Co-clustering documents and words by minimizing the normalized cut objective function[*]

Chares-Edmond Bichot
Université de Lyon, CNRS,
Ecole Centrale de Lyon, LIRIS, UMR5205, F-69134, France

May 3, 2010

Abstract

This paper follows a word-document co-clustering model independently introduced in 2001 by several authors such as I.S. Dhillon, H. Zha and C. Ding. This model consists in creating a bipartite graph based on word frequencies in documents, and whose vertices are both documents and words. The created bipartite graph is then partitioned in a way that minimizes the normalized cut objective function to produce the document clustering. The fusion-fission graph partitioning metaheuristic is applied on several document collections using this word-document co-clustering model. Results demonstrate a real problem in this model: partitions found almost always have a normalized cut value lowest than the original document collection clustering. Moreover, measures of the goodness of solutions seem to be relatively independent of the normalized cut values of partitions.

Keywords

Graph partitioning, normalized cut, fusion-fission, document clustering, information retrieval, data mining

1 Introduction

Co-clustering has received lots of attention in recent years, with several applications in text mining, image, speech and video analysis [WNL05, RDT06], and bioinformatics [MO04]. It has also been studied more theoretically [DMM03, ADK08]. In this paper, we only focus on the text mining application.

Both document clustering and word clustering are well studied problems in data mining. Document clustering methods are generally based on agglomerative hierarchical clustering algorithms or k-means algorithms [ZKF05, SKK00]. Although word clustering is a different approach, its finality is often document classification [BM98].

Given a collection of unstructured text data, the document clustering problem is to group documents of similar subjects/topics together. The basic idea is first to extract unique content-bearing words from the set of documents treating

[*]This article has been published in the Journal of Mathematical Modelling and Algorithms (JMMA): Volume 9, Issue 2 (2010), Pages 131–147

these words as features and second to represent each document as a vector (also known as bag-of-words) in this feature space. Word frequencies are taken into account in this feature space as vector entries. A clustering is then made based on these feature vectors.

Although most of the clustering literature focuses on one-sided clustering algorithms, co-clustering has recently gained attention as a powerful tool that allows to circumvent some limitations of classical clustering approach.

Clustering techniques yield *global patterns* because they use cohesion measure which are most of the time based on (global) syntactic similarities of the objects in a cluster [JMF99]. However, in high-dimensional setting, some groups of objects tend to be co-related only under subsets of attributes. Hence, though semantically-related, the similarity of two objects with several differences in their attribute values would hardly be recognized by a global model. In fact, the object cohesion measure is better viewed in terms of *local patterns*. Precisely, an object can be described by a set of concepts, each of which being a collection of characterizing attributes. Then, two objects can be considered as similar if they have at least one concept in common. Thus, in the global perspective, two objects are only compared based on their attribute values, and, in the local perspective, two objects are compared based on mutual concepts (two different concepts can have attribute values in common). By taking into account the multimodal dimension of the problem, co-clustering combines more finely local and global patterns than clustering alone.

The goal of co-clustering documents and words is to cluster documents based on the common words that appear in them and to cluster words based on the common documents that they appear in. The model commonly used for documents is the vector space model. A set of words is chosen from the set of all words in all documents using some criterion. Each document is rendered as a vector formed by the presence of the chosen words in the document. Thus, the entire document set can be represented by a word-document matrix $\mathbf{M}$ where rows denote the words and columns represent the documents.

For any moderately sized document set, the number of words runs into a few thousand. Thus the word-document matrix $\mathbf{M}$ is extremely sparse. It is known that graph partitioning performs better when the matrix is sparse because the average degree of a vertex is low [PSL90]. Thus in this work we applied graph partitioning methods based on metaheuristics to co-cluster documents and words by simultaneously clustering rows and columns of $\mathbf{M}$.

Section 2 details the construction of the word-document bipartite graph use to cluster documents. Section 3 presents related work of word-document co-clustering. The fusion-fission graph partitioning metaheuristic is presented in section 4. Section 5 enumerates several evaluation measures of performance for document clustering. Fusion-Fission results are presented in section 6. Finally, section 7 concludes the paper.

2 Co-clustering graph model

In this section, we introduce the graph model on which co-clustering documents and words is made. This graph model has already been presented in [ZHD+01, DHZ+01, Dhi01].

2.1 Word-document co-clustering

The basic idea behind word-document co-clustering is that document clustering induces word clustering while word clustering induces document clustering.

Given a set of documents collections $D_1, \ldots, D_k$, the corresponding word clusters $W_1, \ldots, W_k$ are constructed as follows. A word w_i of document D_j is in the word cluster W_j if and only if the association between w_i and D_j is greater than any other association between w_i and another document collection D_l $(l \neq j)$.

Similarly, given a set of word clusters $W_1, \ldots, W_k$, the documents collections $D_1, \ldots, D_k$ are constructed by putting a document d_i in the document collection D_j if and only if d_i is more connected with W_j than with any other word cluster.

Clearly, this is a recursive process, because document collections produced word clusters which induced new document collections and so on. This recursive process is achieved when a global optimum is reached. This optimum corresponds to a partition of the set of documents collections and of the set of word clusters into k parts, $W_1 \cup D_1, \ldots, W_k \cup D_k$, such that in each part the sum of the associations between words and documents is maximized.

