uses an expert evaluated set of documents, performs the clustering process and then compares the results of the algorithm with those of the experts. Statistical measure can be computed from these data and, thus, allowing us to compare algorithms between them.
- **Merge-then-cluster approach.** This is an usual evaluation method which relies on running the clustering algorithm on the specially manufactured data set. This data set is the result of the merging of several smaller data sets, in which all the documents share the same topic. The quality of the clustering algorithm is then judged by the percentage with which the original data sets are reconstructed into topics. We consider this method inappropriate because our clustering algorithm perform an overlapping clustering, even if the original sets were disjointed. That means that this evaluation means seriously penalizes our algorithm compared to crisp algorithms. Also, we consider that running our algorithm on crisp data sets would also be unrealistic: real life textual data rarely has only one topic.
- **User evaluation.** In this approach, a considerable(so that subjectivity could be eliminated through the means of average responses) number of subjects are asked to manually rate the results of the algorithm. While this method has the disadvantage of being subjective and also slow and not automatized, it is also the closest to the human user. The real human feeling towards the results can be measured. We stress out that this method is also very general and could be used for any kind of *Information Retrieval* results.

We have chosen to use the "standard IR metric" approach for evaluation the clustering phase and the user evaluation for the name extraction phase.

6.1 Clustering evaluation

For evaluation the clustering process, we have used a standard IR approach, using the precision and recall. Following the experimentation in [4], we have used a data set of 262 documents, a sub-partition of the Reuters corpus. Reuters is a collection of articles in the English language, some of them have one or more topics associated. The documents we selected had at least one topic.

Also, [4] proposes the following evaluation: starting from this data set we execute our algorithm. We call two documents as being "associated" if they both belong to one of the classes found in the clustering process. Further more, we say that this association is correct only if the two documents share an common Reuters label/topic. The partition is then evaluated based on the total number of associations (noted n_a), the number of correct ones (noted n_b) and the total number of expected associations (noted n_c). With this indicators, the precision, recall and F_{score} can be calculated:

$$precision = \frac{n_b}{n_a};\ \ recall = \frac{n_b}{n_c};\ \ F_{score} = \frac{(\beta^2 + 1) * precision * recall}{\beta^2 * precision + recall}$$

Using the value $\beta = 1$, we conducted our experiments on the specially prepared data set, varying the number of classes from 5 to 30, for each value 10 times and taking the highest score. We did this experimentation also for the classical K-Means algorithm, so that the F_{score} could be compared.

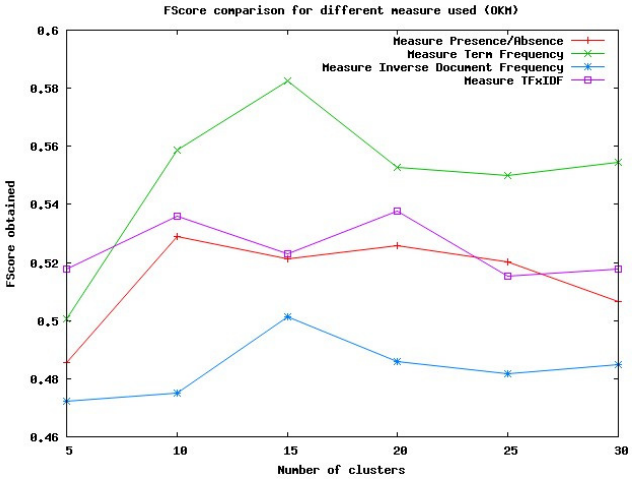


Figure 5: F_{score} obtained for **OKM**, for different term weighting schemes.

