International Conference on Control, Engineering & Information Technology (CEIT'13)
Proceedings Engineering & Technology - Vol.1, pp. 121-126, 2013

Parallel architectures for fuzzy triadic similarity learning

Sonia Alouane Ksouri[1], Minyar Sassi Hidri[2], Kamel Barkaoui[3]
[1,2]Université Tunis El Manar
[1,2]Ecole Nationale d'Ingénieurs de Tunis
Laboratoire Signal, Images et Technologies de l'Information
BP. 37, Le Belvèdère 1002, Tunis, Tunisia
[3]CEDRIC-CNAM
Rue Saint-Martin Paris 75003, France
{[1,2]sonia.alounae,minyar.sassi}@enit.rnu.tn,[3]kamel.barkaoui@cnam.fr

***Abstract*—In a context of document co-clustering, we define a new similarity measure which iteratively computes similarity while combining fuzzy sets in a three-partite graph. The fuzzy triadic similarity (FT-Sim) model can deal with uncertainty offers by the fuzzy sets. Moreover, with the development of the Web and the high availability of storage spaces, more and more documents become accessible. Documents can be provided from multiple sites and make similarity computation an expensive processing. This problem motivated us to use parallel computing. In this paper, we introduce parallel architectures which are able to treat large and multi-source data sets by a sequential, a merging or a splitting-based process. Then, we proceed to a local and a central (or global) computing using the basic FT-Sim measure. The idea behind these architectures is to reduce both time and space complexities thanks to parallel computation.**

Keywords: Document co-clustering, Three-partite graph, Fuzzy sets, Parallel computing.

I. Introduction

Nowadays information on the internet is exploding exponentially through time, and approximately 80% are stored in the form of text. So text mining has been a very hot topic. One particular research area is document clustering, which is a major topic in the Information Retrieval community. It allows to efficiently capture high-order similarities between objects described by rows and columns of a data matrix. In the domain of text clustering, a document is described as a set of words.

The relationship between documents and words allows for exploitation of the relationship between groups of words that occur mostly in a group of documents.

In [1], a co-similarity measure has been proposed, called X-Sim [1] which builds on the idea of iteratively generating the similarity matrices between documents and words, each of them built on the basis of the other. This measure works well for unsupervised document clustering.

However, in recent researches, the sentence has been considered as a more informative feature term for improving the effectiveness of document clustering [2]. While considering three levels Documents $\times$ Sentences $\times$ Words to represent the data set, we are able to deal with a dependency between Documents-Sentences, as also between Sentences-Words and, by deduction, between Documents-Words.

Another important aspect in co-clustering is the weight computing. A weighted value may be assigned as a link from a document to a word (or sentence) indicating the presence of the word (sentence) in that document. The 0/1 encoding denotes the presence or absence of an object in a given document.

Different weighting schemes such as the tf-idf [3] may be incorporated to better represent the importance of words in the corpus, but it has spawned the view that classical probability theory is unable to deal with uncertainties in natural language and machine learning.

So, we proceed to a fuzzification control process which converts crisp similarities to fuzzy ones. The conversion to fuzzy values is represented by the membership functions [4]. They allow a graphical representation of a fuzzy set [5]. These fuzzy similarity matrices are used to calculate fuzzy similarity between documents, sentences and words in a triadic computing called FT-Sim (Fuzzy Triadic Similarity).

Moreover, with the development of the Web and the high availability of the storage spaces, more and more documents become accessible. Data can be provided from multiple sites and can be seen as a collection of matrices. By separately processing these matrices, we get a huge loss of information.

Several extensions to the co-clustering methods have been proposed to deal with such multi-view data. Some works aim at combining multiple similarity matrices to perform a given learning task [6], [7]. The idea being to build clusters from multiple similarity matrices computed along different views.

Multi-view co-clustering such as MV-Sim [8] architecture, based on X-Sim measure [1] deals with the problem of learning co-similarities from a collection of matrices describing interrelated types of objects. It was proved that this architecture provides some interesting properties both in terms of convergence and scalability and it allows an efficient parallelization of the process.

For this, we provide parallel architectures for FT-Sim to tackle the problem of learning similarities from a collection of matrices. For multi-source or large matrices, we propose different parallel architectures in which each FT-Sim is the basic component or node we will use to deal with multiple

matrices.

Thus, we consider a model in which data sets are distributed into N sites (or relation matrices). They describe the connections between documents for each local data set.