2.2 Bipartite graph

An undirected bipartite graph $G = \{W, D, E\}$ has two sets of vertices, W and D, and a set of edges, E. As G is a bipartite graph, every edge of E connects a vertex in W to one in D; that is, W and D are independent sets. Let n and m represent the number of vertices in D and W. Every edge $(w_i, d_j) \in E$ is weighted by $\mathbf{M}_{ij}$. Thus $\mathbf{M}$ is the m by n graph weight matrix of G.

If we let D be the set of documents and W be the set of words, then the weighted bipartite graph G represents documents and words. An edge (w_i, d_j) exists if word w_i exists in document d_j. Thus, an edge signifies an association between a document and a word. By putting positive weights on the edges in our model, we can capture the strength of the association between a document and a word. Several possibilities exist for the weighting of the edges. One of the most simplest and intuitive is word frequencies in documents. Such as most of the previous word-document co-clustering works, we use this possibility in this paper.

Co-clustering is realized by partitioning G into as many clusters as there are document collections. Figure 1 shows a word-document bipartite graph partitioned into two document collections using dotted lines.

2.3 Graph partitioning problem

Now that we have sets of documents and words modeled as a bipartite graph, the co-clustering task becomes a graph partitioning job.

Let $G = (V, E)$ be an undirected edge-weighted graph with vertex set $V = \{v_1, \ldots, v_n\}$ and edge set E. Each edge $(v_i, v_j) \in E$ is weighted by w_{ij}.

A partition of the vertex set V into k parts is a set $\Pi_k = \{V_1, \ldots, V_k\}$ of disjoint non-empty subsets of V such that their union is V.

A classical problem in graph partitioning is to find a partition which minimizes the cut of the partition, i.e. the sum of the weights of the crossing edges between parts of the partition.

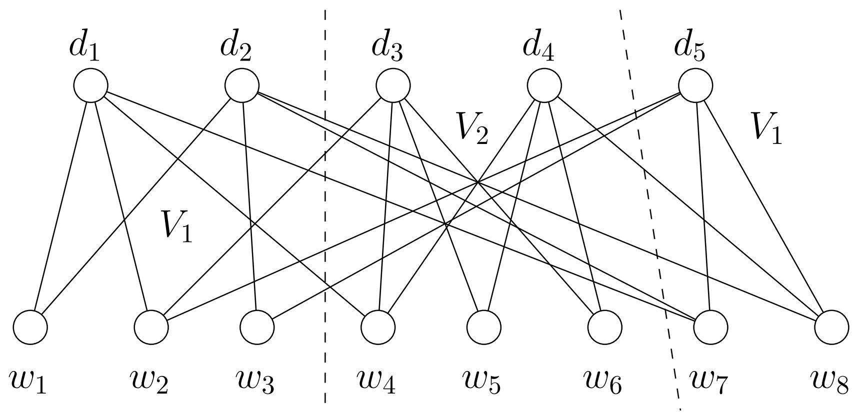

Figure 1: Vertices on the top indicate documents and those on the bottom indicate words. Co-clustering is realized by partitioning this bipartite graph into as many clusters as there are document collections (in this example 2 clusters, V_1 and V_2)

More formally, given a partition of the vertex set V into two subsets V_1 and V_2, the cut between them is defined as :

$$cut(V_1, V_2) = \sum_{v_i \in V_1, v_j \in V_2} w_{ij}$$

This definition can be easily extended to a partition of V into k parts $V_1, \ldots, V_k$:

$$cut(V_1, \ldots, V_k) = \sum_{i<j} cut(V_i, V_j)$$

Section 2.2 modeled documents and words as a bipartite graph $G = (W, D, E)$. The above definitions can be used for a bipartite graph if we consider that the set of vertices V as the union of the set of documents D and the set of words W. Thus a k partition of the bipartite graph is a set $\{V_1 = W_1 \cup D_1, \ldots, V_k = W_k \cup D_k\}$ where $\{W_1, \ldots, W_k\}$ and $\{D_1, \ldots, D_k\}$ are partitions of W and D. Moreover, for any edge i, j in E, $w_{ij} = \mathbf{M}_{ij}$.

As explained in section 2.1, the word-document co-clustering problem is to maximize in each part of the bipartite graph partition associations between words and documents. If parts' sizes would have been equalled, then this problem would have been equivalent to a cut minimization problem. But parts are not equalled because document collections and word clusters could be of different sizes.

By using this objective, the optimal bi-partition of a word-document bipartite graph will be composed by a set containing all documents and all words minus one and by a set containing only the word which appears the less in all documents. Because this mathematical optimum is obviously not the document clustering result searched, there is a need of another objective function. Constraints on parts' sizes will not be useful too because there is no information about the size of document collections.

2.4 Normalized cut objective function

Clearly, we need an objective function which captures the need of more balanced parts in addition to maximize associations within each part. An objective function which responds to these needs is the normalized cut objective function. This function is widely used for word-document co-clustering; the first works to use it are [ZHD+01, DHZ+01, Dhi01].

Let $G = (V, E)$ be an undirected edge-weighted graph and $\Pi_k = \{V_1, \ldots, V_k\}$ be a partition of V. The normalized cut objective function may be expressed as

$$Ncut(\Pi_k) = \sum_{i=1}^{k} \frac{cut(V_i, V - V_i)}{weight(V_i)}$$

where $weight(V_i)$ is the sum of the weights of the edges which have at least one vertex in part V_i, more formally

$$weight(V_i) = \sum_{v_l \in V_i} \sum_{j} w_{lj} = cut(V_i, V) \ ,$$

and $cut(V_i, V - V_i)$ is the sum of the weights of the edges which have exactly one vertex in part V_i.