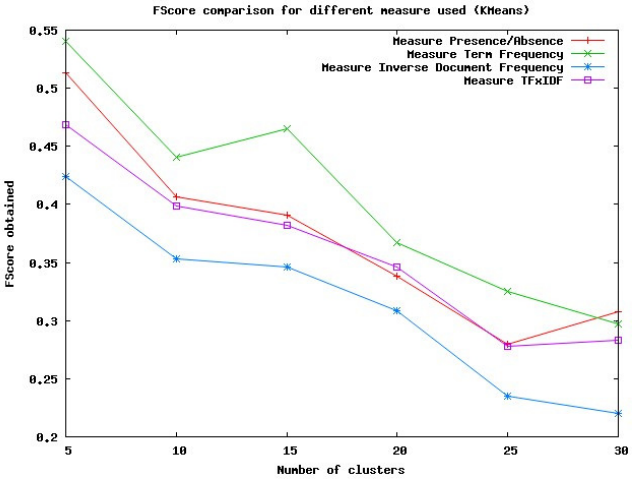


Figure 6: F_{score} obtained for **KMeans**, for different term weighting schemes.

What we bring new to this experimentations is also the variation of the term weighting scheme. In this way we could experimentally evaluate the performances of each term weighting scheme. The results for the comparison over the different schemes with the OKM algorithm can be found in figure 5 and the ones with KMeans in figure 6. The side by side comparison of the precision and recall over the different term weighting schemes for the two algorithms can be found in appendix A. It is very interesting to observe that in both cases the measure that performed best was the "Term Frequency" measure. That is why we chose this measure to create the graphic that compares the performances of OKM and KMeans.

We observe that all three figures 5, 6 and 7 show a for almost all graphs a local maximum for the number of clusters $k = 15$. This suggest, in fact, that 15 is the optimum number of clusters for that specific data set.

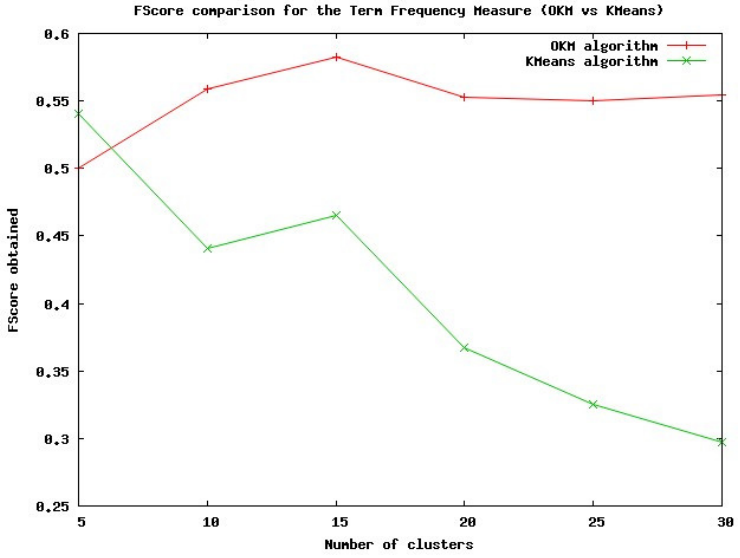


Figure 7: F_{score} comparison between OKM and KMeans.

Figure 7 confirms the initial hypothesis that an overlapping approach is more suitable to Natural

Language Textual Data. While the quality of the crisp clustering decreases as the number of groups increases, the overlapping approach maintains a steady score. The decrease of the F_{score} observed in the KMeans' case is because of the reduction of the number of associations (and the recall) with the augmentation of the number of classes. If in the case of KMeans this decrease cannot be compensated by the increase in precision, in the case of OKM intersection of the classes attenuates the decrese of recall. [4] [5]

6.2 Name Extraction evaluation

To evaluate the Name Extraction phase results, we chose an expert based approach. Basically, this means users will evaluate manually each result and score it. Using the scores from them, statistical calculations can be done in order to extract interesting data.

The reason why we chose this approach is that we are trying to evaluate "human tastes". Because each human is different from the other, a name that one could find meaningful for a certain group of text, another could find that it does not synthesize well the content. And while the Information Retrieval literature does not provide with a widely accepted mean of automatically evaluate the results, we came to the conclusion that this is the best approach for the moment.

