χ-Sim: A New Similarity Measure for the Co-clustering Task

Gilles Bisson[1], Fawad Hussain[2]
Laboratoire TIMC-IMAG, CNRS / UJF 5525
Université de Grenoble
Domaine de la Merci, 38710 La Tronche, France
{gilles.bisson}@imag.fr
{fawad.hussain}@imag.fr

Abstract.

Co-clustering has been widely studied in recent years. Exploiting the duality between objects and features efficiently helps in better clustering both objects and features. In contrast with current co-clustering algorithms that focus on directly finding some patterns in the data matrix, in this paper we define a (co-)similarity measure, named χ-Sim, which iteratively computes the similarity between objects and their features. Thus, it becomes possible to use any clustering methods (k-means, …) to co-cluster data. The experiments show that our algorithm not only outperforms the classical similarity measure but also outperforms some co-clustering algorithms on the document-clustering task.

1. Introduction

Clustering is important for automatically organizing or summarizing data coming from databases or experiments. Classically, these data are described as a set of objects characterized by a set of features. In some cases, these features are homogeneous enough to cluster them, in the same way as we do for the objects. For instance, when using the *vector space model* popularized by [7], text corpora are represented by a matrix whose rows represent the object (document vectors) and whose columns represent the feature (word vectors). The similarity between two documents obviously depends on the similarity between the words they contain and vice-versa. In the classical clustering methods, such dependencies are not exploited. The purpose of co-clustering is to take into account such duality between rows and columns to identify the relevant clusters.

In this regard, co-clustering has been largely studied in recent years both in document classification [3] [9] [6] [15] and Bioinformatics [8] [14] [11] for the analysis of gene expression data. In both domains, data can be easily disposed as a co-occurrence table, namely a document-word matrix or a gene-expression matrix.

In the document classification domain, co-clustering approaches are well suited to deal with high dimensional and sparse data [6]. Another benefit of this framework is its capability to find relationships between documents that apparently do not share any common words. For instance, in Table 1 documents d_1 and d_2 do not share any common word w^i; with a classical similarity measure such as the Minkowski distance or the Cosine measure, their similarity equals zero (or the distance between them is maximal). Yet we can observe that both d_1 and d_2 share words with d_3 meaning that words w^2 and w^3 have some similarities in the *document space*. With a co-clustering approach, it is possible to associate a similarity between d_1 and d_2, which of course will be smaller than the ones between d_1 and d_3 or d_2 and d_3 but nonetheless not zero. The justification is based on the fact that two documents discussing the same topic may contain two different sets of words, but the words belonging to each set frequently co-occur in other documents concerning this topic.

Table 1. Illustration of the advantage of co-clustering.

M	w^1	w^2	w^3	w^4
d_1	1	1	0	0
d_2	0	0	1	1
d_3	0	1	1	0

While most authors have focused on creating new methods to co-cluster the data, in this paper we introduce the concept of *co-similarity*. We propose a new similarity measure, named χ-Sim, based on the idea of generating simultaneously the similarity matrices between rows and between columns, each of them iteratively being built on the basis of the other. For example, in the context of document clustering, we create the similarity matrix between documents by taking into consideration the similarity matrix between the words and vice-versa. Hence, it becomes possible to choose the clustering method (K-means, Density-based approaches…) that is best suited to co-cluster the data.

[1] Member of the french National Center for Scientific Research.
[2] This work is part of a Phd Thesis supported by the Higher Education Commission of Pakistan

The rest of this paper is organized as follows: in section 2 we provide a detailed description of χ-Sim and its relationship with classical similarities. In section 3, we propose an efficient implementation of the co-similarity algorithm. In section 4 we discuss the related work in the field of co-clustering that has been used as the basis of our experimental comparison. These experimental results are presented and discussed in section 5.

2. Principle of the Method χ-Sim

2.1. Notations

Throughout this paper we use the classical notation: matrices (in capital letters) and vectors (in small letters) are in bold and all variables (in small letters) are in italic.

Data matrix: let **M** be the data matrix representing a corpus having *r* rows (documents) and *c* columns (words); m_{ij} corresponds to the number of occurrences of the *jth* word in the *ith* document; $\mathbf{m_i} = [m_{i1} \,..\, m_{ic}]$ is the row vector representing the document *i* and $\mathbf{m^j} = [m_{1j} \,..\, m_{rj}]$ the column vector corresponding to word *j*.