Thus, the minimization of the $Ncut$ function allows to find a partition which maximizes associations within each part of the partition while balancing the partition.

It should be noticed that the minimization of the normalized cut function is just an approximation of the word-document co-clustering problem. Therefore, an optimal partition regarding the normalized cut objective function could not be optimal regarding the word-document co-clustering problem. This drawback will be pointed out in section 6.

3 Related work

Recently co-clustering of document and words has become a topic of extensive interest due to its applications to text, Web and multimedia documents.

Slonim and Tishby used an agglomerative clustering version of the Bottleneck method [ST00]. This double clustering procedure captures first the mutual information with the set of document by clustering words, and then it finds documents clusters that preserve the information about the word clusters.

Zha et al. [ZHD+01] proposed the data clustering method used in this paper to create a word-document bipartite graph. After the graph construction step, the graph is partitioned using a partial singular value decomposition of the associated edge weight matrix of the bipartite graph.

Dhillon [Dhi01] used a classical graph partitioning technique, the spectral method, for partitioning the bipartite graph constructed in the same way as in [ZHD+01]. The word-document matrix $\mathbf{M}$ can be viewed as a graph G where vertices are both documents and words and edges word frequencies in documents. The spectral method is based on the Fiedler's eigenvectors of the Laplacian matrix of the graph G.

Based on its previous work and on the work of Slonim and Tishby, Dhillon et al. have presented in [DMM03] a co-clustering algorithm that intertwines word and document clustering at all stages and continually improves both until a local minimum is found (the difference between the mutual information of the clustering found and those of the original clustering is minimized).

Mandhani et al. proposed a two-step algorithm that hierarchically cluster documents [MJK03]. The first step is a partitioning step made by an algorithm which divides the word-document matrix **M** into a great number of submatrices (or clusters). The second step is an agglomerative step which reduces the number of clusters and makes it coarser.

Long et al. introduced the block value decomposition framework [LZY05]. Their idea is to factorize the word-document matrix **M** into three components, the row-coefficient matrix, the block value matrix ant the column-coefficient matrix. The coefficients denote the degrees of the rows and columns associated with their clusters and the block value matrix is an explicit and compact representation of the hidden block structure of **M**.

Rege et al. proposed an isoperimetric co-clustering algorithm (ICA) for partitioning the word-document matrix **M** [RDF06]. ICA used the same model than spectral partitioning but instead of searching the solutions of the singular word-document system of linear equations, it converts the system to a non-singular system of equations which is easiest to solve.

Tjhi and Chen have presented an algorithm of fuzzy co-clustering with Ruspini's condition for co-clustering documents and words [TC06]. Their work, based on previous fuzzy document clustering works, is the first fuzzy algorithm which allows the generation of natural fuzzy word clusters and fuzzy document clusters.

Costa et al. introduced a hierarchical model-based co-clustering algorithm [CGO08]. In their method the co-occurrence matrix is characterized in probabilistic terms, by estimating the joint distribution between rows and columns.

Recently, a new topic of interest has been to cluster multi-topic documents. Andrea Tagarelli and George Karypis proposed a segment-based document clustering framework, which is designed to induce a classification of documents starting from the identification of cohesive groups of segment-based portions of the original documents [TK08]. Their approach aims to create "natural" composition of documents in text segments.

Our contribution aims to give a new look at the co-clustering documents and words method proposed in [ZHD$^+$01, DHZ$^+$01, Dhi01]. Some drawback are pointed out in this paper which put back into question the usefulness of this model, especially the use of the normalized cut objective function.

4 The fusion-fission metaheuristic

The fusion-fission algorithm was introduced in its first form in [Bic06]. This first version has demonstrated its performances on air traffic management partitioning against classical partitioning tools and metaheuristics in [Bic07]. The algorithm has evolved since this first version to become a more flexible and efficient single-parameter metaheuristic for graph partitioning [Bic08]. The fusion-fission metaheuristic presented in this paper has only two parameters, the number of iterations of the algorithm and the fundamental graph partitioning setting: the number of parts of the partition, k.

Like the multilevel method [HL95, Wal04], the fusion-fission method is a graph partitioning metaheuristic. The multilevel method is a three step method based on the coarsening of the vertices of the graph. When the number of coarsened vertices has reached some threshold, the coarsened graph is partitioned and projected recursively to the original graph using a refinement heuristic. This refinement heuristic is almost always based on the Kernighan-Lin notion of gain [KL70] coupled with the implementation of Fiduccia-Mattheyses [FM82].

The fusion-fission method is also a three step method. As the method's name suggests, the two first steps are a fission step and a fusion step. The last step is a refinement step. Algorithm 1 shows these three steps.

There are some fundamental differences between Kernighan-Lin refinement heuristic, multilevel methods and fusion-fission method. The Kernighan-Lin refinement heuristic is based on vertices moves, the multilevel algorithm is based on coarsening vertices moves, and the fusion-fission method is designed to be based on partition's parts moves.

Algorithm 1 Fusion-Fission

```
procedure FusionFission($G = (V, E)$, $k$)
    $\Pi \leftarrow$ partition $G$ into $k$ parts
    LocalSearch($\Pi$)
    repeat
        Choose $l' \neq l$ in the neighborhood of $k$
        % FISSION step
        for $V_i \in \Pi = \{V_1, \ldots, V_l\}$ do
            Split $V_i$ into $l'$ parts
        end for
        % FUSION step
        Create a graph $G'$ with vertices the $l * l'$ parts created during the fission step
        $\Pi \leftarrow$ partition $G'$ into $l'$ parts
        % REFINEMENT step
        LocalSearch($\Pi$)
    until Stopping condition
    return the best $k$-partition found
end procedure
```

The fusion-fission algorithm starts by creating a k-partition of the graph G. For this purpose a multilevel algorithm is used, the MeTiS serial graph partitioning program[1] [KK98]. A local search heuristic is then applied to locally optimize the partition. The local search heuristic used is the Global Kernighan-Lin Refinement heuristic presented in [KK98]. This refinement heuristic is also those of MeTiS.