Our goal is then to compute a fuzzy Documents $\times$ Documents matrix $\widetilde{D}_2^{(i)}$ for each site i $(i = 1..N)$ trying to take into account all the representative information expressed in the relations.

To combine multiple occurrences of FT-Sim, we propose sequential, merging and splitting based parallel architectures.

The rest of the paper is organized as follows: in section 2 we highlights backgrounds related to similarity measures in a multi-view data sets. In section 3 we provide our fuzzy triadic similarity measure. In section 4 we present the three proposed architectures allowing parallel computing for co-clustering. Section 5 concludes the paper and gives indications of some future work.

II. Dealing with Multi-view Data sets

Most of the existing clustering methods focus on data sets described by a unique data matrix, which can either be a matrix which describes objects by their characteristics, or a relation matrix that describes the intensity of the relation between instances of two types of objects, such as a Documents $\times$ Words matrix. In the latter case, both types of objects can be clustered ; methods dealing with this task are referred to as co-clustering approaches and have been extensively studied.

However, in many applications, data sets involving more than two types of interacting objects, or simply related, are also frequent. A simple way to represent such data sets is to use as many matrices as there are relations between the objects. Then, one could use classical co-clustering methods to separately cluster the objects occurring in the different matrices but, in this way, interactions between objects are not taken into account, thus leading to a loss of information. Therefore, handling the views together, referenced as the multi-view clustering task, is an interesting challenge in the learning domain to resolve limits of classical clustering.

Many extensions to the clustering methods have been proposed to deal with multi-view data. In [9], they describe an extension of k-means (MVKM) and of EM algorithms using multi-view model. In [6] and [7], the authors build clusters from multiple similarity matrices computed along different views. In [10], a co-clustering system called MVSC has been proposed. It permits a multi-view spectral clustering while using the co-training that has been widely used in semi-supervised learning problems. The general idea is to learn the clustering in one view and use it to label the data in an other view so as to modify the graph structure (similarity matrix).

Closer to our approach, some works aim at combining multiple similarity matrices to perform a given learning task. The MVSim architecture [8] which is an extension of the X-Sim algorithm [1], adapts the previous algorithm to the multi-view context. It computes simultaneously the co-similarity matrix for each of N different kinds of objects T_i described by M relation matrices. The basic idea is to create a learning network isomorphic to these data sets structures. It was shown that it is possible to use this architecture to efficiently compute co-similarities on large data sets by splitting a data matrix into smaller ones.

III. FT-Sim: Fuzzy Triadic Similarity

Sentence-based analysis means that the similarity between documents should be based on matching sentences rather than on matching single words only. Sentences contain more information than single words (information regarding proximity and order of words) and have a higher descriptive power[11] [12][13]. Thus a document must be broken into a set of sentences, and a sentence is broken into a set of words. We focus on how to combine the advantages of two representation models in document co-clustering.

To represent our textual data set, two representations have been proposed: the collection of matrices and the k-partite graph [14]. In the first, each matrix describes a view on the data. In the second, a graph is said to be k-partite when the nodes are partitioned into k subsets with the condition than no two nodes of the same subset are adjacent. Thus in the k-partite graph paradigm [14], a given subset of nodes contains the instances of one type of objects, and a link between two nodes of different subsets represents the relation between these two nodes.

To explain our model we consider matrices to represent the data sets and we use a three-partite graph representation of the data matrices with three relations linking to explain our model.

From a functional point of view, the proposed FT-Sim model can be represented in the following way as shown in figure 1, where SD and WS are two data matrices representing a corpus and describing the connection between Documents/Sentences and Sentences/Words, brought by the three-partite graph [15].

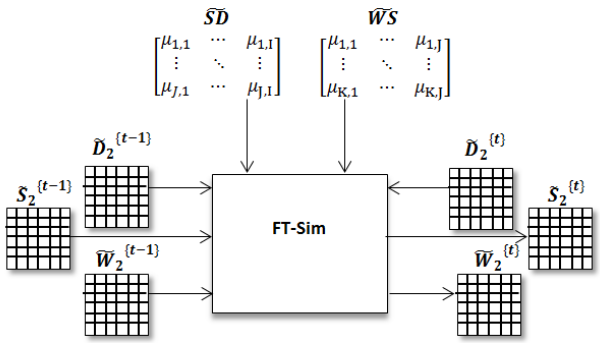

Fig. 1. Functional diagram of FT-Sim.