We conducted our evaluation on two different data sets: one in English (the same corpus used in the cluster evaluation) and one in French - the content of a forum: "Commémoration". Due to the limited time that we had at our disposal for doing the testing, only three test subjects were involved in the process. For this reason we cannot make any judgment over the overall quality of the naming process, but still some interesting conclusion can be drawn from the comparison of different term weighting schemes over the corpus.

For each data set, we executed the algorithm varying the number of clusters from 5 to 25 (in two of the experimentations, and 8 to 12 in the third one), using each measure and, for each combination of parameters, 5 times. The results were saved into a file and distributed to our participants. They rated each cluster name on the following scale:

- **0** (zero) - the cluster name is totally meaningless, it brings no information on the content of the documents in it;
- **1** (one) - the name is somewhat interesting;
- **2** (two) - the name synthesizes well the content of the cluster and brings useful information.

Once the names were graded, the data was statistically processed - the named partition was assigned a grade between 0 (worst possible) and 1(perfect naming) and an average was calculated for all executions with identical parameters - in order to obtain the percentage of good quality, medium quality and bad quality names for each combination of parameters, for each measure and globally.

In Figures 8 and 9 are shown the results obtained on the Reuters data set. While Figure 8 presents the quality of the naming process for different measures used, in figure 9, the score (calculated as in the previous paragraph) is shown over different cluster numbers.

There are 2 observations that can be made from this experimentation:

- in all experimentations (the graphics of the other 2 are presented in Appendix B), highest percentage of good quality names were obtained by the schemes "Presence/Absence" and "Term Frequency". This is an experimental confirmation of a theoretical fact: the IDF measure (also present in TFx-IDF), tents to penalize the words that appear in many documents. While this might be useful in the case of clustering, it is rather bothering in the case of cluster naming, as the chosen keyphrase is supposed to be representative for all the documents in the cluster, though the terms composing it tend to appear often in the text.

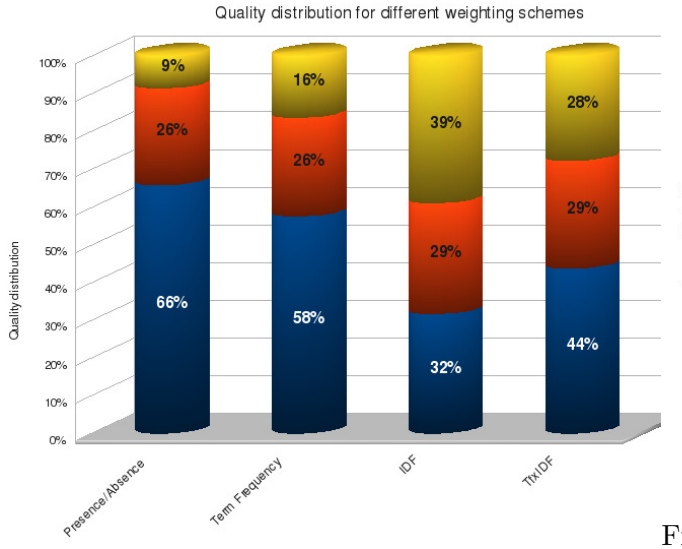


Figure 8: Name Quality: Different Term Weighting Schemes used (Reuters)

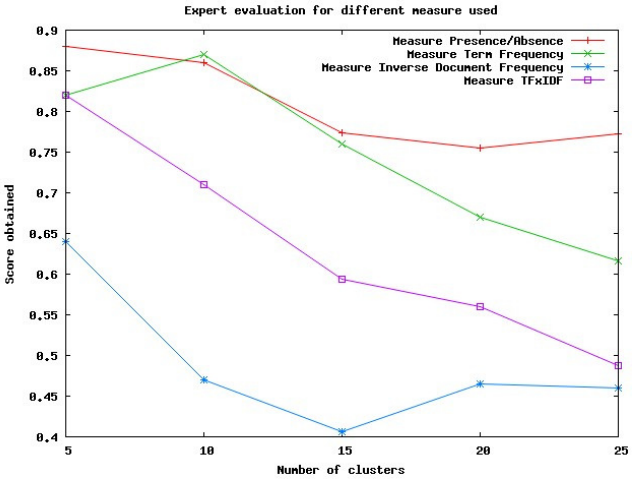


Figure 9: Name Quality: Different Term Weighting Schemes and Cluster Number Variation (Reuters)