The main loop of the fusion-fission algorithm iteratively repeats the three steps: fission, fusion and refinement. A random iteration starts with a partition Π of l parts, with $l \geq 2$ an integer in the neighborhood of k. The loop begins by choosing an integer l' in the neighborhood of k. This neighborhood is created using a binomial distribution centered on k. During the fission step, each of the l parts of Π is split into l' parts. The MeTiS program makes this splitting. The

[1]The MeTiS serial graph partitioning program is available at http://glaros.dtc.umn.edu/gkhome/views/metis

fusion step starts by creating a graph G' with vertices the $l * l'$ parts computed during the fission step. Edges of G' have weights the cut between two parts. The partition of G' into l' parts is produced by the MeTiS program. This partition is directly projected to the original graph G to create the new partition Π. After the fusion step, the refinement step refines the partition Π with the help of the Global Kernighan-Lin Refinement heuristic.

The MeTiS program is used intensively all along the fusion-fission algorithm. Of course, all graph partitioning program should be working in substitution of MeTiS. It can be noticed that MeTiS only serves in the fusion-fission algorithm as a graph partitioning heuristic.

5 Evaluation

The problem of evaluating word-document co-clustering has been solved in many ways in previous works. However, the evaluation process almost always begins with the clustering of document collections for which the "true" classification is known.

Two document collections are commonly used in word-document clustering works. The Classic3 document collection (see section 6.1) is the most widely used [Dhi01, MJK03, LZY05, RDF06, CGO08]. Some works also used the Yahoo_K document collection (see section 6.1) [Dhi01, MJK03, RDF06].

The most easiest evaluation process is to compare directly the clustering result with the true documents' classification on a matrix (named confusion matrix, see section 5.1 below) whose lines represent the true classification and rows represent the clustering result. Such an evaluation is represented by table 3. This evaluation process is commonly proposed as the only measure to evaluate performance of algorithms in word-document clustering works. This evaluation process has two significant drawbacks.

First, this process does not allow to evaluate the reproducibility of results. Is the result proposed relevant or is it the best among all results ever found by the algorithm ? You can bet on the second possibility. Thus, there is a need for several runs of the algorithm to compute an average measure of the results to evaluate the performance of the algorithm. To compare results with the true classification, measures of comparison between partitions are presented in sub-section 5.1.

The second drawback is about the evaluation of the algorithm's performance on the optimization problem independently to the document clustering problem. The word-document co-clustering problem is modeled as an optimization process where the objective function to minimize is the normalized cut objective function and the constraint is only the number of parts of the partition. Given this model, it is difficult to evaluate the performance of the algorithm if we do not have the normalized cut value of the result. It would be interesting to have the normalized cut of the true classification too.

Furthermore, the computation time is often not indicated in word-document co-clustering works.

5.1 Measures of comparison between two partitions

In order to compare clustering results against external criteria, a measure of agreement is needed [RM05]. The following measures can be used to judge cluster quality given an external classification of the documents $\Pi_c = \{c_1, \dots, c_k\}$. Given a set $V = \{v_1, \dots, v_n\}$ of n objects, suppose $\Pi_r = \{r_1, \dots, r_k\}$ represents the clustering result to compare to Π_c.

5.1.1 Purity

One of the ways of measuring the quality of a clustering solution is cluster purity. Purity is the classification accuracy assuming all members of a cluster were predicted to be members of the dominant class in that cluster.

Formally, purity is defined to be:

$$\frac{1}{d}\sum_{j=1}^{k} max_{c_i \in \Pi_c} n_{ij} \ ,$$

where d is the number of documents and n_{ij} is the number of objects that are in both part c_i and part r_j. More formally, $n_{ij} = |c_i \cap r_j|$. The $k * k$ values of n_{ij} form the confusion matrix (or matching matrix) of Π_c and Π_r.

In general, larger the value of purity better the solution.

Purity does not take into account all n_{ij} values of the confusion matrix of Π_c and Π_r, but other measures like entropy or F-measure do. However, some word-document co-clustering works used purity [MJK03], thus, for comparison purpose, we used it too.

5.1.2 F-measure

F-measure (or F-score) is a statistical classification measure of a test's accuracy. It considers both the precision and the recall of the test to compute the score: the precision is the number of correct results divided by the number of all returned results and the recall is the number of correct results divided by the number of results that should have been returned.

Formally,

$$precision = \frac{1}{k}\sum_{j=1}^{k} \frac{n_{jj}}{n_{.j}}$$

and

$$recall = \frac{1}{k}\sum_{i=1}^{k} \frac{n_{ii}}{n_{i.}} \ ,$$

where $n_{i.}$ is the number of objects in part c_i and $n_{.j}$ is the number of objects in part r_j.

The traditional/balanced F-measure (also known as F1-measure) is defined as:

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

The F1-measure can be interpreted as a weighted average of the precision and recall, where an F1-measure reaches its best value at 1 and worst value at 0.

