FOCT: Fast Overlapping Clustering for Textual data

ATEFEH KHAZAEI1, HAMIDREZA KHALEGHZADEH2, and MOHAMMAD GHASEMZADEH3

1School of Computer Science, University College Dublin, Dublin, Ireland (e-mail: atefeh.khazaei@ucdconnect.ie)
2School of Computing, University of Portsmouth, United Kingdom (e-mail: hamidreza.khaleghzadeh@port.ac.uk)
3Computer Group, Engineering Campus, Yazd University, Yazd, Iran (e-mail: m.ghasemzadeh@yazd.ac.ir)

Corresponding author: Atefeh Khazaei (e-mail: atefeh.khazaei@ucdconnect.ie).

ABSTRACT Text clustering is used to extract specific information from textual data and even categorizes text based on topic and sentiment. Due to inherent overlapping in textual documents, overlapping clustering algorithms have become a suitable approach for text analyzing. However, state-of-the-art algorithms are not fast enough to analyse a large volume of textual data within tolerable time limits. In this research, we propose our text clustering algorithm, FOCT, which is a fast overlapping extension of SOM, one of the best algorithms for clustering textual data. We apply some heuristics to extract special characteristics presenting in textual data and establish a very fast overlapping clustering algorithm. We use fast methods to represent the vectors of documents, compute the similarity of documents and neurons and update the weights of neurons. In our algorithm, each document can belong to one or more neurons and this is in line with what many documents have in their essence. We analyse the efficiency of the proposed algorithm over k-means, OKM, SOM and OSOM clustering approaches and experimentally demonstrate that it runs 12 to 690 times faster, and the overlap size of FOCT clusters is closer to the overlap size of the original data. The quality of clusters is also measured by four different internal and external evaluation criteria where FOCT clusters represent up to 64% better quality.

INDEX TERMS FOCT, Overlapping clustering, Self-organizing feature maps, Text mining.

I. INTRODUCTION

Overlapping clustering algorithms are unsupervised methods for knowledge extraction from data. These algorithms classify a set of objects so that each object can belong to one or more groups [1]. Overlapping clustering algorithms are momentous because a large number of real-world datasets (such as textual data, computational biology, and social networks) contain innate overlapping. In this paper, we especially target text overlapping clustering methods which are used for different tasks, such as grouping similar articles, news or tweets, the analysis of customer feedback and comments, and finding meaningful implicit subjects in documents. In addition to the challenges in usual clustering tasks, such as evaluation, parametrization and scale-up, overlapping clustering algorithms also face their issues, including high computational complexity and the impact of overlap size on the number of clusters [2], where N’Cir et. al. highlight the slow processing speed as one of the most critical challenges in overlapping clustering algorithms for big volumes of data [2]–[4].

Self-Organizing Map (SOM) is one of the common algorithms for clustering textual data. It is based on artificial neural networks (ANN) and generally partitions high-dimensional data into a low-dimensional space (typically two-dimensional). It is experimentally demonstrated that SOM is one of the most suitable algorithms for clustering textual data. In text clustering, the inputs are usually high-dimensional sparse vectors; while the SOM output is often a two-dimensional map that can be understood and interpreted by humans. In addition, SOM is relatively resistant to noisy documents. Therefore, these features make SOM suitable for text-based applications such as digital libraries [5], [6].

Considering the importance of text clustering (which often have overlap between categories), and the ability of SOM for text clustering, presenting an overlapping clustering algorithm based on SOM that is suitable and fast for textual data, can be effective to extract knowledge from big textual datasets.

The basic SOM algorithm does not consider the overlap existing between clusters. The first overlapping version of SOM, called OSOM, was introduced by Cleuziou in which an instance can belong to one or more neurons [7]. However, Cleuziou conducted all experiments using non-textual
datasets. In this paper, we will experimentally analyse the efficiency of OSOM and show that it is extremely slow for text clustering.

In this paper, we propose a new Fast Overlapping Clustering algorithm for Textual data (FOCT), which is an overlapping extension of SOM. FOCT takes advantage of a set of fast methods and techniques to appropriately represent the vectors of documents and accelerate the computation of similar documents and neurons and speed up the process of updating the weights of neurons. The efficiency of our algorithm is compared with k-means, OKM (Overlapping K-Means) [8], SOM [9] and OSOM [7] algorithms. We experimentally demonstrate that that FOCT is fast and efficient in overlapping clustering of textual data.

The rest of this paper is organized as follows: Section II presents relevant literature. In Section III our suggested algorithm is introduced. Our evaluation measures, data and feature extraction method, and experiment conditions and their results are addressed in Section IV. Finally, Section V conclusions the paper.

II. LITERATURE REVIEW

N’Cir et al. in chapter 8 of “Partitional Clustering Algorithms” book reviewed the most important overlapping clustering algorithms. They introduced a categorisation of overlapping clustering methods based on their conceptual algorithms” book reviewed the most important overlapping clustering algorithms. The efficiency of our algorithm is compared with k-means, OKM (Overlapping K-Means) [8], SOM [9] and OSOM [7] algorithms. We experimentally demonstrate that that FOCT is fast and efficient in overlapping clustering of textual data.