- the quality of the names decreases with the increase of the number of clusters. This comes from the fact that name candidates are phrases that appear in the text of the documents in the cluster with a minimum frequency. As cluster number increases, cluster size decreases, and so does the candidate list size. Basically, there are fewer candidates to chose from.

6.3 Sample results

We present here an example of the results that algorithm outputs. The execution was performed over the data set "Commémoration", in the French language. The parameters used:

- number of clusters $k = 5$;
- measure used - "Term Frequency";
- minimum frequency for phrases to become candidates - 3;
- threshold for stopping the clustering algorithm - minimum variation between iterations - $\epsilon < 0.001$;

Centroid No	Highest Rated Words	Cluster Name
1	"francai" "question" "franc" "guer" "histoi" "oubli" "memoi" "sujet" "faut" "fair"	français
2	"comemo" "jour" "fer" "suprim" "histoi" "journ" "acord" "novemb" "bon" "juilet"	jours de commémoration
3	"ariv" "hi" "gose" "aniversai" "comemoron" "divorc" "ajouton" "afirmatif" "souvient" "rous"	anniversaire
4	"travail" "gagn" "suprim" "polit" "dimanch" "francais" "just" "question" "grand" "pinard"	travailler plus pour gagner
5	"jour" "fete" "americain" "franc" "fer" "day" "jambon" "recomencait" "emul" "moutonism"	jours fériés

7 Conclusions and Perspectives

Being faced with increasing quantities of information, must of which has no predetermined structure, we consider that devising means of effective textual clustering and topic extraction is of great interest. Faced with a huge number of documents, the user needs to have a way of dividing them by their thematic. And to quickly judge if a group presents interest do him, he need to be presented with a label of that group, a name, an expression that he can be easily read.

While the last two decades has seen considerable work in this field, most proposed approaches leave out an essential characteristic of the Natural Language Texts: the fact that they can approach multiple subjects. Therefore, dividing documents into crisp partitions, reduces the overall quality of the clustering process.

Starting with this observation, we directed our research to find overlapping solutions. We decided to use the OKM (Overlapping K-Means) [4], a recently developed algorithm, that is inspired from K-Means. In the evaluation phase, we tried to improve and complete the authors experiments, by studying the behavior of the algorithm when using different term weighting schemes.

In our search to make our algorithm as noise resistant and language independent as possible, we chose a statistical approach for extracting the keyphrases from the text. We have tested our approach with two languages and data sets of different types: newspaper articles and informal forum discussions.

Another goal of our work was to be able to compare the performances of the chosen algorithms with other proposed in the literature. That resulted in a modular application design, that allow replacing of any module with a similar one. This way a non-overlapping approach was studied along with the OKM algorithm. With a design compatible to other Data Mining platforms (ex. WEKA) , we believe that we will be able to test even more clustering algorithms and compare their performances.

In our work, we gave special attention to the study of the influence of different term weighting schemes on the general performances of both the clustering phase and the name association phase. In our experiments, we tried to determine which one behaves better when faced with real-world data sets.