5.1.3 Adjusted Rand index

In the context of classification tasks, the statistical type I and type II errors are used to compare the cluster quality of a result, Π_r, with the true classification of the documents, Π_c. There are four different types among the $\binom{n}{2}$ distinct pairs that could be found:

- the number of pairs of objects that are placed in the same part in Π_c and in the same part in Π_r, is the number of true positives, a;
- the number of pairs of objects that are placed in the same part in Π_c, but not in the same part in Π_r, is the number of false positives, b;
- the number of pairs of objects that are placed not in the same part in Π_c, but in the same part in Π_r, is the number of false negatives, c;
- the number of pairs of objects that are placed not in the same part in Π_c and not in the same part in Π_r, is the number of true negatives, d.

The Rand index [Ran71] is simply

$$\frac{a+d}{a+b+c+d} .$$

The Rand index lies between 0 and 1, and when two partitions agree perfectly, its value is 1. The problem with the Rand index is that the expected value of the Rand index of two random partitions does not take a constant value (say zero).

The adjusted Rand index proposed by [HA85] assumes the generalized hypergeometric distribution as the model of randomness which allows correcting the Rand index drawback. The general form of an index with a constant expected value is

$$\frac{index - expectedindex}{maximumindex - expectedindex},$$

which is bounded above by 1, and takes the value 0 when the index equals the expected value. The adjusted Rand index has the form:

$$aRi = \frac{\sum_i \sum_j \binom{n_{i,j}}{2} - \left[\sum_i \binom{n_{i\cdot}}{2} \sum_j \binom{n_{\cdot j}}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_{i\cdot}}{2} + \sum_j \binom{n_{\cdot j}}{2}\right] - \left[\sum_i \binom{n_{i\cdot}}{2} \sum_j \binom{n_{\cdot j}}{2}\right] / \binom{n}{2}} .$$

6 Results

In this section, we will first introduce the data set collections used for our experiments, then we will present fusion-fission results and finally we will compare these results with those of other papers.

6.1 Data sets

We have used the two data sets introduced in section 5 for our experiments:

Name	# Docs	# parts	# Words	# Edges
MedCisi	2493	2	11 616	118 304
MedCran	2431	2	10 683	129 601
CisiCran	2858	2	7 432	137 809
MedCranCisi	3891	3	13 182	192 857
Yahoo_K1	2340	6	21 839	349 792
Yahoo_K5	2340	6	1 458	237 969

Table 1: Details of the document test collections

- Classic3: It contains 1033 medical abstracts from the Medline collection, 1400 aerospace systems abstracts from the Cranfield collection and 1460 information retrieval abstracts from the Cisi collection. It can be downloaded from ftp://ftp.cs.cornell.edu/pub/smart/. For testing algorithm FusionFission, we created mixtures consisting of two or three of these document collections. For example, the document test collection MedCisi contains documents from the Medline and Cisi collections.

- Yahoo_K: It contains 2340 Reuters news articles downloaded from Yahoo in 1997 [BGG+99]. These articles are from 6 categories. There are 494 articles from health, 1389 from entertainment, 141 from sport, 114 from politics, 60 from technology and 142 from business. This data set can be downloaded from
http://www-users.cs.umn.edu/~boley/ftp/PDDPdata/.

Table 1 presents details of all the document test collections. Contrary to many other works, we take into account only 1398 documents from the Cranfield collection because two of these documents are empty. Thus, the number of documents in MedCran, CisiCran and MedCranCisi collections differs of two from other works.

Before co-clustering word and documents, Classic3 document test collections needs a preprocessing step to create word-document graphs [Dhi01]. The preprocessing step constructs a graph for each of the document test collections. First, stop words are removed from documents and words are stemmed using Porter's algorithm [Por80]. Then, words' frequencies are counted into each document to create the bipartite word-document graph presented in section 2.2. The preprocessing step is useless for Yahoo_K document test collections because stemmed word occurrences are given with words' frequencies.

6.2 Fusion-Fission results

The fusion-fission algorithm presented in section 4 has been applied to the document test collections presented in table 1. To be meaningful, these results are the arithmetic mean of 500 iterations of the fusion-fission algorithm running on permutations of the initial word-document bipartite graph. Table 2 summarizes these results. Each word-document partition is evaluated by four measures, the normalized cut function which is the objective function to minimize, the measure of purity, the F-measure and the adjusted Rand index (shorted as aRi). Normalized cut values are compared for each document test collection to the normalized cut value of the true (original) document classification. The

Test coll.	$Ncut$	purity	Fmeasure	aRi	t$Ncut$	# best	time
MedCisi	0.3353	0.974	0.974	0.951	0.3402	500	6.4
MedCran	0.3035	0.993	0.993	0.986	0.3037	415	7.8
CisiCran	0.3487	0.993	0.993	0.985	0.3492	500	7.5
MedCranCisi	0.7516	0.971	0.971	0.946	0.7586	500	11.6
Yahoo_K1	3.2044	0.817	0.601	0.740	3.4307	500	25.7
Yahoo_K5	3.7514	0.775	0.537	0.601	4.0920	500	12.1
Best if	Min	Max	Max	Max	–	Max	Min

Table 2: Mean of 500 fusion-fission results for the document test collections. t$Ncut$: $Ncut$ value of the true document classification; # best: the number of fusion-fission results with lowest $Ncut$ values than t$Ncut$; time is the mean time of execution in seconds

	Medline	Cranfield	Cisi
C_1	967	0	5
C_2	11	1390	13
C_3	55	8	1442
Sum	1033	1398	1460
purity=0.976, F-measure=0.976, aRi=0.954			

Table 3: One result of the fusion-fission algorithm applied to MedCranCisi. $Ncut = 0.751$

last column of table 2 enumerates the number of time fusion-fission algorithm returns a partition with a normalized cut value lower than those of the true document classification. This value is bounded by the number of iterations of the fusion-fission algorithm, 500.