Similarity matrices: **SR** and **SC** represent the square and symmetrical Row Similarity and Column Similarity matrices of size *rxr* and *cxc* respectively with $sr_{ij} \in [0,1]$, $1 \le i,j \le r$ and $sc_{ij} \in [0,1]$, $1 \le i,j \le c$; $\mathbf{SC_i} = [sc_{i1} \ \ldots \ sc_{ic}]$ (respectively $\mathbf{SR_i} = [sr_{j1} \ \ldots \ sr_{jc}]$) is the similarity vector between the word *i* and all the other words (respectively between the document *j* and the other documents).

Similarity function: function $F_s(.,.)$ is a generic similarity function that takes two elements m_{ij} and m_{kl} of **M** as input and returns a measure of similarity $F_s(m_{ij}, m_{kl}) \in [0,1]$ between these two elements. For brevity, we will also use the shorthand notation $\vec{F}(m_{ij,} \mathbf{m_k})$ for representing a vector that contains the pair-wise similarity $F_s(.,.)$ between the scalar $m_{i,j}$ and each element of the vector $\mathbf{m_k}$.

2.2. Calculating the Co-Similarity

First, we present how to compute the co-similarity matrix **SR** between rows. Usually, the similarity (or distance) measure between two documents $\mathbf{m_i}$ and $\mathbf{m_j}$ is defined as a function, denoted here as $Sim(\mathbf{m_i}, \mathbf{m_j})$, that is the sum of the similarities between shared words.

$$Sim(\mathbf{m_i}, \mathbf{m_j}) = F_s(m_{i1},m_{j1}) + F_s(m_{i2},m_{j2}) \ \ldots \ + F_s(m_{ic},m_{jc}) \quad (1)$$

Other factors may be introduced, for instance to normalize the similarity in the interval [0, 1], but the main idea is always the same: we consider the values between columns having the same index. Now let's suppose we have a matrix **SC** whose entries provide a measure of similarity between the columns (words) of the corpus. Equation (1) can be re-written as follows without changing its meaning if $sc_{i,i} = 1$:

$$Sim(\mathbf{m_i}, \mathbf{m_j}) = F_s(m_{i1},m_{j1}).sc_{11} + \ldots + F_s(m_{ic},m_{jc}).sc_{cc} \quad (2)$$

Here our idea is to *generalize* equation (2) in order to take into account *all the possible pairs of features* (or words) occurring in documents $\mathbf{m_i}$ and $\mathbf{m_j}$. In this way, not only we "capture" the similarity of their common words but also the similarity coming from words that are not directly common in the documents but are shared with some other documents. For each pair of words not directly shared by the documents, we need to take into account the similarity between them as provided by the **SC** matrix. Thus the overall similarity between documents $\mathbf{m_i}$ and $\mathbf{m_j}$ is defined in equation (3) in which the terms in the boxes are those occurring in equation (2):

$$\begin{aligned} &Sim(\mathbf{m_i}, \mathbf{m_j}) = \quad (3)\\ &\boxed{F_s(m_{i1},m_{j1}).sc_{11}} + F_s(m_{i1},m_{j2}).sc_{12} + \ldots + F_s(m_{i1},m_{jc}).sc_{1c} + \\ &F_s(m_{i2},m_{j1}).sc_{21} + \boxed{F_s(m_{i2},m_{j2}).sc_{22}} + \ldots + F_s(m_{i2},m_{jc}).sc_{2c} + \\ &\ldots \\ &F_s(m_{ic},m_{j1}).sc_{c1} + F_s(m_{ic},m_{j2}).sc_{c2} + \ldots + \boxed{F_s(m_{ic},m_{jc}).sc_{cc}} \end{aligned}$$

By using our shorthand notation for representing a row vector, we may rewrite the above formula as follows. The sign "x" means a dot product:

$$Sim(\mathbf{m_i}, \mathbf{m_j}) = \vec{F}(m_{i1},\mathbf{m_j}) \times \mathbf{sc_1} + \ldots + \vec{F}(m_{ic},\mathbf{m_j}) \times \mathbf{sc_c} \quad (4)$$