We strongly believe that our work in not complete. For the future, there are some enhancements that we would like to bring forward:

- **a better cluster - name association.** In the present state, our algorithm associates names to clusters based only on the similarity between the center of each cluster and the name candidate. This can sometimes result in similar names for different clusters. We believe that, by modifying the name association scheme, so that it will take into consideration also the similarity with the other centers, we can manage to find names that are representative for the cluster in question, but dissimilar to the others;
- **adding a wOKM implementation.** wOKM [5] a weighted version of OKM, that uses weights internally and achieves even better performances in terms of **precision**, **recall** and **FScore**. We believe that switching from OKM to wOKM would improve the overall quality of the clustering process.
- **extraction of name candidates from the entire corpus.** Experiments have shown that with the increase of number of clusters, the size of the text from which the name candidates are extracted diminishes. That leads to a decrease of name quality. We believe that we could avoid this problem by extracting the name candidates from the entire corpus of documents. Still this will mean that a better name association scheme must be devised.
- **a new semi-automated manner of cluster names evaluation.** Though we performed an expert-based evaluation, we believe that we could automatize the process. In order to test the

performances of topic extraction with different clustering approaches, we can not rely only on user evaluation. Our future solution would be the extraction of all cluster name candidates from the text of the documents in the corpus and the scoring of each one. Once the list of scored phrases is complete, using the partition scoring that we proposed for our evaluation, we could automatically evaluate the performances of different clustering techniques.

- **experiments on a larger scale.** Due to the limited time allocated to this project, we were unable to perform detailed experiments on the design. We would like to have the chance of experimenting on more data sets, possibly in other languages than English and French.

References

[1] Henry Anaya-Sánchez, Aurora Pons-Porrata, and Rafael Berlanga-Llavori. A new document clustering algorithm for topic discovering and labeling. In *CIARP '08: Proceedings of the 13th Iberoamerican congress on Pattern Recognition*, pages 161–168, Berlin, Heidelberg, 2008. Springer-Verlag.

[2] I Biskri, J G Meunier, and S Joyal. L'extraction des termes complexes : une approche modulaire semiautomatique. In *Données Textuelles, Louvain-La-Neuve, Belgique), Gérard Purnelle, Cédrick Fairon & Anne Dister (eds). Presses Universitaires de Louvain, Volume 1, pp 192201, ISBN*, pages 2–930344, 2004.

[3] David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:2003, 2003.

[4] Guillaume Cleuziou. Okm : une extension des k-moyennes pour la recherche de classes recouvrantes. In Monique Noirhomme-Fraiture and Gilles Venturini, editors, *EGC*, volume RNTI-E-9 of *Revue des Nouvelles Technologies de l'Information*, pages 691–702. Cépaduès-Éditions, 2007.

[5] Guillaume Cleuziou. Okmed et wokm : deux variantes de okm pour la classification recouvrante. In Jean-Gabriel Ganascia and Pierre Gancarski, editors, *EGC*, volume RNTI-E-15 of *Revue des Nouvelles Technologies de l'Information*, pages 31–42. Cépaduès-Éditions, 2009.

[6] Guillaume Cleuziou and Jacques-Henri Sublemontier. Étude comparative de deux approches de classification recouvrante : Moc vs. okm. In Fabrice Guillet and Brigitte Trousse, editors, *EGC*, volume RNTI-E-11 of *Revue des Nouvelles Technologies de l'Information*, pages 667–678. Cépaduès-Éditions, 2008.

[7] Joaquim da Silva, Gaël Dias, Sylvie Guilloré, and Pereira. Using localmaxs algorithm for the extraction of contiguous and non-contiguous multiword lexical units. *Progress in Artificial Intelligence*, page 849, 1999.

[8] G. Dias, S. Guilloré, and J. Gabriel Pereira Lopes. Extraction automatique d'associations textuelles à partir de corpora non traités. In M. Rajman and J.-C. Chapelier, editors, *JADT 2000 - 5èmes Journées Internationales d'Analyse Statistique de Données Textuelles*, volume 2, pages 213–220, Lausanne, 22-24 March 2000. Ecole Polytechnique Fédérale de Lausanne.