After the generation of SD and WS matrices, we proceed to a fuzzification process. It converts crisp values to fuzzy ones. The conversion to fuzzy values is represented by the membership functions [4]. They allow a graphical representation of a fuzzy set [5]. There are various methods to assign membership values or the membership functions to fuzzy variables. We mention essentially the triangular and trapezoidal ones. The

second form is the most suitable one for modeling fuzzy Sentences $\times$ Documents and Words $\times$ Sentences similarities.

For each document, we define a fuzzy membership function through a linear transformation between the lower bound value L_i, a membership of 0, to the upper bound value U_i, which is assigned a membership of 1. This function is used because smaller values linearly increase in membership to the larger values for a positive slope and opposite for a negative slope.

The following formulas show the fuzzy linear membership functions for $\widetilde{SD}_i$ and $\widetilde{WS}_j$.

$$\widetilde{SD}_i = [\mu]_{ji} = \begin{cases} 1, & if \;\; SD_{ji} \geq L_i \\ \frac{SD_{ji}-U_i}{U_i-L_i}, & if \;\; L_i < SD_{ji} < U_i \\ 0, & if \;\; SD_{ji} \leq L_i \end{cases} \quad (1)$$

and

$$\widetilde{WS}_j = [\mu]_{kj} = \begin{cases} 1, & if \;\; WS_{kj} \geq L_i \\ \frac{WS_{kj}-U_i}{U_i-L_i}, & if \;\; L_i < WS_{kj} < U_i \\ 0, & if \;\; WS_{kj} \leq L_i \end{cases} \quad (2)$$

Before proceeding to fuzzy triadic computing, we must initialize Documents $\times$ Documents, Sentences $\times$ Sentences and Words $\times$ Words fuzzy matrices with the identity ones denoted as $\tilde{D}_2^{(0)}$, $\tilde{S}_2^{(0)}$ and $\tilde{W}_2^{(0)}$. The similarity between the same documents (resp. sentences and words) have the value equal to 1. All others values are initialized with zero. $\tilde{D}_2^{(t)}$ is as follows:

$$\tilde{D}_2^{(t)} = [\mu]_{lm}^{(t)} = \begin{array}{c} \begin{matrix} D_1 & \dots & D_m \end{matrix} \\ \begin{bmatrix} 1 & \dots & \mu_{1m}^{(t)} \\ \vdots & \ddots & \vdots \\ \mu_{l1}^{(t)} & \dots & 1 \end{bmatrix} \end{array} \begin{matrix} D_1 \\ \\ D_l \end{matrix} \quad (3)$$

where $\mu_{lm}^{(t)}$ ($l = 1..I$, $m = 1..I$ is the membership degree of the l^{th} document according the m^{th} one. Similarly, we determine the $\tilde{S}_2^{(t)}$ and $\tilde{W}_2^{(t)}$.

After initializing $\tilde{D}_2^{(t)}$, we calculate the new matrix $\tilde{D}_2^{(t)}$ which represents fuzzy similarities between documents while using $\tilde{S}_2^{(t-1)}$ and $\widetilde{SD}$.

Usually, the similarity measure between two documents D_l and D_m is defined as a function that is the sum of the similarities between shared sentences.

Our idea is to generalize this function in order to take into account the intersection between all the possible pairs of sentences occurring in documents D_l and D_m. In this way, not only can we capture the fuzzy similarity of their common sentences but also the fuzzy ones coming from sentences that are not directly common in the documents but are shared with some other documents.For each pair of sentences not directly shared by the documents, we need to take into account the fuzzy similarity between them as provided by $\tilde{S}_2^{(t-1)}$.

Since we work with fuzzy matrices formed by membership degrees, we should certainly be applied in accordance with the operators for fuzzy sets, especially the intersection and union. Thus, $\mu_{l,m}^{(t)}$, except the case $l = m$, can be formulated as follows:

$$\mu_{l,m}^{(t)} = \sum_{i=1}^{J}\sum_{j=1}^{J} \min(\mu_{il}, \mu_{jm}) * \mu_{ij}^{\tilde{S_2}^{(t-1)}} \quad (4)$$

As we have shown for $\tilde{D}_2^{(t)}$ computing, we generalize fuzzy similarities in order to take into account the intersection between all the possible pairs of words occurring in sentences S_l and S_m. In this way, not only do we capture the fuzzy similarity of their common words but also the fuzzy ones coming from words that are not directly common in the sentences but are shared with some other sentences.