Conversely, when we express the similarity between two words (or columns) $\mathbf{m^i}$ and $\mathbf{m^j}$ of **M** we use the same approach. Hence:

$$Sim(\mathbf{m^i}, \mathbf{m^j}) = \vec{F}(m_{1i}, \mathbf{m^j}) \times \mathbf{sr_1} + \ldots + \vec{F}(m_{ri}, \mathbf{m^j}) \times \mathbf{sr_r} \quad (5)$$

Thus, in this framework the similarity between any two documents depends on the similarity between the words appearing in these documents (weighted by the matrix **SC**) and reciprocally the similarity of any two words depends on the similarity between the documents in which they occur (weighted by the matrix **SR**).

However, the values obtained in (4) and (5) cannot be used directly to compute the elements sr_{ij} of **SR** and sc_{ij} of **SC**. As previously stated, each element of the matrices **SR** and **SC** must belong to [0, 1] but neither $Sim(\mathbf{m_i}, \mathbf{m_j})$ nor $Sim(\mathbf{m^i}, \mathbf{m^j})$ verify this property, thereby making it necessary to *normalize* these values. Since the maximum value for the function $F_s(,)$ is 1, from (3), it follows that the upper bound of $Sim(\mathbf{m_i}, \mathbf{m_j})$ with $1 \le i,j \le r$, is given by the product of the number of elements of $\mathbf{m_i}$ and $\mathbf{m_j}$ (i.e. the number of words in the *ith* and *jth* documents) denoted by $|\mathbf{m_i}|$ and $|\mathbf{m_j}|$. The reasoning is similar for **SC**. This normalization makes sense to deal with textual datasets since it allows taking into account the actual length of the document and word vectors. Thus, we can now define how to compute the elements sr_{ij} of **SR** and sc_{ij} of **SC**:

$$\forall i,j \in 1..r,\ sr_{i,j} = \frac{Sim(\mathbf{m_i},\mathbf{m_j})}{|\mathbf{m_i}|.|\mathbf{m_j}|} \quad (6)$$

$$\forall i,j \in 1..c,\ sc_{i,j} = \frac{Sim(\mathbf{m^i},\mathbf{m^j})}{|\mathbf{m^i}|.|\mathbf{m^j}|} \quad (7)$$

Equations (6) and (7) are not independent because each one uses values defined by the other. Thus, to evaluate **SR** and **SC** we use iteratively equations (6) and (7) to update their values. The algorithm is detailed in the next section.

3. Efficient implementation of χ-Sim

Classically, the overall complexity of computing a similarity matrix between documents represented by a matrix **M** of size N (assuming it is square matrix), is equal to $O(N^3)$ in terms of the number of calls to $F_s(.,.)$. However, in χ-Sim the complexity is higher due to the fact that to compute **SR** and **SC,** all pairs of documents and all pairs of words must be explored. Thus, we have a complexity of $O(N^4)$ in terms of calls to $F_s(,)$. That is clearly too high to cope efficiently with real datasets.

Fortunately, there are two very efficient ways to deal with this problem, making the algorithm at the same level of complexity, $O(N^3)$, as the other distance measures.

The first method uses the fact that when the values in **M** are discrete (i.e. belong to an enumerated type E containing |E| elements) a whole set of calculations for each pair of documents (or words) in the naïve algorithm is repetitive and can be avoided, thus reducing the complexity to $O(|E|N^3)$. We do not go into details due to space limitation.

The second method is based on the fact that if the similarity function $F_s(m_{ij}, m_{kl})$ can be defined as a simple product (as in the cosine similarity) the system of equations (6) and (7) used to compute **SR** and **SC** can be expressed very simply as a product of three matrices without any loss of generality. The algorithm follows:

1. We initialize the similarity matrices **SR** (documents) and **SC** (words) with the identity matrix **I**, since, at the first iteration, the similarity between a document (resp. word) and itself equals 1 and equals zero in the other cases. We denote these matrices as $\mathbf{SC_0}$ and $\mathbf{SR_0}$.
2. At each iteration *t*, we calculate the new similarity matrix between documents $\mathbf{SR_t}$ by using the similarity matrix between words $\mathbf{SC_{t-1}}$. To normalize the values, we multiply, using the Hadamard[3] product (denoted by "•"), the matrix $\mathbf{SR_t}$ by the matrix **NR** defined as follows: $\forall i,j \in [1,r]$, $nr_{i,j} = 1/|\mathbf{m_i}|.|\mathbf{m_j}|$. We do the same thing for the similarity matrix $\mathbf{SC_t}$. Here are the two relations:

$$\mathbf{SR_t} = \mathbf{M}.\mathbf{SC_{(t-1)}}.\mathbf{M^T} \bullet \mathbf{NR} \quad with\ nr_{i,j} = \frac{1}{|\mathbf{m_i}|.|\mathbf{m_j}|} \quad (8)$$

$$\mathbf{SC_t} = \mathbf{M^T}.\mathbf{SR_{(t-1)}}.\mathbf{M} \bullet \mathbf{NC} \quad with\ nc_{i,j} = \frac{1}{|\mathbf{m^i}|.|\mathbf{m^j}|} \quad (9)$$

3. Set the diagonal of $\mathbf{SR_t}$ and $\mathbf{SC_t}$ to 1
4. Step 2-4 are repeated to update $\mathbf{SR_t}$ and $\mathbf{SC_t}$

The natural question that arises is the number of iterations needed. Fortunately, the notion of iteration has an intuitive meaning: iterating *n* times corresponds to take into account the *n^{th} higher order co-occurrences* between words and documents respectively. As pointed out by [1], it is generally irrelevant to take into account the co-occurrences between documents (or words) above the third or fourth higher order. In the results presented in section 5, we set the number of iterations to 4.

Thus, to update **SR** (or **SC**) we just have to multiply three matrices. Given that the complexity of matrix multiplication[4] is in $O(N^3)$ and that at each iteration we have to compute 4 multiplications, the overall complexity is $O(kN^3)$ with $k \approx 16$ using 4 iterations (corresponding to the search of the 4^{th} high order co-occurrences).

4. Related Work

Most of the co-clustering literature is based on finding sub-matrices that would relate the rows and the columns. Earlier work on co-clustering, called direct clustering, is found in [4] which uses a greedy approach to iteratively splice the matrix to reduce the "within block" variance.

Several matrix decomposition based approaches like Singular Valued Decomposition (SVD) or eigenvector based approaches [2] [12] have been proposed that try to capture the semantic structure of the co-occurrence matrix, hence assigning a low similarity value to otherwise non-similar documents. The Latent Semantic Analysis (LSA) method projects the reduced k-dimensional document singular vectors obtained by SVD and then computes the document-document similarity (typically with the cosine measure). By doing so, LSA implicitly takes into consideration the word similarities in the resulting approximation. In their analysis of the Latent semantic indexing technique, the authors in [1] show that LSA is capable of assigning a similarity value between words that do not necessarily occur together. Another matrix decomposition based approach is the Block Value Decomposition [6] which factorizes the dyadic data matrix into a row coefficient matrix, the block value matrix and column coefficient matrix. The algorithm iteratively clusters rows and columns, implicitly reducing the dimensionality at each iteration.

[3] In the Hadamard product $\mathbf{A}=\mathbf{B}\bullet\mathbf{C}$, the elements $a_{i,k}$ of matrix **A** are defined as: $a_{i,k} = b_{i,k}.\ c_{i,k}$

[4] The complexity of the Hadamard product is $O(N^2)$

Table 2. Different subsets of the NG20 dataset and SMART collection used to evaluate our approach.