A surprising result was to find a word-document partition with a lowest normalized cut than those of the true classification. Moreover, such partitions are found by the fusion-fission algorithm almost every time. However, finding a partition with a lowest normalized cut than those of the true partition does not means that the partition found is close to the true classification. As an example, for Yahoo_K1 – which contains more word-document informations than Yahoo_K5 –, the F-measure value is only 0.601 and the adjusted Rand index value is only 0.740 while $Ncut < \mathrm{t}Ncut$.

	Health	Entertain	Sport	Politics	Tech	Bus
C_1	484	2	0	4	0	4
C_2	4	1179	42	25	2	6
C_3	0	38	84	1	0	0
C_4	0	35	0	82	0	10
C_5	0	55	0	2	58	121
C_6	6	80	15	0	0	1
Sum	488	1384	141	114	60	142
purity=0.858, F-measure=0.631, aRi=0.804						

Table 4: One result of the fusion-fission algorithm applied to Yahoo_K1. $Ncut = 3.254$

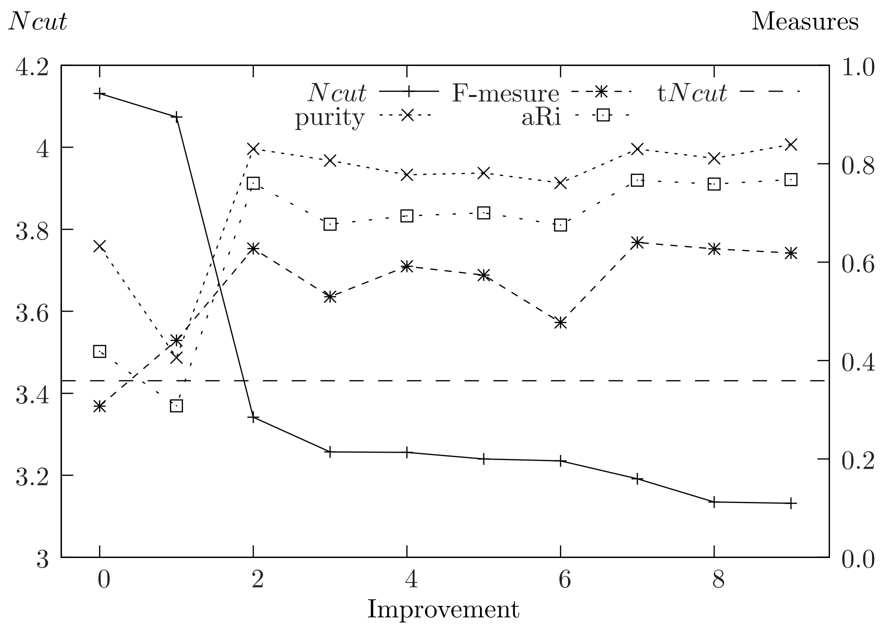

Figure 2: Evolution of purity, F-measure and adjusted Rand index when $Ncut$ decreases during a run of the fusion-fission algorithm. t$Ncut$: true classification's normalized cut

As examples of the results found by the fusion-fission algorithm, table 3 shows the confusion matrix of a result found by fusion-fission for the MedCran-Cisi document collection, while table 4 shows the confusion matrix of a result found by fusion-fission for the Yahoo_K1 document collection.

Figure 2 presents the evolution of the three measures, purity, F-measure and adjusted Rand index, when partitions' normalized cuts values decrease during a single run of the fusion-fission algorithm. This figure clearly shows that measures' values do not really increase when the normalized cut value is decreasing. Moreover, it shows that measures' values seem relatively independent of the normalized cut's value.

Fusion-Fission results raise at least a deficiency, at most a failure, in the word-document co-clustering modeling proposed in [ZHD$^+$01, DHZ$^+$01, Dhi01]. Indeed, in a good modeling measures' values will have exactly followed normalized cut's values in their opposite directions.

[ZK04] has already pointed out a problem with this modeling: the clusters produced when minimizing the normalized cut objective function are greatly unbalanced. But this problem in the case of word-document clustering seems not to be really a problem because the number of documents in each cluster is not known *a priori* neither the balance of the partition, as we notice in section 2.3. Moreover, even if this was a problem, the balance of the partition can be controlled by the algorithm when minimizing the normalized cut as a constraint of the problem.

However the real problem of this modeling is that the normalized cut objective function is not representative of the goodness of the clustering.

Test coll.	$Ncut$	purity	Fmeasure	aRi	t$Ncut$	# best
MedCisi	0.3672	0.902	0.904	0.838	0.3402	0
MedCran	0.4094	0.842	0.847	0.763	0.3037	0
CisiCran	0.3574	0.987	0.988	0.975	0.3492	0
MedCranCisi	0.7677	0.951	0.950	0.913	0.7586	0
Yahoo_K1	3.1244	0.701	0.549	0.590	4.1056	500
Yahoo_K5	3.4088	0.742	0.528	0.634	4.0920	500

Table 5: Mean of 500 Graclus results for the document test collections. t$Ncut$: $Ncut$ value of the true document classification; # best: the number of fusion-fission results with lowest $Ncut$ values than t$Ncut$; time is the mean time of execution in seconds

6.3 Comparison of results

The papers presented in section 3 have not published their code on the Web, thus no software for comparison are available. Nevertheless, for the comparison with fusion-fission results, we use the Graclus software made by I.S. Dhillon [DGK07] which can be used for document co-clustering.