For each pair of words not directly shared by the sentences, we need to take into account the fuzzy similarity between them as provided by $\tilde{W}_2^{(t-1)}$. The overall fuzzy similarity between documents S_l and S_m is defined in the following equation:

$$\mu_{l,m}^{(t)} = Min[\sum_{i=1}^{I}\sum_{j=1}^{I} \min(\mu_{il}, \mu_{jm}) * \mu_{ij}^{\tilde{D_2}^{(t-1)}}, \quad (5)$$

$$\sum_{i=1}^{K}\sum_{j=1}^{K} \min(\mu_{il}, \mu_{jm}) * \mu_{ij}^{\tilde{W_2}^{(t-1)}}]$$

Similarly, for each pair of words not directly shared by the sentences, we need to take into account the fuzzy similarity between them as provided by $\widetilde{W}_2^{(t-1)}$. The overall fuzzy similarity between documents W_l and W_m is defined in the following equation:

$$\mu_{l,m}^{(t)} = \sum_{i=1}^{J}\sum_{j=1}^{J} \min(\mu_{il}, \mu_{jm}) * \mu_{ij}^{\tilde{S_2}^{(t-1)}} \quad (6)$$

IV. Parallel FT-Sim

For multi-source or large data sets, we propose different parallel architectures in which each FT-Sim is the basic component or site we will use to deal with multiple matrices.

Thus, we consider a model in which the data sets are composed of N relation matrices $\widetilde{SD}^{(i)}$ and $\widetilde{WS}^{(i)}$ ($i = 1..N$). They describe the connections between documents for each local data set. Our goal is then to compute a fuzzy matrix $\widetilde{D}_2^{(i)}$ for each data set trying to take into account all the information expressed in the relations.

To combine multiple occurrences of FT-Sim, we can adopt three different architectures: a sequential, a merging or a splitting based one.

A. Sequential-based parallel architecture

In this first model, an instance of $FT - Sim^{(i)}$ is associated to each local site i. Each site is represented by the relation matrice corresponding to the similarity between sentences/documents $\widetilde{SD}^{(i)}$ and words/sentences $\widetilde{WS}^{(i)}$ for $(i = 1..N)$. N being the number of data sources. This instance is denoted $FT - Sim^{(i)}$. Figure 2 shows the sequential-based parallel architecture.

As shown in figure 2, we assume a link between each $FT - Sim^{(i)}$ and the following one. Then it computes the similarity matrices from the data matrices of the first data set $\widetilde{SD}^{(1)}$ and $\widetilde{WS}^{(1)}$, and uses the resulting document similarity matrix to initialize the next site.

The document similarity issue of the $1^{(st)}$ data-set $\widetilde{D}_2^{(1)}$ is used to initialize the next document similarity denoted by $(\widetilde{D}_2^{(2)})^{\{0\}}$ (the second document similarity matrice at iteration

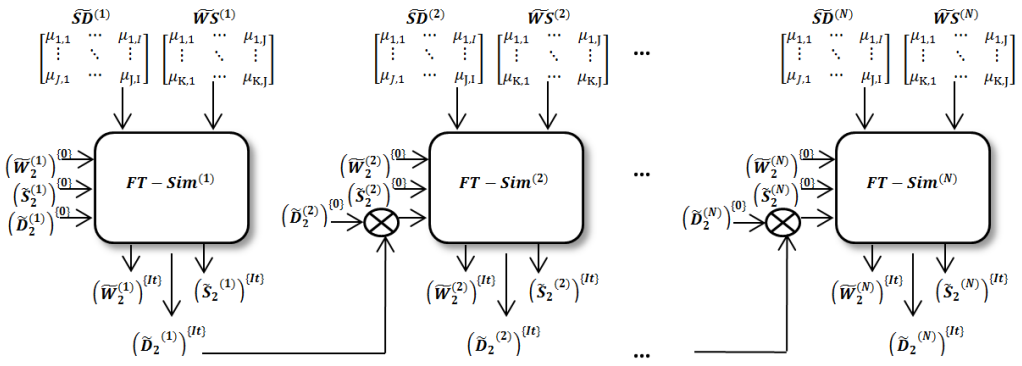

Fig. 2. Sequential-based parallel architecture.