Subset	Newsgroups/Abstracts included	# of clusters	# of docs/cluster	# of docs
M2	Talk.politics.mideast, talk.politics.misc	2	250	500
M5	comp.graphics, rec.motorcycles, rec.sport.baseball, sci.space , talk.politics.mideast.	5	100	500
M10	alt.atheism, comp.sys.mac.hardware, misc.forsale, rec.autos, rec.sport.hockey, sci.crypt, sci.electronics, sci.med, sci.space, talk.politics.gun.	10	50	500
NG1	rec.sports.baseball, rec.sports.hockey	2	200	400
NG2	comp.os.ms-windows.misc, comp.windows.x,rec.motorcycles,sci.crypt,sci.space	5	200	1000
NG3	comp.os.ms-windows.misc, comp.windows.x, misc.forsale, rec.motorcycles, sci.crypt, sci.space, talk.politics.mideast, talk.religion.misc	8	200	1600
Classic3	MEDLINE, CRANFIELD, CISI	3	1033,1460,1400	3893

Recently, some other algorithms have focused on information gain and mutual information [13] [3]. Dhillion et al. [3] presented an algorithm that monotonically increases the preserved mutual information by intertwining both row and column clustering at all stages. In their Information-Theoretic Co-Clustering approach, they try to group documents and words maximizing the information gain with the hypothesis that words related to the documents of one category are similar and would increase the preserved mutual information.

Finally, some bipartite graph partitioning algorithms for co-clustering have been proposed [17] [10]. In [17], the algorithm tries to capture the hidden structure using a "relation summary network" by introducing some hidden nodes to capture the relationship between actual nodes. Hence if two nodes are related to each other, the corresponding hidden nodes would also be related.

Our approach is more similar to LSA in that we try to capture 'latent' or higher order similarities. However, instead of computing the latent values implicity based on a lower rank document space model, in our method we use an explicit approach to iteratively take into account the similarity of the words to cluster the documents and vice-versa. This explicit back and forth procedure enables us to find real higher-order relations between pairs of words (or documents).

5. Experiments

To evaluate our algorithm, we perform a series of tests in which we cluster sets of documents using the document similarity matrices **SR** generated by χ-Sim. We compare the results of our approach with those of other classical similarity and co-clustering based algorithms. In these experiments we have used the well-known 20-Newsgroup dataset, which we refer to as the NG20, and the SMART collection dataset which we refer to as the CLASSIC3 dataset [5]. We chose these datasets since they have been widely used as a benchmark for document classification and co-clustering [3] [6] [16] [17], thus allowing us to compare our results with those reported in the literature.

5.1. Preprocessing and methodology

The NG20 dataset consists of approximately 20 000 newsgroup articles that have been collected evenly from 20 different Usenet groups. Many of the newsgroups share similar topics and about 4.5% of the documents are cross-posted making the boundaries between some newsgroups fuzzy. In order to replicate the experimental environment used by the previous authors, we create the subsets of NG20 named M2, M5 and M10 used in [3][6], as well as the subsets NG1, NG2, and NG3 used by [17]. Details about these various subsets are given in table 2.

Each subset has been created with the same procedure. For instance, the M10 dataset has been created by randomly selecting 50 documents (from about 1000) for each of the 10 subtopics, thus forming a subset with 500 documents. For each subset, to be able to evaluate the standard deviation over datasets, we repeat this process 10 times, each time randomly choosing documents from the subtopics to generate 10 different samples of each data subset (M2, M5, M10, NG1, NG2, and NG3). Only documents with two or more words were selected.

The dataset (CLASSIC3) has been used in [3]. The SMART collection consists of MEDLINE (1033 abstracts from medical journals), CISI (1460 abstracts from information retrieval papers) and CRANFIELD (1400 abstracts from the field of aerodynamic systems). Following [3], we do the same preprocessing steps ignoring subject lines, removing stop words and selecting the top 2000 words based on mutual information for all the data sets.

With these seven benchmarks, we compared our co-similarity based measure (χ-Sim) with two similarity measures and three co-clustering methods as follows:

- ***Cosine***: a very classical similarity measure [11]
- ***LSA***: Latent Semantic Analysis as described in [1]
- ***ITCC***: Information theoretic co-clustering algorithm [3];
- ***BVD***: based on matrix block value decomposition [6];
- ***RSN***: k-partite graph partitioning algorithm [17].

We consider Cosine as a baseline for comparison since it does not capture higher order similarities.

Table 3. Precision(Pr) and Normalized mutual Information(NMI) along with standard deviation for the various subsets of the newsgroup dataset (NG20) and Classic3 dataset. Results are rounded to 2 digits after the decimal.