Table 5 presents results found with the Graclus graph partitioning package to the document test collections presented in table 1. This table is formated in the same way as table 2 (for more precision, see section 6.2). Compared with table 2, results of Graclus show that the minimization of the $Ncut$ objective function is better with fusion-fission for the first four document collections and then, it is better with Graclus for the Yahoo document collections. However, regarding purity, F-measure and adjusted Rand index, fusion-fission results are clearly better than Graclus results for any document collection.

Table 6 compares results found on MedCranCisi and Yahoo_K1 document collections by fusion-fission, Graclus, Spectral co-clustering [Dhi01], RPSA [MJK03] and NBVD [LZY05]. For the last three methods, because softwares were not available, results published in the corresponding articles are taken. $Ncut$ results are often not available, thus we can not give any comparison for this objective function. However, this table shows that the purity of results are almost the same for the MedCranCisi document collection (a little lower for Graclus and then fusion-fission), and clearly different for the Yahoo_K1 document collection. For this collection — more difficult to cluster — fusion-fission achieve very good results compared to the other methods, followed by Graclus. For F-measure and adjusted Rand index fusion-fission achieves very good results too. Graclus results are in second position.

This comparison of results shows the fusion-fission efficiency for document co-clustering regarding the other method, but it does not hide the fundamental drawback of the co-clustering approach based on graph partitioning using $Ncut$ presented in section 6.2.

7 Conclusion

This paper studied the word-document co-clustering model introduced in [Dhi01, ZHD+01, DHZ+01]. This model lies on the creation of a bipartite graph which will be afterward partitioned using a graph partitioning algorithm by minimiz-

Algorithm	$Ncut$	purity	Fmeasure	aRi	Comments
MedCranCisi					
Fusion-Fission	0.7516	0.971	0.971	0.946	mean of 500 results
Fusion-Fission	0.7521	0.982	0.982	0.965	one result
Graclus	0.7677	0.951	0.950	0.913	mean of 500 results
Graclus	0.7652	0.971	0.971	0.946	one result
Spectral	NA	0.980	0.979	0.962	one result
RPSA	NA	0.985	0.984	0.970	one result
NBVD	NA	0.988	0.988	0.977	one result
Yahoo_K1					
Fusion-Fission	3.2044	0.817	0.601	0.740	mean of 500 results
Fusion-Fission	3.1314	0.866	0.665	0.800	one result
Graclus	3.1244	0.701	0.549	0.590	mean of 500 results
Graclus	3.0773	0.827	0.639	0.764	one result
Spectral	NA	0.668	0.460	0.564	one result
RPSA	NA	0.648	0.533	0.542	one result
Best if	Min	Max	Max	Max	

Table 6: A comparison between results of fusion-fission, Graclus, Spectral co-clustering [Dhi01], RPSA [MJK03] and NBVD [LZY05]

ing the normalized cut objective function. The fusion-fission method which has already been successfully applied to different graph partitioning problems is tested on several document collections using this word-document co-clustering model.

The results found demonstrate the usefulness of using this co-clustering model. Indeed, results show that the normalized cut objective function is not representative of the goodness of the document and word co-clustering. Thus a new couple objective function, constraints should be proposed to improve this co-clustering model.

Nevertheless, the results stress the ability of the fusion-fission method to be adapted on a new problem and to achieve good results.

References

[ADK08] Aris Anagnostopoulos, Anirban Dasgupta et Ravi Kumar : Approximation algorithms for co-clustering. *In Proceedings of the 27th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 201–210, 2008. 1

[BGG$^+$99] Daniel Boley, Maria Gini, Robert Gross, Sam Han, Kyle Hastings, George Karypis, Vipin Kumar, Bamshad Mobasher et Jerome Moore : Partitioning-Based Clustering for Web Document Categorization. *Decision Support Systems*, 27(3):329–341, 1999. 11

[Bic06] Charles-Edmond Bichot : A metaheuristic based on fusion and fission for partitioning problems. *In Proceedings of NIDISC'08 in conjunction with the 20th IEEE International Parallel and Distributed Processing Symposium*, avril 2006. 6

[Bic07] Charles-Edmond Bichot : A New Method, the Fusion Fission, for the Relaxed k -way Graph Partitioning Problem, and Comparisons with Some Multilevel Algorithms. *Journal of Mathematical Modeling and Algorithms (JMMA)*, 6(3):319–344, 2007. 6

[Bic08] Charles-Edmond Bichot : A new meta-method for graph partitioning. *In Proceedings of the 2008 IEEE Congress on Evolutionary Computation*, pages 3498–3505, juin 2008. 6

[BM98] L. Douglas Baker et Andrew Kachites McCallum : Distributional clustering of words for text classification. *In Proceedings of the 21st ACM SIGIR International Conference on Research and development in information retrieval*, pages 96–103, 1998. 1

[CGO08] Gianni Costa, GiuseppeManco et Riccardo Ortale : A hierarchical model-based approach to co-clustering high-dimensional data. *In Proceedings of the ACM symposium on Applied computing*, pages 886–890, 2008. 6, 8

[DGK07] Inderjit S. Dhillon, Yuqiang Guan et Brian Kulis : Weighted Graph Cuts without Eigenvectors: A Multilevel Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29: 1944–1957, 2007. 14