0). The initialization function presented in algorithm 1 is then run with a second $\widetilde{SD}^{(2)}$ and $\widetilde{WS}^{(2)}$ matrices etc.

Algorithm 1 Initialization function

Require: $\widetilde{D}_2^{(i)}$
Ensure: $\tilde{D}_2$
1: $I \leftarrow$ Compute the number of documents in $\widetilde{D}_2^{(i)}$ and $(\widetilde{D}_2^{(i+1)})^{\{0\}}$
2: Let $\tilde{D}_2 = [\mu_{l,m}]$ $(l = 1..I$ and $m = 1..I)$
3: $\tilde{D}_2 \leftarrow Identity$
4: **for** $l = 1..I^{(i)}$ **do**
5: **for** $m = 1..I^{(i)}$ **do**
6: $\mu_{l,m} \leftarrow \mu_{l,m}^{(i)}$
7: **end for**
8: **end for**

The natural question that arises is: how to initialise $(\widetilde{D}_2^{(i+1)})^{\{0\}}$ with $\widetilde{D}_2^{(i)}$?

In the beginning, $(\widetilde{D}_2^{(i+1)})^{\{0\}}$ must contain all documents existing in the i^{th} and the $(i+1)^{th}$ data sets. They are initialized as an identity matrix denoted by $Identity$.

After that, the obtained $(\widetilde{D}_2^{(i+1)})^{\{0\}}$ is updated with the similarities in $\widetilde{D}_2^{(i)}$. The different steps for the sequential-based parallel process are presented in algorithme 2.

Algorithm 2 Sequential-based algorithm

Require: $\widetilde{D}_2^{(0)}$, N
Ensure: $\tilde{D}_2$
1: Execute $FT - Sim^{(1)}$ with $\tilde{D}_2^{(0)}$
2: **for** $i = 1..N$ **do**
3: Initializing with $\widetilde{D}_2^{(i)}$
4: Execute $FT - Sim^{(i+1)}$ with $\widetilde{D}_2^{(i)}$
5: **end for**

Each $FT - Sim^{(i)}$ is connected to the inputs of the following one which creates a chain. In that way, the instances are sequentially run in a static or dynamic order and the similarity matrices $\widetilde{D}_2^{(i)}$ are progressively updated.

The problem with this model is that the order matters. How do we choose the order of the matrices? How many iterations do we perform for each local $FT - Sim^{(i)}$?

Thus, without any prior knowledge about the relative interest of the relation matrices and the number of iterations for each local computing, this model seems difficult to optimize.

B. Merging-based parallel architecture

In the second model, we propose to compute the similarity matrices from several sites and merge them before performing the co-clustering algorithm on it. Figure 3 shows the merging-based parallel architecture.

In this topology, all local $FT - Sim^{(i)}$ instances $(i = 1..N)$ are run in parallel, then the similarity matrices $\widetilde{D}_2^{(i)}$ are simultaneously updated with an aggregation function. This policy offers the benefit that all the instances of $FT - Sim^{(i)}$ have the same influence.

The aggregation function takes N matrices $(\widetilde{D}_2^{(1)})^{\{t\}}$, $(\widetilde{D}_2^{(2)})^{\{t\}}$,..,$(\widetilde{D}_2^{(N)})^{\{t\}}$ issue from each data source i for a given iteration t. Two rules are adopted:

Rule 1: If a given document does not appear in a single site then we assign its corresponding similarity measures directly in $\tilde{D}_2$.

Rule 2: If a particular document appears in several different sites, we assign the minimum of all similarity measures relevant to this document to $\tilde{D}_2$ without taking into account the value of 0.

The different steps of aggregation computing are presented in algorithm 3.