The rest of this paper is organized as follows: Section II presents relevant literature. In Section III our suggested algorithm is introduced. Our evaluation measures, data and feature extraction method, and experiment conditions and their results are addressed in Section IV. Finally, Section V conclusions the paper.

II. LITERATURE REVIEW

N’Cir et al. in chapter 8 of “Partitional Clustering Algorithms” book reviewed the most important overlapping clustering algorithms. They introduced a categorisation of overlapping clustering methods based on their conceptual approach [2]: the summary of these main categories are shown in Table 1.

TABLE 1: The categorisation of overlapping clustering algorithms.

| Categories  | Description                                                                 | Example |
|-------------|-----------------------------------------------------------------------------|---------|
| Correlation | These algorithms are the overlapping extensions of the correlation clustering algorithms. They are defined as optimization problems to allow overlaps by assigning set of one or more labels for each data object. | 10–13   |
| Generative  | These algorithms are based on biological processes. They hypothesize that each data is the result of a mixture of distributions. | 14–16   |
| Graphical   | These algorithms are based on graph theory and are mostly used for community detection in complex networks. | 17–20   |
| Hierarchical| These algorithms try to reduce the differences between the original dataset and the obtained hierarchical structure. | 21–23   |
| Partitional | These algorithms are the most popular overlapping clustering research group among researchers and consist in modifying the clusters resulting from a standard method into overlapping clusters or in proposing new objective criteria to model overlaps. | 8, 24–30|
| Topological | These algorithms are overlapping extensions of topological maps (such as SOM). Our proposed algorithm is in this class. | 7       |

Textual data have inherent overlapping and some researchers have focused on this issue (e.g. [31]–[34]). In this research, we introduce a topological overlapping clustering algorithm that is suitable for textual data. The proposed algorithm is compared with k-means, OKM (Overlapping K-Means), SOM and OSOM algorithms. The k-means and OSM are non-overlapping clustering algorithms, and their performance is our baseline. The OKM [8], and OSOM [7] were selected among the overlapping clustering algorithms. The OKM is one of the most common base algorithms among the overlapping clustering algorithms, and according to Cleuziou’s claim [8] is suitable for textual data. The structure of OSOM is the most similar algorithm to ours; both of them are topological and this method is relatively new. In the remaining of this section, a little more details of OKM and OSOM algorithms will be mentioned.

Cleuziou, in 2008, has presented the OKM algorithm, an overlapping version of k-means. In OKM, a new objective criterion has been proposed, and this criterion generalizes the least squared objective criterion used in k-means. The goal of the OKM objective function is minimization under constraints of multi-assignment. In a nutshell, OKM prospects the space coverage rather than the space partitions like k-means does. Using this proposed criterion each sample can belong to one or more clusters [8].

In 2013, Cleuziou introduced an algorithm called OSOM (Overlapping SOM), in which each sample can assign to one or more neurons of the map. He showed SOM as a topological method that is suitable for the overlapping clustering because it provides a structure with the following characteristics [7]:

- In this clustering method, the number of clusters is not an input parameter. The input data is structured on a map of neurons; and the number of neurons is usually much greater than the number of potential final clusters;
- Using the topological methods, the topological correctness of the overlaps is convinced in a simple and straightforward way. Topological connections are clear and explicit on the map. Using these methods, we can easily propose efficient assignment heuristics on the overlaps;
- Topological methods can be applied as a first clustering structure on the data; and they can be used for a further hierarchical structuration.

In OSOM published results none of the investigated datasets were textual [7]. In this research, we experimentally show that OSOM algorithm is not suitable for textual data.

In summary, Table 2 shows a comparison of the used clustering algorithms in this research. More details of complexity are provided in Section III-E.

III. THE FOCT ALGORITHM

Because of inherent overlapping in textual data, overlapping clustering algorithms extract more appropriate results from textual datasets. In fact, overlapping clustering algorithms
assign textual documents to more than one cluster; this is exactly a common and essential feature, which can be observed in many real-world documents.

SOM is a classic and suitable algorithm addressing the problem of unsupervised text clustering. There are research works developing new methods for text clustering and mining based on SOM. As mentioned in the previous section, Cleuziou \cite{7} introduced an overlapping SOM algorithm, named OSOM. In this section, we present a topological overlapping clustering algorithm based on OSOM. In section \cite{7} it is experimentally demonstrated that OSOM, unlike the presented algorithm, is not suitable for textual data in the real world.

Textual datasets usually have considerable numbers of documents where the extracted feature vectors from these documents are most of the time sparse and high-dimensional. Furthermore, the OSOM complexity is relatively high, and it cannot run big volumes of textual data with acceptable speed. These limitations are the most important challenges to apply OSOM algorithm for textual data.

In this section, we propose a novel algorithm, FOCT (Fast Overlapping Clustering for Textual data) that addresses these challenges. FOCT deploys a number of optimisations to shrink the amount of computation and therefore speed up the process of text clustering. These optimisations are listed below:

- In comparison to conventional SOM vectors, the vectors of neurons and documents in FOCT have different lengths. This variety in vectors decreases the execution time and the required memory.
- The most frequent and most time consuming operation in SOM is the distance computation of neurons and documents. To speed up this operation, FOCT uses a fast similarity computation method.
- FOCT is an overlapping clustering algorithm; it means each instance can belong to one or more neurons. To achieve a faster algorithm, the winner neuron sets are obtained by a greedy heuristic.
- The neurons’ weights are updated using a fast method which increases the corresponding weight values of the document words (by simple sum operation).