[9] Filippo Geraci, Marco Pellegrini, Marco Maggini, and Fabrizio Sebastiani. Cluster generation and cluster labelling for web snippets: A fast and accurate hierarchical solution. pages 25–36. 2006.

[10] Khaled M. Hammouda, Diego N. Matute, and Mohamed S. Kamel. Corephrase: Keyphrase extraction for document clustering. *MLDM*, 2005:265–274, 2005.

[11] N. Jesper Larsson. Notes on suffix sorting. (LU-CS-TR:98-199, LUNDFD6/(NFCS-3130)/1–43/(1998)), jun 1998.

[12] J. B. Macqueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[13] Jaeseok Myung, Jung-Yeon Yang, and Sang-goo Lee. Picachoo: a tool for customizable feature extraction utilizing characteristics of textual data. In *ICUIMC '09: Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication*, pages 650–655, New York, NY, USA, 2009. ACM.

[14] Aristomenis Thanopoulos Nikos, Nikos Fakotakis, and George Kokkinakis. Comparative evaluation of collocation extraction metrics. In *In Proceedings of the 3rd Language Resources Evaluation Conference*, pages 620–625, 2002.

[15] Stanislaw Osinski. An algorithm for clustering of web search results. Master's thesis, Poznań University of Technology, Poland, 2003.

[16] Stanislaw Osinski and Dawid Weiss. Conceptual clustering using lingo algorithm: Evaluation on open directory project data. pages 369–377, 2004.

[17] Aurora Pons-Porrata, Rafael Berlanga-Llavori, and José Ruiz-Shulcloper. Topic discovery based on text mining techniques. *Inf. Process. Manage.*, 43(3):752–768, 2007.

[18] M. Roche. *Intégration de la construction de la terminologie de domaines spécialisés dans un processus global de fouille de textes.* PhD thesis, Université de Paris 11, Décembre 2004.

[19] Frank A. Smadja. From n-grams to collocations: an evaluation of xtract. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics*, pages 279–284, Morristown, NJ, USA, 1991. Association for Computational Linguistics.

APPENDICES

A Side-by-side Precision and Recall, OKM vs. KMeans.

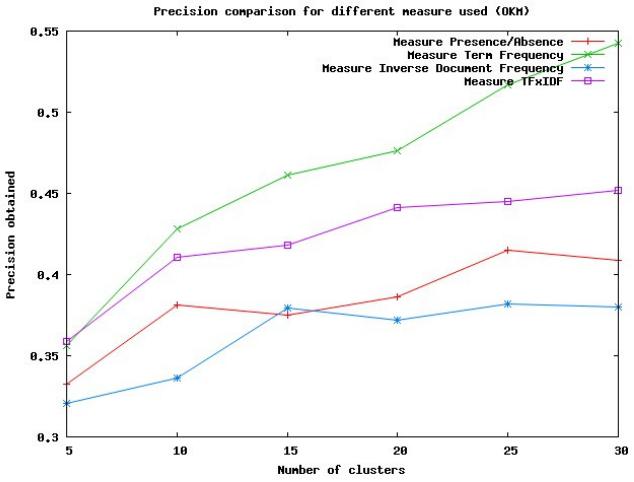


Figure 10: Precision obtained for **OKM**, for different term weighting schemes.

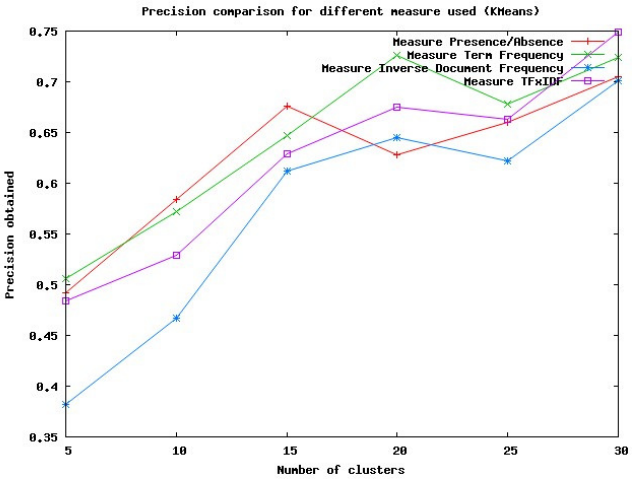


Figure 11: Precision obtained for **KMeans**, for different term weighting schemes.