Algorithm 3 Merging Function

Require: Collection of N matrices $\{(\widetilde{D}_2^{(1)})^{\{t\}}, (\widetilde{D}_2^{(2)})^{\{t\}}, .., (\widetilde{D}_2^{(N)})^{\{t\}}\}$
Ensure: $\widetilde{D}_2$
1: $I \leftarrow$ Number of documents in $\{(\widetilde{D}_2^{(1)})^{\{t\}}, (\widetilde{D}_2^{(2)})^{\{t\}}, .., (\widetilde{D}_2^{(N)})^{\{t\}}\}$
2: Let $\widetilde{D}_2 = [\mu_{l,m}]$ $(l = 1..I$ and $m = 1..I)$
3: $\widetilde{D}_2 \leftarrow Identity$
4: **for** Each document D_l of $\widetilde{D}_2$ **do**
5: **if** D_l Appear in only one data set s **then**
6: $\mu_{l,*} \leftarrow \mu_{l,*}^{(s)}$
7: **else**
8: $\mu_{l,*} \leftarrow \min(\text{ All}\mu_{l,*}^{(i)})\ i \in \{$ sites where D_l appear$\}$ with $\mu_{l,*}^{(i)} \neq 0$
9: **end if**
10: **end for**

So, for a given iteration t, each instance $FT - Sim^{(i)}$ produces its own similarity matrix $(\widetilde{D}_2^{(i)})^{\{t\}}$. We thus get a set of output similarity matrices $\{(\widetilde{D}_2^{(1)})^{\{t\}}, (\widetilde{D}_2^{(2)})^{\{t\}}$,..,$(\widetilde{D}_2^{(N)})^{\{t\}}\}$ the cardinal of which being equal to the number of data-sets related to N.

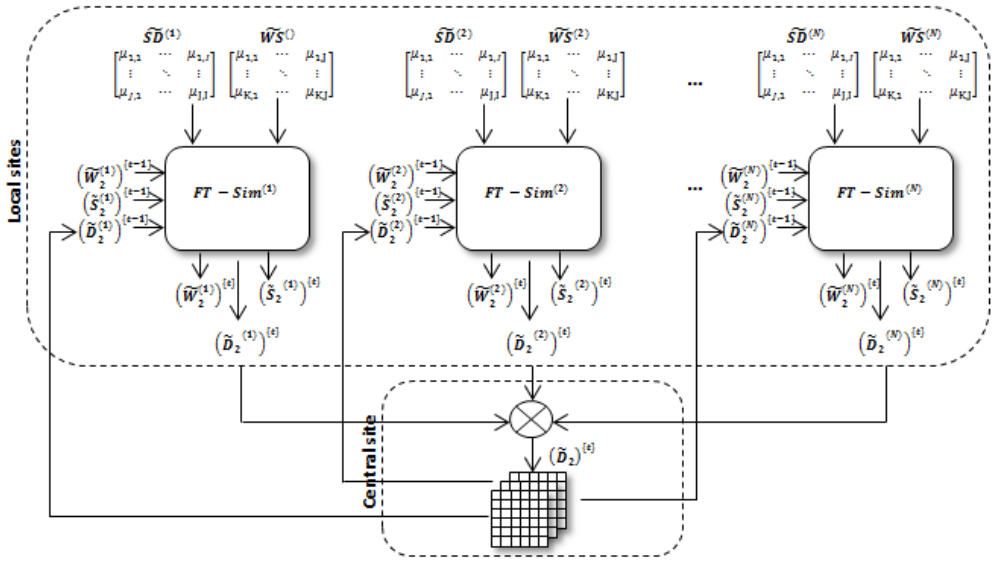

Fig. 3. Merging-based parallel architecture.

Therefore, we use the aggregation function denoted by $\bigotimes$ and developed in the merging based function to compute a consensus similarity matrix merging all of the $\{(\widetilde{D}_2^{(1)})^{\{t\}}$, $(\widetilde{D}_2^{(2)})^{\{t\}}$,..,$(\widetilde{D}_2^{(N)})^{\{t\}}\}$ with the current matrix $\widetilde{D}_2^{\{t\}}$.

In turn, this resulting consensus matrix is connected to the inputs of all the $FT - Sim^{(i)}$ instances, to be taken into account in the $t+1^{th}$ iteration, thus creating feedback loops allowing the system to spread the knowledge provided by each $(\widetilde{D}_2^{(i)})^{\{t\}}$ within the network. The different steps for the merging-based parallel process are presented in algorithme 4.