Applying the above optimisations lead to a fast algorithm that address the problem of fast textual data clustering. A simple flow chart of our algorithm is showed in Figure 1. In the remainder of this section, each of the above optimisations is explained in detail. We then describe our proposed algorithm, FOCT, and analyse its complexity.

TABLE 2: The comparison of this research clustering algorithms.

| Name   | Overlapping support? | Category     | Complexity                                                                 |
|--------|----------------------|--------------|----------------------------------------------------------------------------|
| K-means| No                   | Partitional  | $O(t,n,k)$: $k$: number of neurons, $t$: number of iterations, $n$: size of the dataset |
| OKM    | Yes                  | Partitional  | $O(t,n,\log k)$: $k$: number of neurons, $t$: number of iterations, $n$: size of the dataset |
| SOM    | No                   | Topological  | $O(t,2^k)$: $k$: number of neurons, $t$: number of iterations               |
| OSOM   | Yes                  | Topological  | $O(t,2^k)$: $k$: number of neurons, $t$: number of iterations               |
| FOCT   | Yes                  | Topological  | $O(k^d.t.d)$: $k$: number of neurons, $t$: number of iterations, $d$: maximum length of feature vectors |

FIGURE 1: A simple flow chart of FOCT algorithm.
**A. VECTORS OF DOCUMENTS AND NEURONS**

In text clustering, most algorithms suffer from high-dimensional feature vectors and many iterations which result in a remarkable execution cost. Here we explain how to present the feature vectors in FOCT. As shown in Section IV-B, the feature extraction process will be elucidated.

In most SOM-based approaches, the vectors of input instances and neurons are represented with the same dimension number. Using the equal length vectors for textual data consisting of high dimensional and sparse vectors leads to significant memory consumption and longer runtime.

To tackle this issue, in this research, each document is represented as indexes of words (instead of a high dimensional vector). The index representation significantly reduces the length of vectors, which leads to less memory consumption and shorter execution time. Although each document is represented by indexes, neurons have their high-dimensional vectors. In other words, the vector of each neuron determines the weight of each word for that neuron. The number of neurons is often much less than the total number of documents. Figure 2 shows how to create document indexes using lists of words. In this procedure, for each document, the index of each word in the vector of the current document is extracted (using $\text{VectorW}$ and $\text{List}$) and added to the corresponding index of that document (Vector). This procedure with minor changes can be used dynamically for online applications.

**B. FAST SIMILARITY COMPUTATION**

After representing documents in form of indexes of words, we need a function to compute the similarity between neurons and documents, which are represented in different ways. Figure 3 shows the proposed algorithm computing the similarity between documents and neurons. The algorithm considers equal importance for all words extracted from documents. To obtain the similarity between a given document and a neuron, the algorithm first extracts the weight of each word existing in the document from the neuron weight vector and then sum up the weights.

**Lemma III.1.** The time complexity of the Similarity function is $O(k.d)$, where $k$ is the number of neurons and $d$ represents the maximum length of feature vectors.

**Figure 3** shows the proposed algorithm computing the similarity between documents and neurons.

**Proof:** As shown in Figure 3, Similarity function consists of two nested loops. The outer for loop counts the total number of neurons ($k$), and the inner one scans a feature vector of size $d$. Therefore the time complexity of this function is $O(k.d)$.

**C. CANDIDATE AND WINNER NEURONS SETS**

The main difference between our overlapping algorithm (FOCT) and the non-overlapping one (SOM) is in finding the optimal winner neurons set. To find such an optimal set, one straightforward approach is to create all possible combinations and find the optimal set. Since the number of neurons subsets can be exponential ($2^K$ subsets with $K$ neurons in the map), this approach has exponential complexity.

To address this problem, we introduce a greedy heuristic algorithm (such as Cleuziou’s OSOM algorithm [7]), which is called SWN (Set of Winner Neurons), to speed up the process of finding optimal winner neurons and obtain more appropriate topological organization of the map. The steps in the algorithm are listed as follows:

1. Find the most similar neuron to the current document (first winner neuron) and assign the document to it.
2. Investigate the neighbors of the most similar neuron as candidate neurons and find the most similar neuron to the current document among them. It should be noted that the Similarity function (see III-B) is invoked to find the most similar neighbor neuron (Step 3 and 4 in Figure 4).
3. If the winner neuron in the candidate neurons set has at least one of the required conditions, then the current document will be assigned to it too; and after that, the candidate neurons set will be updated with neighbors of the new winner neuron and this process will be repeated; otherwise, the process of winner neurons set creation will be finished.