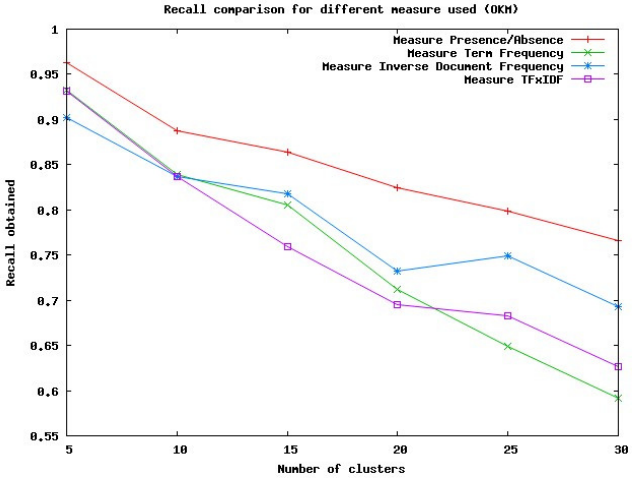


Figure 12: Recall obtained for **OKM**, for different term weighting schemes.

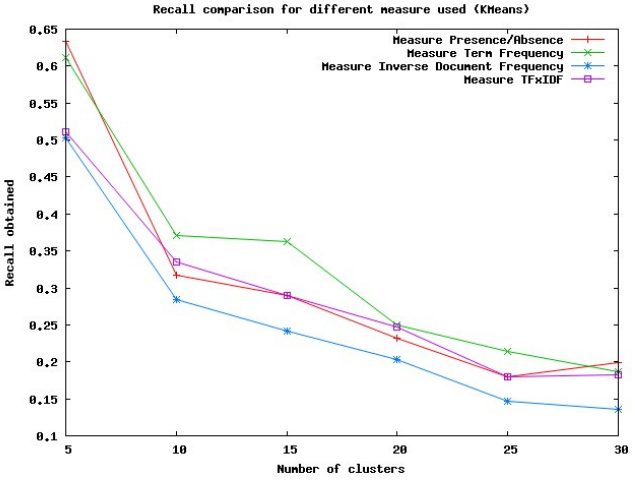


Figure 13: Recall obtained for **KMeans**, for different term weighting schemes.

B Cluster Name Quality. Forum Commémoration

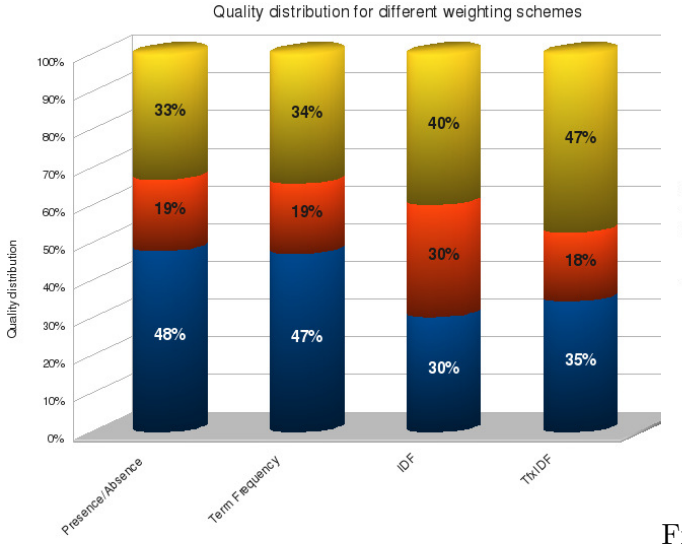


Figure 14: Name Quality: Different Term Weighting Schemes used (Commémoration 5-25).