Algorithm 4 Parallel merging-based Algorithm

Require: collection of matrices $\widetilde{SD}^{(i)}$, $\widetilde{WS}^{(i)}$ $(i = 1..N)$, T
Ensure: $\widetilde{D}_2$
1: **for all** i **do**
2: $(\widetilde{D}_2^{(i)})^{\{0\}} \leftarrow Identity$
3: $(\widetilde{S}_2^{(i)})^{\{0\}} \leftarrow Identity$
4: $(\widetilde{W}_2^{(i)})^{\{0\}} \leftarrow Identity$
5: **for** $i = 1..T$ **do**
6: Execute every $FT - Sim^{(i)}$ with $\widetilde{SD}^{(i)}$, $\widetilde{WS}^{(i)}$ and $t = 1$
7: $(\widetilde{D}_2)^{\{t\}} \leftarrow$ Merging with all $(\widetilde{D}_2^{[i)})^{\{t\}}$
8: Update each $(\widetilde{D}_2^{[i)})^{\{t\}}$
9: **end for**
10: **end for**

The complexity of this architecture is obviously related to that of the $FT - Sim^{(i)}$ algorithm. In the parallel merging-based architecture, as each instance of $FT - Sim^{(i)}$ can run on an independent core, the method can easily be parallelized, thus keeping the global complexity unchanged (considering the number of iterations as a constant factor). So, the complexity of the merging function can be ignored.

C. Splitting-based parallel architecture

In this section we present a generated model that can use previous architectures to efficiently compute FT-Sim on large data sets by splitting a data matrix into smaller ones. Figure 4 shows the splitting-based parallel architecture.

In order to reduce the complexity of a problem of treating huge data sets, it is possible to split a given data matrix into a collection of smaller ones, each sub-matrix becoming a component of our network and processed as a separate view.

We have to evaluate the splitting approaches with the aim of finding the one most suitable with our solution. Here, our goal is to cluster the documents and to explore the behavior of the proposed architecture when varying the number of H splits, obtaining H sub-matrices. Then we adopt a random split sentence method. For each $SD^{(i)}$ matrix, the sentences are divided into H sub-sets thereby forming H sub-matrices $SD^{(i)}$. So, The number of $FT - Sim^{(i)}$ instances in the proposed network is equal to the number of splits H.

For example, let us consider a problem with one [documents/sentences] matrix of size I by J in which we just want to cluster the documents. we can divide the problem into a collection of h matrices of size n by m/H. Thus, by using a distributed version of $FT - Sim^{(i)}$ on h cores, we will gain both in time and space complexity.

By splitting a matrix, we lost some information. The solution does not compute the co-similarities between all pairs of sentences but only between the words occurring in each $\widetilde{SD}^{(i)}$. Thanks to the feedback loops of this architecture and to the presence of the common similarity matrix $\tilde{D}_2$, we will be able to spread the information through the network and alleviate the problem of inter-matrice comparisons.

Thus, by using a parallel version of $FT - Sim^{(i)}$ on H cores, we will gain both in time and space complexity: indeed,the time complexity decreases, leading to an overall gain of $1/H^2$ [16]. In the same way, the memory needed to store the similarity matrices between words will decrease by a $1/H$ factor.

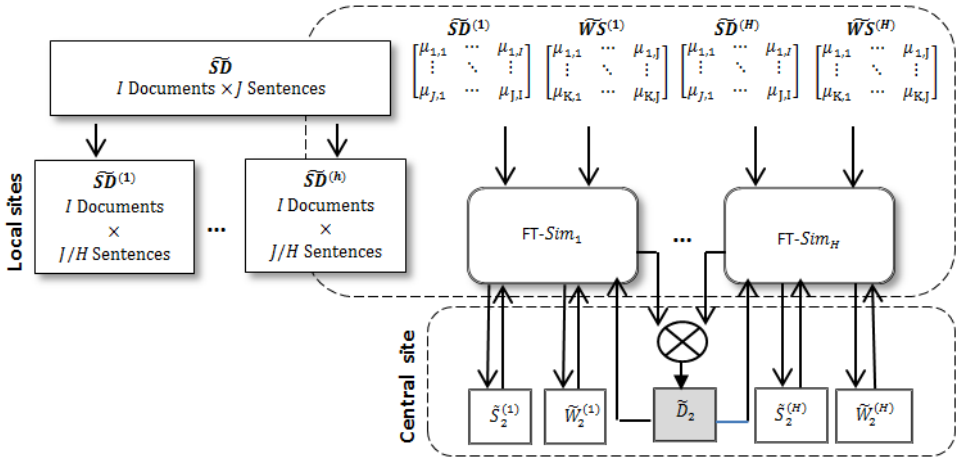

Fig. 4. Splitting-based parallel architecture.