**Figure 4** presents SWN pseudocode. As shown in the Figure, to assign a neuron to the winner neurons set, at least

---

**Table 1**

| Function | Input | Output |
|----------|-------|--------|
| **VECTOR** | vectorW: The list of total words, List: List of words for each document | Vector: The word indexes for each document |
| **SIMILARITY** | Vector(d), VectorN | Similarity: The similarity vector between the document and neurons |

---

**Figure 2** shows how to create document indexes using lists of words.
FIGURE 4: Finding the winner neurons set.

one of the following conditions must be satisfied (step 7 in Figure 4):

1) The similarity of current document with average weights of new winner set is more than the similarity of current document with average weights of previous winner set.
2) The absolute difference of the current document similarity with average weights of new winner neurons set and the current document similarity with average weights of previous winner neurons set is less than a small threshold (ε). ε is one of the effective factors for the overlap size of FOCT output.

Lemma III.2. The time complexity of the SWN function is \( O(k^2, d) \), where \( k \) is the number of neurons and \( d \) represents the maximum length of feature vectors.

Proof: As shown in Figure 4, the single while loop in SWN function iterates \( k \) times in the worst case scenario. The Similarity function, with the time complexity of \( O(k, d) \) (Lemma III.1) is invoked in each iteration. Therefore, one can conclude that the execution cost of function SWN does not exceed \( O(k^2, d) \).

D. UPDATING WEIGHTS OF NEURONS

After finding the winner neurons set that contains the most similar neurons to the current document, the weight vector of each winner neuron and its neighbors are updated. To update the weight vector of winner neurons, the update rate, represented by \( ϕ \), is added to dimensions of the neuron’s vector that its corresponding word exists in the document.

FIGURE 5: Updating weight vector of each winner neuron and its neighbors.

Then, other documents which are similar to the current document have more chance to assign to this neuron. However, documents that are completely different from the current document still have a chance of being assigned to this neuron because other dimensions of the winner neuron weight vector are not adjusted. In order to overcome this problem, \( ϕ \) is also subtracted from other dimensions of the winner neuron weight vector that were not indexed in the current documents. The update rate \( ϕ \) decreases in each iteration.

To obtain a more appropriate topological organization of the map, the weight vectors of winner neurons neighbors are also updated. Updating neighbors is implemented by adding \( ϕ/2 \) value to the dimensions of neighbor weigh vectors that were indexed for the current document. The procedure of updating the weight vector of each winner neuron and its neighbors is presented in Figure 5.

Lemma III.3. The time complexity of the Update function is \( O(d) \), where \( d \) represents the maximum length of feature vectors.

Proof: As shown in Figure 5, Update function consists of two nested loops. The outer for loop iterates the max length of feature vectors (\( d \)), in the worst-case scenario. The number of each neuron neighbor is 8, so the inner for loop has a constant cost. Therefore the time complexity of this function is \( O(d) \).

E. FOCT AND ITS COMPLEXITY

In previous subsections, it was explained how to compute similarity (SIMILARITY procedure), find the winner neurons set (SWN procedure) and update winner neurons and their neighbors (UPDATE procedure). Now, we propose our clustering algorithms, FOCT, which use these procedures as building blocks. FOCT, as a fast extension of SOM, is an
FOCT(Vector, K1, K2, Tmax, 𝜑𝑖, 𝜑𝑓)

Input: Vector: The word indexes of documents, K1, K2: The number of lines and columns respectively on the map. Tmax: Number of iterations, 𝜑𝑖: The initial value of 𝜑. 𝜑𝑓: The final value of 𝜑.

Output: VectorN: The vectors of neurons, II = [𝜋𝑖]𝑐⋅1: The final clustering where 𝜋𝑖 = {Vector(d𝑖) ∈ Vector} VectorN(k) ∈ A𝑖

1. Initialize VectorN with random values;
2. For t:1..Tmax
   3. Select randomly one data Vector(d𝑖) ∈ Vector;
   4. Similarity = SIMILARITY(Vector(d𝑖), VectorN);
   5. Winner = max(Similarity);
   6. Initialize the assignment A𝑖 = {VectorN(Winner)}.
      the combination 𝜋𝑖 = VectorN(Winner), and the neighborhood
      vi = {VectorN(nj) ∈ VectorN(Winner)} |d(zi, di) = 1];
   7. Winners = SWN(Vector(d𝑖), A𝑖, 𝜀, vi, 𝜀)
   8. 𝜑𝑡 = 𝜑𝑖(𝜑𝑡/𝜑𝑖)1/Tmax
   9. For each neuron (nj) in Winners
      10. VectorN = UPDATE(VectorN, nj, Vector(di), 𝜑t);
   11. End for
12. End for

**FIGURE 6:** The FOCT algorithm.

overlapping clustering algorithm, mainly suitable for textual data. Figure 6 shows the algorithm. In each iteration of this algorithm, it takes a document as input and performs the following steps: 1) compute the similarity of document and neurons, 2) find the winner neurons set and 3) update weight vectors of neurons.

In FOCT, 𝜑𝑡 is the update rate of tth iteration and decreases in each iterations. Equation 1 formulates how to calculate 𝜑. In this equation, 𝜑𝑖 is the initial value of 𝜑, and 𝜑𝑓 represents the final value of 𝜑.

\[
\varphi_t = \varphi_i \left(\frac{\varphi_f}{\varphi_i}\right)^{t/T_{max}}
\] (1)

One of the most challenging problems in overlapping algorithms is their time complexity. The overlapping clustering algorithms can be drastically more complex than usual ones. It is because of considering overlapping structures which leads to a significant increase in the search space size [2].