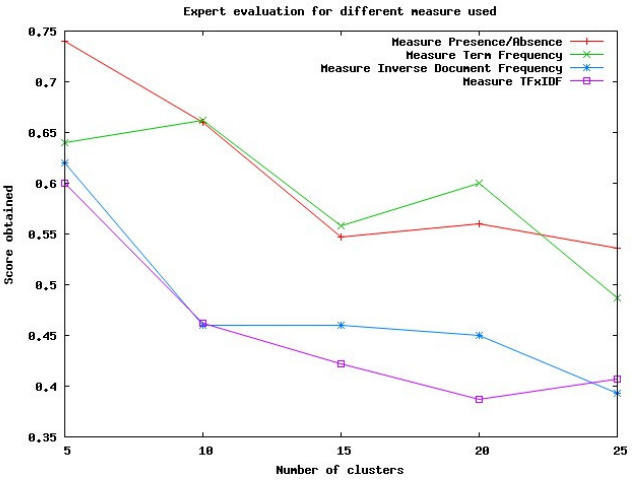


Figure 15: Name Quality: Different Term Weighting Schemes and Cluster Number Variation (Commémoration 5-25)

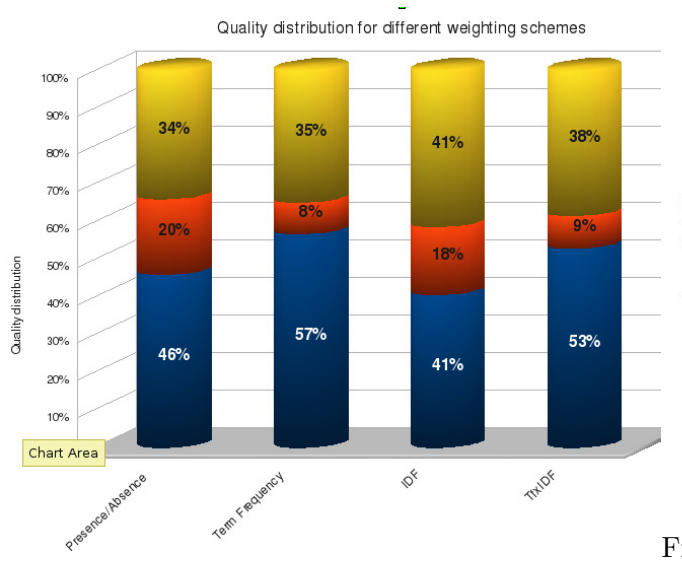


Figure 16: Name Quality: Different Term Weighting Schemes used (Commémoration 8-12)..

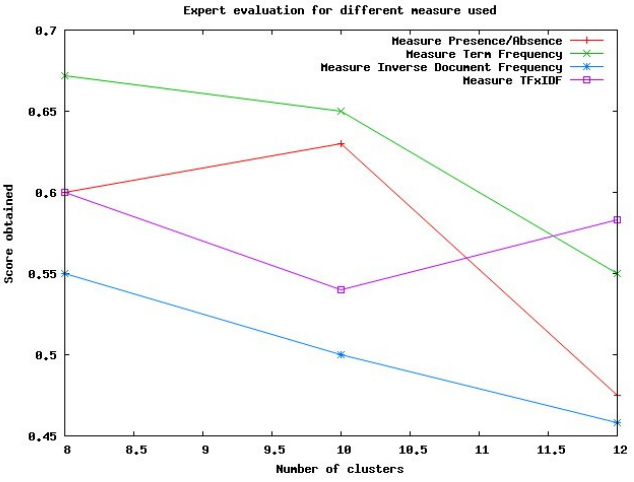


Figure 17: Name Quality: Different Term Weighting Schemes and Cluster Number Variation (Commémoration 8-12)