V. Conclusion

In this paper, a fuzzy triadic similarity model, called FT-Sim, for the co-clustering task has been proposed. It takes, iteratively, into account three abstraction computing levels Document × Sentences × Words. The sentences consisting of one or more words are used to designate the fuzzy similarity of two documents. We are able to cluster together documents that have similar concepts based on their shared (or similar) sentences and in the same way to cluster together sentences based on words. This also allows us to use any classical clustering algorithm such as Fuzzy-C-Means (FCM) [17] or other fuzzy partitioned-based clustering approaches [18].

Our proposition has been extended to suit with multi-view models. Because the domain of text clustering focuses on documents and their similarities, in our proposition we spread informations about document similarities. We have presented three parallel architectures that combine FT-Sim instances to compute similarities from different sources.

Actually, we need to further analyze the theoretical points of view and the behavior of the three architectures in a multi-threading programming.

References

[1] F. Hussain, *X-Sim: A New Cosimilarity Measure: Application to Text Mining and Bioinformatics*, Phd Thesis, 2010.

[2] H. Chim and X. Deng, *Efficient Phrase-Based Document Similarity for Clustering*, Knowledge and Data Engineering, IEEE Transactions, vol. 20, pp. 1217-1229, 2008.

[3] G. Salton and C. Buckley, *Term-weighing approaches in automatic text retrieval*, In Information Processing and Management, vol. 34, pp. 513-523, 1988.

[4] S. Kundu, *Min-transitivity of fuzzy leftness relationship and its application to decision making*, Fuzzy Sets and Systems, vol. 93, pp. 357-367, 1997.

[5] L.A. Zadeh, *Fuzzy sets*, Information and Control 8, pp. 338-353, 1965.

[6] W. Tang and Z. Lu and I.S. Dhillon, *Clustering with multiple graphs*, proceedings of the 9^{th} IEEE International Conference on Data Mining, pp. 1016-1021, 2009.

[7] F. de Carvalho and Y. Lechevallier and F.M. de Melo, *Partitioning hard clustering algorithms based on multiple dissimilarity matrices*, Pattern Recognition 45, pp. 447-464, 2012.

[8] G. Bisson and C. Grimal, *Co-clustering of Multi-View Datasets: a Parallelizable Approach*, IEEE International Conference on Data Mining, pp. 828-833, 2012.

[9] I. Drost and S. Bickel and T. Scheer, *Discovering communities in linked data by multi-view clustering*, proceedings of the 29^{th} Annual Conference of the German Classication Society, Studies in Classication, Data Analysis, and Knowledge Organization, pp. 342-349, 2005.

[10] A. Kumar and H. Daume, *A co-training approach for multi-view spectral clustering*, proceedings of the 28^{th} International Conference on Machine Learning, pp. 393-400, 2011.

[11] H. Chim and X. Deng, *Efficient Phrase-Based Document Similarity for Clustering*, Knowledge and Data Engineering, IEEE Transactions, vol. 20, pp. 1217-1229, 2008.

[12] J.M. Torres-Moreno and P.Velzquez-Morales and J.-G. Meunier, *Cortex: Un algorithme pour la condensation automatique de textes*, Colloque Interdisciplinaire en Sciences Cognitives, 2001.

[13] M. Sven and L. Jorg and N. Hermann, *Algorithms for bigram and trigram word clustering*, Speech Communication, vol. 24, pp. 19-37, 1998.

[14] B. Long and X. Wu and Z.M. Zhang and S.Y. Philip, *Unsupervised learning on k-partite graphs*, proceedings of the 12^{th} ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 317-326, 2006.

[15] S. Alouane, M. Sassi Hidri and K. Barkaoui, *Fuzzy Triadic Similarity for Text Categorization: Towards Parallel Computing*, International Conference on Web and Information Technologies, pp. 32-37, 2013.

[16] G. Bisson and C. Grimal, *An Architecture to Efficiently Learn Co-Similarities from Multi-View Datasets*, International Conference on Neural Information Processing, pp. 184-193, 2012.

[17] J.C. Bezdek, *FCM: The fuzzy C-Means clustering algorithm*, Computers et Geosciences, vol. 10(2-3), pp. 191-203, 1984.

[18] J.B. MacQueen, *Some methods for classification and analysis of multivariate observation*, proceedings of the 5^{th} Berkeley Symposium on Mathematical Statistics and Probability, pp. 281-297, 1967.