	Runtime		χ-Sim	Cosine	LSA	ITCC[3]	BVD[6]	RSN[17]
M2	*12.01s*	Best Pr	**0.96**	0.72	0.94	0.92	0.95	-
		$Pr_{(avg)}$	**0.93 ± 0.01**	0.62 ± 0.04	0.78 ± 0.15	0.71 ± 0.15		
		$NMI_{(avg)}$	**0.65 ± 0.05**	0.15 ± 0.05	0.37 ± 0.23	0.21 ± 0.21		
M5	*10.93s*	Best Pr	**0.98**	0.76	0.95	0.56	0.93	-
		$Pr_{(avg)}$	**0.94 ± 0.04**	0.6 ± 0.08	0.82 ± 0.09	0.48 ± 0.06		
		$NMI_{(avg)}$	**0.85 ± 0.07**	0.55 ± 0.11	0.65 ± 0.14	0.27 ± 0.09		
M10	*13.53s*	Best Pr	**0.80**	0.55	0.67	0.33	0.67	
		$Pr_{(avg)}$	**0.75 ± 0.04**	0.45 ± 0.07	0.54 ± 0.09	0.28 ± 0.04		-
		$NMI_{(avg)}$	**0.65 + 0.05**	0.43 ± 0.07	0.45 ± 0.08	0.16 ± 0.04		
NG1	*6.60s*	Best Pr	**1**	0.97	0.98	0.83		
		$Pr_{(avg)}$	**1 ± 0**	0.90 ± 0.11	0.95 ± 0.02	0.67 ± 0.12		
		$NMI_{(avg)}$	**1 ± 0**	0.62 ± 0.20	0.75 ± 0.08	0.13 ± 0.13	-	0.64 ± 0.16
NG2	*14.56s*	Best Pr	**0.94**	0.72	0.87	0.72		
		$Pr_{(avg)}$	**0.93 ± 0.01**	0.60 ± 0.08	0.82 ± 0.03	0.57 ± 0.08		
		$NMI_{(avg)}$	**0.81 ± 0.04**	0.51 ± 0.07	0.64 ± 0.03	0.35 ± 0.09	-	0.75 ± 0.07
NG3	*20.20s*	Best Pr	**0.91**	0.62	0.80	0.65		
		$Pr_{(avg)}$	**0.85 ± 0.06**	0.60 ± 0.04	0.72 ± 0.05	0.55 ± 0.06		
		$NMI_{(avg)}$	**0.77 + 0.02**	0.55 ± 0.03	0.59 ± 0.03	0.48 ± 0.05	-	0.70 ± 0.04
Classic3	*164.38s*	Pr	**0.99**	0.89	0.98	0.97	-	-
		NMI	**0.96**	0.63	0.93	0.90		

For χ-Sim, Cosine and LSA, in order to create the clusters, we used an *Agglomerative Hierarchical Clustering* (AHC) method on the similarity matrices with Ward's linkage. We did not use a more classical *k-means* approach in order to avoid the cluster stability problems. χ-Sim is implemented in C++, and we used MatLab to calculate cosine distances, SVD (for LSA) as well as to perform AHC. For ITCC, we used the implementation provided by the authors and set the number of word clusters as the best numbers reported in [3]. For BVD and RSN, as we don't have a running implementation, we quote the best values from [6] and [17] respectively. We set the number of document clusters to the actual document categories for all the algorithms.

Several measures have been proposed to compare the accuracy and quality of the clustering results based on the confusion matrix whose elements (*i*, *j*) corresponds to the number of documents with actual class *i* and predicted. class *j*. Here we used two widely used indices: the micro-averaged Precision (Pr) [3] for comparing the accuracy of the document classification and the Normalized Mutual Information (NMI) [18] to compare the quality of the resulting clusters.

5.2. Results

Unless otherwise stated, we run our algorithm on each of the 10 samples of the each of the different subsets of the NG20 newsgroup (see table 2) and report the average value and the standard deviation. Moreover, in order to find the optimal values of the parameters of LSA and ITCC methods, we proceeded as follows:

For LSA, we ran two separate experiments, in the first one we use the frequency measure in the co-occurrence data matrix and in the second one the "tf-idf". We tested the LSA algorithm iteratively keeping the k-highest singular values from k=10..200 by steps of 10. We report the *best* value as the one which provides the highest accuracy average amongst all the tested values of k for all the samples of a dataset.