The complexity of basic SOM is \(O(k.t.m)\), where \(k\) is the number of neurons, \(t\) determines the number of iterations, and \(m\) represents the training time [9]. The OSOM, another overlapping algorithm based on SOM, has an exponential complexity equals to \(O(t.2^k.m)\) [7].

As proved in Theorem III.4 in this paper, we have proposed FOCT that performs overlapping clustering in a time complexity of \(O(k^2.t.d)\).

**Theorem III.4.** The FOCT algorithm has a time complexity of \(O(k^2.t.d)\).

**Proof:** The main for loop of the FOCT algorithm iterates \(t\) times, and a set of functions is invoked in each iteration where the costliest one is SWN with time complexity of \(O(k^2.d)\). This way, it is proved that the time complexity of FOCT is \(O(k^2.t.d)\).

In comparison with SOM and OSOM, we can see that the maximum theoretical value for \(m\) (training time), in the FOCT algorithm, is \(k.d\) (\(k\) and \(d\) are the number of neurons and the maximum length of feature vectors respectively).

Since in text mining, the binary vectors of documents are too sparse, FOCT represents each document as indexes of words (instead of using high-dimensional vectors used in SOM and OSOM). This leads to a significant decrease in the size of feature vectors compared to the vector sizes in SOM and OSOM. In addition, due to employing a fast similarity computation method (Section III-B) and a fast neuron weights update method (Section III-D), the training time of FOCT algorithm is less than the time taken by SOM and OSOM. In Section IV, we practically analyse the efficiency of FOCT and show that it is faster than others.

**IV. EXPERIMENTS**

In this section, we evaluate our proposed overlapping text clustering. We first describe the evaluation measures and then introduce a dataset, which satisfies the particular requirements for overlapping clustering applications. Our feature extraction method is also explained. Finally, experiment conditions and obtained results are elucidated.

A. EVALUATION MEASURES

Overlapping clustering evaluation is a controversial issue. Clustering evaluation measures are divided into two main categories: internal and external. The internal measures aim to quantify the original information that is retrieved by clusters; whereas the external measures compare the obtained clusters with reference classes [2, 55].

Cleuziou proposed some measures to evaluate overlapping clustering algorithms, considering both internal and external measures. These measures use the Frobenius norm on two \(N \times N\) matrices \(A\) and \(B\). One of these matrices is obtained from the input data and the other from clusters (more detailed information in [7]).

\[
Q(A, B) = \sqrt{\frac{1}{N^2} \sum_{i,j} (A_{i,j} - B_{i,j})^2}
\] (2)

Cleuziou defined four matrices, each one containing a strategic information [7]:

- In matrix \(V\), \(V_{i,j}\) element includes the dissimilarity between the labels associated to data \(x_i\) and \(x_j\).
- Matrix \(U^{(1)}\) includes the distances on the grid between the neurons correlated to the data,
• Matrix $U^{(2)}$ includes the distances between the prototypes of the neurons correlated to the data,
• Matrix $U^{(3)}$ includes the initial distances between the data samples.

In our research, Cleuziou’s measures [7] are used to compare different clustering algorithms. Table 3 gives an overview of the matrix instantiating for $Q(A, B)$ and a short description of these measures.

TABLE 3: The evaluation measures based on Frobenius norm.

| Notation       | Description                                                                 | $A$ | $B$ |
|----------------|------------------------------------------------------------------------------|-----|-----|
| $Q_{\text{labels}}$ | This measure shows the relevance of labels and initial distances. It is inherent to data and independent of the clustering algorithm. It indicates matching between data descriptions and their labels. The lower value of this measure shows the higher matching between descriptions and labels. | $U^{(3)}$ | $V$ |
| $Q_{\text{ext topo}}$ | This external measure is used to evaluate the relevance of the map topology and the class labels. In other words, considering the sample labels, how close the similar samples are on the map. | $U^{(1)}$ | $V$ |
| $Q_{\text{ext classif}}$ | This external measure is used to evaluate the relevance of the obtained clusters and the class labels. In other words, considering the sample labels, how similar the samples are in each cluster. | $U^{(2)}$ | $V$ |
| $Q_{\text{int topo}}$ | This internal measure is used to evaluate the relevance of the map topology and initial distances. In other words, how close the similar samples are on the map. | $U^{(1)}$ | $U^{(3)}$ |
| $Q_{\text{int classif}}$ | This internal measure is used to evaluate the relevance of the obtained clusters and initial distances. In other words, how similar the samples are in each cluster. | $U^{(2)}$ | $U^{(3)}$ |

It should be mentioned that the $Q$ parameter in Table 3 represents the distance of two matrices. A lower value for $Q$s shows a higher matching between the two matrices.

In our research, in addition to the above measures, the overlap size is also considered. The overlap size is defined as the average number of clusters that each instance belongs to. In other words:

$$\text{OverlapSize} = \frac{\sum_{i=1}^{N} c_i}{N} \quad (3)$$

where $c_i$ is the number of clusters that the $i^{th}$ instance belongs to, and $N$ is the number of instances [2].