[Dhi01] Inderjit S. Dhillon : Co-clustering documents and words using bipartite spectral graph partitioning. *In Proceedings of the 7th ACM SIGKDD International Conference on Knowledge discovery and data mining*, pages 269–274, 2001. 2, 5, 6, 8, 11, 13, 14, 15

[DHZ$^+$01] Chris Ding, Xiaofeng He, Hongyuan Zha, Ming Gu et Horst D. Simon : A Min-max Cut Algorithm for Graph Partitioning and Data Clustering. *In Proceedings of the 1st IEEE International Conference on Data Mining*, pages 107–114, 2001. 2, 5, 6, 13, 14

[DMM03] Inderjit S. Dhillon, Subramanayam Mallela et Dharmendra S. Modha : Information-Theoric Co-clustering. *In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge discovery and data mining*, pages 89–98, 2003. 1, 5

[FM82] C. M. Fiduccia et R. M. Mattheyses : A Linear-Time Heuristic for Improving Network Partitions. *In Proceedings of 19th ACM/IEEE Design Automation Conference*, pages 175–181, 1982. 7

[HA85] Lawrence Hubert et Phipps Arabie : Comparing partitions. *Journal of classification*, 2(1):193–218, 1985. 10

[HL95] Bruce Hendrickson et Robert W. Leland : A Multilevel Algorithm For Partitioning Graphs. *In Proceedings of Supercomputing*, 1995. 6

[JMF99] A. K. Jain, M. N. Murty et P. J. Flynn : Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999. 2

[KK98] George Karypis et Vipin Kumar : Multilevel k-way Partitioning Scheme for Irregular Graphs. *Journal of Parallel and Distributed Computing*, 48(1):96–129, 1998. 7

[KL70] B. W. Kernighan et S. Lin : An Efficient Heuristic Procedure for Partitioning Graphs. *Bell System Technical Journal*, 49(2):291–307, 1970. 7

[LZY05] Bo Long, Zhongfei M. Zhang et Philip S. Yu : Co-clustering by block value decomposition. *In Proceedings of the 11th ACM SIGKDD International Conference on Knowledge discovery and data mining*, pages 635–640, 2005. 6, 8, 14, 15

[MJK03] Bhushan Mandhani, Sachindra Joshi et Krishna Kummamuru : A matrix density based algorithm to hierarchically co-cluster documents and words. *In Proceedings of the 12th International Conference on World Wide Web*, pages 658–665, 2003. 6, 8, 9, 14, 15

[MO04] Sara C. Madeira et Arlindo L. Oliveira : Biclustering Algorithms for Biological Data Analysis: A Survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004. 1

[Por80] M.F. Porter : An algorithm for suffix stripping. *Program*, 14(3): 130–137, 1980. 11

[PSL90] Alex Pothen, Horst D. Simon et Kang-Pu Liou : Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, 1990. 2

[Ran71] W.M. Rand : Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66:846–850, 1971. 10

[RDF06] Manjeet Rege, Ming Dong et Farshad Fotouhi : Co-clustering Documents and Words Using Bipartite Isoperimetric Graph Partitioning. *In Proceedings of the 6th IEEE International Conference on Data Mining*, pages 532–541, 2006. 6, 8

[RDT06] Manjeet Rege, Ming Dong et Farshad Toouhi : Co-Clustering Image Features and Semantic Concepts. *In Proceedings of IEEE International Conference on Image Processing*, pages 137–140, 2006. 1

[RM05] Lior Rokach et Oded Maimon : *Data mining and knowledge discovery handbook*, chapitre Clustering methods. Springer, 2005. 9

[SKK00] Michael Steinbach, George Karypis et Vipin Kumar : A comparison of document clustering techniques. *In Proceedings of ACM SIGKDD Workshop on Text Mining*, 2000. 1

[ST00] Noam Slonim et Naftali Tishby : Document clustering using word clusters via the information bottleneck method. *In Proceedings of the 23rd ACM SIGIR International Conference on Research and development in informaion retrieval*, pages 208–215, 2000. 5

[TC06] William-Chandra Tjhi et Lihui Chen : A partitioning based algorithm to fuzzy co-cluster documents and words. *Pattern Recognition Letters*, 27(3):151–159, 2006. 6

[TK08] Andrea Tagarelli et George Karypis : A Segment-based Approach To Clustering Multi-Topic Documents. *In Text Mining Workshop, SIAM Datamining Conference*, 2008. 6

[Wal04] Chris Walshaw : Multilevel Refinement for Combinatorial Optimisation Problems. *Annals of Operations Research*, 131:325–372, 2004. 6

[WNL05] Xiao Wu, Chong-Wah Ngo et Qing Li : Co-Clustering of Time-Evolving News Story with Transcript and Keyframe. *In Proceedings of IEEE International Conference on Multimedia and Expo*, pages 117–120, 2005. 1

[ZHD$^+$01] Hongyuan Zha, Xiaofeng He, Chris H. Q. Ding, Ming Gu et Horst D. Simon : Bipartite Graph Partitioning and Data Clustering. *In ACM Conference on Information and Knowledge Management*, pages 25–32, 2001. 2, 5, 6, 13, 14

[ZK04] Ying Zhao et George Karypis : Empirical and Theoretical Comparisons of Selected Criterion Functions for Document Clustering. *Machine Learning*, 55(3):311–331, 2004. 13

[ZKF05] Ying Zhao, George Karypis et Usama M. Fayyad : Hierarchical Clustering Algorithms for Document Datasets. *Data Mining and Knowledge Discovery*, 10(2):141–168, 2005. 1