For the Information theoretic co-clustering (ITCC) algorithm, we ran the algorithm using the different numbers of word clusters, as suggested in [3], for each dataset. Taking into account that the results of ITCC are based on random initialization, we repeated these experiments 3 times in each case and reported the best result.

We provide a summary of all the results in table 3. We provide both the micro-averaged precision value (Pr) as well as the NMI score to facilitate the comparison of results with [6] and [17] respectively. The column named "Runtime" reports the average runtime in seconds needed by χ-Sim to compute the two matrices, **SR** and **SC,** with 4 iterations.

The experimental results show that χ-Sim significantly outperforms the other tested methods on both Pr and NMI scores over all the datasets. It is interesting to observe the behavior of χ-Sim on various datasets (M2, M5, M10) when we keep the total number of documents constant while decreasing the number of documents per cluster and

increasing the cluster numbers (see table 2). As the number of clusters increases, the inherent structure of the dataset gets more complicated and naturally one would expect a decrease of the accuracy of all the algorithms. However, we note that the performance of our algorithm decreases much slowly than other techniques like LSA, ITCC or BVD. Even when the number of documents per cluster is just 50 (in the case of M10), χ-Sim is able to retrieve the clusters more accurately obtaining a micro-averaged precision of up to 0.798 bettere than the best results with LSA and BVD at 0.668 and 0.67 respectively.

It is also interesting to note from the experiments in table 3 that χ-Sim seems more consistent on both the Pr and NMI scores across the 10 randomly generated subsets of each dataset, showing a maximum standard deviation of 0.07. In particular, it is more consistent across datasets with smaller numbers of clusters (M2, M5, NG1 and NG2), where we observe a greater variation in the results of Cosine, LSA and ITCC.

5.3. Co-clustering task

In the previous experiments of document clustering, we only used the **SR** matrix which inherently incorporates the similarity between words (as defined by **SC**). In this section, we present a way to use the co-similarities contained in both matrices **SR** and **SC** to co-cluster the words and documents.

We proceed as follows: we cluster independently the rows and the columns using again AHC with Ward's linkage on the matrices **SR** and **SC** respectively. Suppose the words are clustered into m classes and the documents are clustered into n classes, thus we create a m by n *density* matrix **D** whose entries correspond to the number of occurrences of words of class j in documents of class i ($i \in 1..m$, $j \in 1..n$). Using the Hungarian algorithm [19], we associate each word cluster to a document cluster optimizing the sum of the assignments.

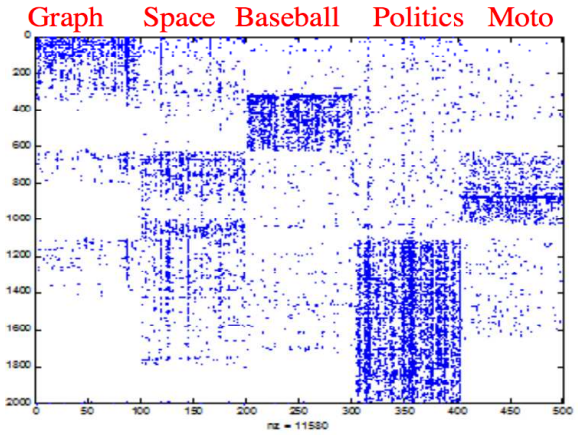

Fig. 1. Sparsity map of a M5 dataset word-document co-occurrence matrix.

Figure 1 shows the matrix plots using MatLab of the M5 subset after co-clustering. We observe that χ-Sim was easily able to find word clusters, that show a good association to the document clusters.

6. Conclusion

In this paper, we proposed a new similarity measure for the co-clustering task. By iteratively taking into account the similarity between words to calculate the similarity between documents and vice-versa, we are able to cluster together documents that have "similar" concepts based on their sharing "similar" words. This also allows us to use any classical similarity-based clustering method to cluster the data while still benefiting from the advantages of co-clustering all along.

We show that our co-similarity based approach not only performs significantly better than a classical distance measure (cosine) but it also surpasses other co-clustering and graph partitioning based algorithms for the task of document classification. In contrast to [3] [6], our algorithm does not need to cluster the words (columns) for clustering the documents, thus avoiding the need to know the number of word clusters. Documents are clustered irrespective of the number of word clusters.