B. DATA AND FEATURE EXTRACTION

We use the Reuters-21578 dataset to build our experimental dataset. Reuters-21578 is one of the popular benchmarks in information retrieval and text mining research areas and consists of 21578 articles where each one belongs to one or more categories (topics) [56]. To build the experimental dataset, the articles from Reuters-21578 are selected which satisfy the following two conditions:

1) Involving three sections ‘Topic’, ‘Title’ and ‘Body’.
2) There are at least 20 articles for each topic in the dataset. This condition ensures that we have enough data for each topic to train neurons.

It should be noted that the Reuters-21578 dataset is a fairly clean dataset since all the articles in the dataset have been revised and proof-read, and therefore, it does not require to apply much pre-processing such as noise detection [37].

After building the experimental dataset (including 9907 articles), the feature vector of each article in the dataset is extracted from its ‘Title’. We consider ‘Title’ to make the size of the feature vector smaller. The main steps of feature vector extraction are:

1) Extracting document words.
2) Removing stop-words: Stop-words are kind of words that are prevalently used in textual documents, and therefore, cannot be effective in distinguishing (classifying) documents. Articles like a and the as well as pronouns such as it and them are examples of stop-words. These common words can be discarded during the feature extraction steps [38]. [39].
3) Reducing words to stems: Stemming is the process of reducing words to their stems. This process reduces the number of words and leaves us with fewer forms of a word to deal with, and therefore it would reduce the length of the feature vectors. In this researcher, we use the Porter stemming algorithm, which is the most widely used stemmer for English texts [38].
4) Selecting frequent word: In this step, the words with a frequency less than 10 are removed. Kohonen in 1995 showed that not only removing words with low frequency (even if they are meaningful) has no negative influence on clusters, but it even sometimes leads to obtaining better results, reduces the length of feature vectors and makes them less sparse. [9].
5) Creating binary feature vectors: In the last step, the binary feature vectors are created that their elements determine the existence or non-existence of words in a document. As mentioned earlier, we use a different feature vector representation in FOCT that leads to faster processing and less required memory. In this research, WVT (word vector tool) is employed for feature extraction, which is a flexible Java library for statistical language modelling [40].

Table 4 summarises the characteristics of our experimental dataset obtained by applying the aforementioned steps.
C. EXPERIMENT CONDITIONS AND RESULTS

In this section, we first explain the conditions that our experiments take place and summarize the experimental results. In this paper, the experiments are conducted under the following conditions:

- In each experiment, the number of iterations, which is the number of selected documents ($T_{\text{max}}$), are considered to be 2000, 4000, 6000, 8000, 9907 documents. We chose the documents randomly and without any replacement.
- The map sizes (in SOM, OSOM and FOCT) and number of clusters (in k-means and OKM) are $8 \times 8$ (64), $10 \times 10$ (100) and $12 \times 12$ (144).
- The value of $\epsilon$ in k-means, OKM and FOCT algorithms is 0.05.
- The values for $\epsilon_t$ (in SOM and OSOM) and $\varphi_t$ (in FOCT) are calculated by equation 1. Their initial and final values are 0.1 and 0.005, respectively.
- In each run, the weight values of neurons (in SOM, OSOM and FOCT) and the values of prototypes (in k-means and OKM) are initialised with random numbers where the initial values for neurons and the initial values for prototypes are the same.
- All experiments are conducted on a PC consisting of an Intel Core i5 2.4 GHz CPU and 4 GB memory running Windows 10. We repeat each experiment five times, and the final result is equal to the average of the five runs.

Table 5 shows $Q^\text{ext}_{\text{topo}}$ and $Q^\text{int}_{\text{topo}}$ measures. It compares the topologies of maps for SOM, OSOM and FOCT algorithms with respect to the classes (external validation) and to the initial description of data (internal validation) respectively.

Table 6 considers only the obtained clusters and ignores the topological organization using $Q^\text{ext}_{\text{classif}}$ and $Q^\text{int}_{\text{classif}}$ measures.

The main challenge of current overlapping clustering algorithms is their slow speed in the clustering of big volumes of data. That is why one of the most important criteria in the evaluation of overlapping clustering algorithms is their running time. Table 7 shows the CPU times of algorithms for the above experiments.

The size of overlap is another measure that is important to evaluate overlapping clustering algorithms. Algorithms with an overlap size closer to the overlap size of the original data return better clustering solutions. Table 8 shows the overlap size for OKM, OSOM and FOCT algorithms in our experiments.

D. DISCUSSION

As mentioned before, a lower value for $Q_{\text{label}}$, as an algorithm-independent measure, shows the higher match between descriptions and labels. Regarding our experiment results, while $Q_{\text{label}}$ is equal to 0.5175 for total instances (including 9907 documents), this value fell in the range of 0.5175 ± 0.003 when all instances were not used. These figures show that descriptions and classes are not well-match enough. This is one of the most important reasons for the weakness of external results ($Q^\text{ext}_{\text{topo}}$).

Table 5 compares $Q^\text{ext}_{\text{topo}}$ and $Q^\text{int}_{\text{topo}}$ for SOM, OSOM and FOCT algorithms. While $Q^\text{ext}_{\text{topo}}$ values for SOM and FOCT are close together, with the maximum difference of ±0.02, but FOCT outperforms SOM when it comes to the $Q^\text{int}_{\text{topo}}$ measure. The results show that the maps obtained by the FOCT algorithm have appropriate topologies, and they can be efficient for clustering. In these experiments, OSOM showed the weakest results.

As shown in Table 6 while the overlapping algorithms OSOM and OKM fall behind FOCT, but non-overlapping algorithms SOM and k-means outperformed FOCT in terms of the $Q_{\text{classif}}$ measure where SOM for all and k-means for some of the experiments showed better results than FOCT.

In a nutshell, in all experiments, FOCT outperforms two overlapping algorithms, OKM and OSOM, but is behind the non-overlapping algorithms, k-means and SOM, in some experiments (especially based on the $Q_{\text{classif}}$ measures). The main reason for this observation is that non-overlapping algorithms only consider the main category of each document while overlapping ones analyze overlapping between document categories. Ignoring other categories makes it easier for non-overlapping algorithms to put more similar documents in each cluster so these results are not far from expectations.

This research aims to present an overlapping clustering algorithm for textual data due to the importance of overlapping in textual documents.

Figure 7 shows the $Q$s per epochs charts and compare FOCT with the other overlapping algorithms. In all these charts, the bottom line belongs to FOCT. It implies that FOCT outperforms the other overlapping clustering algorithms based on all the measures.

In terms of execution time, as one of the most important criteria in clustering algorithms, the FOCT algorithm ran much faster than the other algorithms.

Concerning the overlap size, which is shown in Table 8, the overlap size of this research data, including 9907 documents, is 1.2516. In Table 9, we summarised the maximum and minimum overlap sizes obtained by the overlapping algorithms. As shown in Table 9 the overlap size of clusters found by FOCT is closer to the overlap size of the original data. This signifies that FOCT is the best algorithm in terms of overlap size.

V. CONCLUSION

Nowadays, text clustering techniques are applicable in various areas for extracting specific knowledge from text and even categorising textual documents based on topic and sentiment. Due to inherent overlapping in textual documents, overlapping clustering algorithms have turned into a suitable approach for analysing text.

In this paper, we propose FOCT, a fast overlapping clustering algorithm for textual data. FOCT is a fast overlapping extension of SOM, and it employs some heuristics.
to extract special characteristics presenting in textual data. The proposed algorithm uses fast methods to represent the vectors of documents, compute the similarity of documents and neurons and update the weights of neurons. In FOCT, each document can belong to one or more neurons where this feature represents the characteristic exiting in real-life textual documents.

The efficiency of FOCT is analysed over two non-overlapping (k-means and SOM) and two overlapping clustering algorithms (OKM and OSOM). We experimentally demonstrated that FOCT outperforms the aforementioned clustering algorithms in terms of topological measures and also execution time. In addition, FOCT shows a better overlap size compared to the other overlapping algorithms. This implies that FOCT clusters are closer to the clusters of original data.

The overlap sizes of most of the available text datasets are small (e.g. Routers overlapping size is 1.25). As future works, we find and compile some text datasets with different and more overlapping sizes. A few datasets with different overlapping sizes can better evaluate and compare FOCT and other overlapping algorithms.

REFERENCES

[1] A. E. Ezugwu, A. K. Shukla, M. B. Agbaje, O. N. Oyelade, A. Jose-Garcia, and J. O. Agushaka, “Automatic clustering algorithms: a systematic review and bibliometric analysis of relevant literature,” Neural Computing and
TABLE 7: The required CPU times of algorithms.

| Num of Docs | Algorithm | CPU Time |
|-------------|-----------|----------|
|             |           | Size of map (or clusters) |
|             |           | 8 × 8 (64) | 10 × 10 (100) | 12 × 12 (144) |
| 2000        | Kmeans    | 10.4719   | 10.3594     | 16.6844     |
|             | OKM       | 5.1844    | 6.6344      | 9.9938      |
|             | SOM       | 86.4609   | 198.9065    | 268.1328    |
|             | OSOM      | 68.7219   | 161.1875    | 345.9438    |
|             | FOCT      | 0.3937    | 0.5313      | 0.7531      |
| 4000        | Kmeans    | 16.4219   | 13.1016     | 54.6797     |
|             | OKM       | 12.7422   | 19.6484     | 99.1328     |
|             | SOM       | 1,901.1904| 225.7109    | 332.5072    |
|             | OSOM      | 155.4065  | 555.7444    | 687.7344    |
|             | FOCT      | 0.9141    | 1           | 1.2578      |
| 6000        | Kmeans    | 31.4531   | 43.7891     | 99.1719     |
|             | OKM       | 18.4141   | 20.5469     | 35.7188     |
|             | SOM       | 167.4453  | 382.8438    | 999.1328    |
|             | OSOM      | 191.2500  | 528.1484    | 823.8906    |
|             | FOCT      | 1.2051    | 1.3594      | 1.4297      |
| 8000        | Kmeans    | 41.9063   | 70.1875     | 69.3438     |
|             | OKM       | 29.0938   | 26.5469     | 37.7656     |
|             | SOM       | 224.5938  | 522.5625    | 1073.7      |
|             | OSOM      | 264.9688  | 823.8906    | 1094.2      |
|             | FOCT      | 1.9531    | 1.3000      | 1.6406      |
| 9907        | Kmeans    | 31.1875   | 49          | 88.4844     |
|             | OKM       | 25.3750   | 29.2156     | 53.5625     |
|             | SOM       | 280.7500  | 638.2656    | 1528.1      |
|             | OSOM      | 331.2500  | 683.2031    | 1338.1      |
|             | FOCT      | 1.6406    | 1.7813      | 1.9375      |