It is worth noting, that clustering using our co-similarity measure performs better than LSA, which is also a method able to capture higher order co-occurrences between words and documents. However, in our experiments, in order to accurately compare our work with previous one, we used rather small datasets and LSA is generally used for larger ones. In some ongoing experiments (not cited in this paper) we observed that when the number of documents increases (for instance, when we test M10 with all the 10 000 documents contained in the 10 newsgroups of NG20), the micro-averaged precision value of LSA and χ-Sim increases to 75% and 82% respectively, showing that the difference between the two methods becomes smaller. In practice, our method seems to be more sensitive than LSA in capturing the higher order co-occurrences in small and medium sized corpora.

Acknowledgement : we would like to thank Mirta B. Gordon for her invaluable comments and advice concerning the current work.

References

[1] Kontostathis A., Pottenger W.M. A framework for understanding LSI performance. Information Processing and Management. Volume 42, number 1, 2006, pp 56-73.

[2] Deerwester S. C., S.T. Dumais, T. K. Landaeur, G.W. Furnas and R.A. Harshman. Indexing by latent semantic analysis, Journal of the American Society of Information Science,41(6):391-407. 1990

[3] Dhillon I. S., Mallela S. and Modha D. S. Information Theoretic Co-clustering. Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Aug 24-27, 2003, pp 89-98.

[4] Hartigan J.A. Direct clustering of a data matrix. Journal of American Statistical Association, 67(337):123-129, 1972.

[5] Newsgroup dataset and SMART dataset URL's:
http://people.csail.mit.edu/jrennie/20Newsgroups/
ftp://ftp.cs.cornell.edu/pub/smart

[6] Long B., Zhongfei Z. and Yu P. S. Co-clustering by Block Value Decomposition. In SIGKDD, aug 21-24, 2005.

[7] Salton G., The SMART Retrieval System – Experiments in Automatic Document Processing, Prentice Hall, Inc., Englewood Cliffs, NJ, 1971.

[8] Sara C. Madeira and Arlindo L. Oliveira. Biclustering algorithms for biological data analysis: A survey,INESC-ID technical report, 2004.

[9] Liu, X., Gong, Y., Xu, W., and Zhu, S. Document clustering with cluster refinement and model selection capabilities. In Proceeding of ACM SIGIR Conference on Research and Development in information retrieval, 2002.

[10] Dhillon, S.I., Co-clustering documents and words using bipartite spectral graph partitioning, KDD San Francisco,California, 2001.

[11] Speer N., C. Spieth, A. Zell. A Memetic Clustering Algorithm for the Funct-ional Partition of Genes Based on the Gene Ontology, Proceedings of the IEEE Symposium on Computational Intelligence in Bio-informatics and Computational Biology, 2004.

[12] Susan T. Dumais. LSA and Information Retrieval: Getting back to Basics. Handbook of Latent Semantic Analysis, Lawrench Erlabaum Associates, 2007.

[13] Tishby N., F. Pereira and W. Bialek. The information bottleneck method, In Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing, 1999, pp 368-377.

[14] Cheng Y. and Church G. M. Bi-clustering of expression data, in ICBM, 2000, p p 93-103.

[15] Manjeet R., Ming D. and Farshad F. Bipartite isometric graph partitioning for data co-clustering. Journal of Data Mining and Knowledge Discovery. To appear, 2008.

[16] Zhang Daokiang, Zhi-Hua Zhou and Songcan Chen. Semi-supervised dimensionality reduction, Proceedings of the SIAM International Conference on Data Mining, 2007.

[17] Long Bo, Wu Xiaoyun, Zhang Zhongfei and Yu Phillip: Unsupervised Learning on K- partite Graphs, Proceedings of SIGKDD August 20-23, 2006.

[18] Banerjee, A., Ghosh,J.: Frequency sensitive competitive learning for clustering for clustering of high dimensional hyperspheres, Proc. IEEE joint conf. on neural networks, 2002, pp 1590-1595.

[19] Munkres, J.: Algorithms for the assignment and transportation Problems. J. Siam 5, 1957, pp 32-38.