TABLE 8: Comparing overlap size of overlapping algorithms.

| Num of Docs | Algorithm | Size of overlap |
|-------------|-----------|-----------------|
|             | Size of map (or clusters) |
|             | 8 × 8 (64) | 10 × 10 (100) | 12 × 12 (144) |
| 2000        | OKM       | 3.4000         | 3.8064     | 2.4996     |
|             | OSOM      | 9.4369         | 9.9924     | 6.8005     |
|             | FOCT      | 1.4766         | 1.9443     | 2.0793     |
| 4000        | OKM       | 4.0003         | 3.5000     | 2          |
|             | OSOM      | 7.0505         | 0          | 1          |
|             | FOCT      | 1.4229         | 1.5025     | 1.8575     |
| 6000        | OKM       | 2.9998         | 3.1232     | 1.9998     |
|             | OSOM      | 8.4997         | 10.9206    | 7          |
|             | FOCT      | 1.2019         | 1.2608     | 1.6645     |
| 8000        | OKM       | 10.5874        | 9          | 8.9995     |
|             | OSOM      | 1.1396         | 1.3174     | 1.3976     |
| 9907        | OKM       | 10.5814        | 9.9997     | 5          |
|             | OSOM      | 1.1319         | 1.1905     | 1.2198     |

FIGURE 7: The Q’s per epochs for the overlapping clustering algorithms.
| Algorithm | Min of overlap size | Max of overlap size |
|-----------|---------------------|---------------------|
| OKM       | 1.9998              | 4.0003              |
| OSOM      | 5                   | 10.9206             |
| FOCT      | 1.1319              | 2.0793              |

TABLE 9: Max and Min overlap sizes for algorithms.

---

F. Bonchi, A. Gionis, and A. Ukkonen, “Overlapping correlation clustering,” in 11th IEEE International Conference on Data Mining (ICDM2011), Vancouver: IEEE, 2011, pp. 51–60.

S. Khanmohammadi, N. Adibeig, and S. Shanehbandy, “An improved S. Baadel, F. Thabtah, and J. Lu, “Mcoke: Multi-cluster overlapping k-means extension algorithm,” International Journal of Computer, Electrical, Automation, Control and Information Engineering, vol. 9, no. 2, pp. 427–430, 2015.

C.-E. B. N’Cir, N. Essoussi, and M. Limam, “Kernel-based methods to identify overlapping clusters with linear and nonlinear boundaries,” Journal of classification, vol. 32, no. 2, pp. 176–211, 2015.

S. Khanmohammadi, N. Adibeig, and S. Shafee, “An improved overlapping k-means clustering method for medical applications,” Expert Systems with Applications, vol. 67, pp. 12–18, 2017.

G. O. Chagas, L. A. N. Lorena, and R. D. C. dos Santos, “A hybrid heuristic for the overlapping cluster editing problem,” Applied Soft Computing, vol. 81, p. 105482, 2019.
AUTHOR BIOGRAPHY

ATEFEH KHAZAEI is currently a researcher at the school of computer science, University College Dublin (UCD). She received her B.Sc. degree in computer engineering (1st rank) from Imam Reza International University, Mashhad, Iran, 2009; the M.Sc. degree in Software Engineering, Shiraz University, Shiraz, Iran, 2011; and a PhD degree in Computer Engineering from Yazd University, Yazd, Iran, 2018. She won the “Best PhD Candidate Researcher Award” in the Department of Electrical and Computer Engineering, Yazd University in 2015. Her research interests include Data mining, Overlapping clustering, Text mining, and Software security.

HAMIDREZA KHALEGHZADEH received the B.Sc. and M.Sc. degrees in computer engineering (software) in 2007 and 2011, respectively, and the PhD degree from the School of Computer Science, University College Dublin, Ireland, in 2019. He is currently an Assistant Lecturer at the School of Computing, University of Portsmouth. His research interests include performance and energy consumption optimization in massively heterogeneous systems, machine learning, big data, and parallel/distributed computing.

MOHAMMAD GHASEMZADEH is an Associate professor who has been working at Yazd University in Iran since 1996. He received his B.Sc. degree in computer science and engineering from Shiraz University, Shiraz, Iran, 1989, an M.Sc. degree in artificial intelligence from Tehran Polytechnic in 1995. For his PhD research program, he worked from Feb. 2002 to Feb. 2006 at the University of Trier and at HPI in the University of Potsdam both in Germany, where he received his PhD degree in Computer Science, in Nov. 2005. Currently (from Feb. 2016) he is spending his sabbatical (research leave) again at HPI at the University of Potsdam in Germany.