International Journal of Speech Technology (2020) 23:767–777
https://doi.org/10.1007/s10772-020-09728-5

An optimized iterative clustering framework for recognizing speech

Ashokkumar Palanivinayagam[1] · **Sureshkumar Nagarajan**[1]

Received: 5 March 2020 / Accepted: 15 June 2020 / Published online: 22 July 2020

Abstract
In the recent years, many research methodologies are proposed to recognize the spoken language and translate them to text. In this paper, we propose a novel iterative clustering algorithm that makes use of the translated text and reduces error in it. The proposed methodology involves three steps executed over many iterations, namely: (1) unknown word probability assignment, (2) multi-probability normalization, and (3) probability filtering. In the first case, each iteration learns the unknown words from previous iterations and assigns a new probability to the unknown words based on the temporary results obtained in the previous iteration. This process continues until there are no unknown words left. The second case involves normalization of multiple probabilities assigned to a single word by considering neighbour word probabilities. The last step is to eliminate probabilities below the threshold, which ensures the reduction of noise. We measure the quality of clustering with many real-world benchmark datasets. Results show that our optimized algorithm produces more accurate clustering compared to other clustering algorithms.

Keywords Speech document clustering · Iterative speech error correction · Similarity of documents · Probability clustering · Speech mining

1 Introduction

Speech to text convertion comes with lots of error, these errors can be corrected with document clustering, that is when the converted speech to text are stored in each document, then by using ddocument clustering, most of the errors can be ommited. Document clustering is the process of partitioning unlabeled documents into a set of clusters such that there is more similarity within a set and no (or much less) similarity between sets (Berna and Murat 2018). Document clustering is broadly used in the field of data mining in applications like information retrieval, web mining, and so on. There are various algorithms available for clustering the documents into one or more clusters such as cosine similarity, K-Means and Euclidean distance (ED). All of these algorithms treat a document as a bag of words (normally represented as a vector of words). The unique words from the document are calculated and its number of occurrences is found; based on these two values, a weighted factor is calculated. This weighted factor plays an important part in document clustering. There are various methods for finding this weighted factor such as the term frequency—inverse document frequency (TF-IDF) method, odd ratio (OR) method, and so on.

There are two categories of clustering: offline and online. Online clustering application includes web searching, product recommendation, and so on. Offline clustering is used in pattern finding (Bishop 2006) and information extraction (Powers 2011). Performance consideration for online clustering such as the speed of clustering, accuracy, and space complexity, are more important than offline clustering. The clustering algorithms can be further divided into two classifications: hard clustering (Sima and Omid 2018) such as by using K-means, which assigns a document to one cluster; and soft clustering (Smita and Sudarson 2018) or fuzzy clustering, where one or more topics are assigned to a document. The algorithm implemented in this paper comes under fuzzy clustering.

Some examples of fuzzy clustering are latent semantic indexing (Al-Zoghby and Khaled 2018), latent Dirichlet allocation (LDA) (Yang et al. 2018; Blei et al. 2003), and

✉ Sureshkumar Nagarajan
sureshkumar.n@vit.ac.in

Ashokkumar Palanivinayagam
ashokkumar.p@vit.ac.in

[1] School of Computer Science and Engineering, VIT University, Vellore, TN 632014, India

so on. These clustering algorithms work based on the concept of topic mining. Topic mining is a statistical method of finding the hidden meanings present in documents. Hidden structures can be found easily using fuzzy clustering. In this type of clustering, the probability of certain words occurring in a document is found (Blei 2012). For example, we can expect words like 'boundary', 'runs', etc. in documents which belong to the 'cricket' category. Similarly, we can expect words like 'medicine', 'vitamin', and so on in documents which belong to the 'health' category. The words 'fitness' and 'exercise' appear in both 'cricket' and 'health' categories. Thus, a document can contain multiple associated topics and, by using topic clustering, we can determine that a document contains 20% 'cricket' content and 80% health content or that the number of words belongs to the health category is four times greater than the number of words belonging to the cricket category. Using this topic modelling, we can determine the topic-word distributions while topic-document distributions occur over a document corpus (Ximing et al. 2018).

The fuzzy clustering process involves clustering of documents in five steps: (1) parsing, (2) stemming, (3) stop word removal, (4) topic mapping, and (5) visualization. Parsing is the process of generating terms in the document. A term (or word) is the smallest unit for which clustering decisions can be made. The main part of parsing consists of generating a bag of words (Ryosuke and Tu 2018) or another kind of representation such as an n-gram (Atanu et al. 2018). Stemming (Fahd Saleh and Vishal 2018) is the process of converting a word into its root form; for example, the word 'running' can be converted into its root form ' run', and the word 'climbed' can be converted into its root form 'climb'. In the next step, stop word removal (Leskovec et al. 2011), common words that are not suitable for clustering are removed. There is no standard list of stop words; different clustering algorithms use their own stop word lists to eliminate useless words. Some examples of stop words are 'the', 'is', 'from' etc. These words are removed from the bag-of-words model or N-gram model before proceeding to the next step. The first three steps are called pre-processing steps because they do not involve clustering: the documents are only prepared for clustering. The most crucial process of clustering is topic mapping, in which each term is mapped to a set of topics. This step provides some hints for predicting the topic percentage of a document. In the last step, visualization is performed by taking the hints generated by previous steps and generating a weighted matrix from which the exact topic percentage for a document is determined.

1.1 Contribution

We propose an iterative clustering technique that improves clustering efficiency by minimizing the distance between data points and the topic cluster. Our study includes the following work:

- The first step is to use topic-word modeling for clustering the data files for the first iteration. At the end of clustering a document (in each iteration), the unknown words are added into the word-topic distribution with probability values equal to the topic distribution of the corresponding document in that iteration. Iteration continues until there are no changes in the topic-word distribution and the document-topic distribution. The need for iteration ensures a supervised state is reached.
- To increase the clustering accuracy, we normalize the probability of adjacent words with multiple associated topics. If two consecutive words have the same subset of topics, we measure the highest-probability topic and increase the probability of that topic. For example, consider the sentence 'She will park the car so we can walk in the park'; after preprocessing this sentence we obtain the sentence 'park car walk park'. Here park has two different topics and the neighbor word determines the correct topic.
- While updating the topic-word distribution, we remove entries that are insufficient for making decisions in the next iteration; that is, when the probability of a word is too low (e.g., less than 0.05), we remove that word from the topic-word distribution. After removal, we update the rest of the probabilities so that the sum of all probabilities for words is 1.

The rest of the paper is structured as follows: Sect. 2 describes some existing works related to iterative clustering. Section 3 explains two key problems that arise while dealing with iterative clustering, Sect. 4 describes the solutions to the two problems encountered, and Sect. 5 shows experimental results that prove that our algorithm is more efficient than existing algorithms such as the cosine similarity and k-means.

2 Related work

The main aim of clustering documents from a corpus is to divide the documents into one or more groups. For this purpose, a one-time clustering process may not be very efficient. Clustering process efficiency is improved by running the algorithm more than once (Manochandar and Punniyamoorthy 2018). Some feedback is given at the end of each iteration to improve the quality of the next iteration; for example, some feedback is generated during a user search (Bridgid et al. 2018) and improves the efficiency of clustering. Feedback can also be generated by assigning features (Daniel Carlos et al. 2019) while clustering. Other algorithms

(Manochandar and Punniyamoorthy 2018) improve the efficiency of clustering using TF-IDF (Manochandar and Punniyamoorthy 2018; Morteza et al. 2019).

K-means (Mane and Kulkarni 2018) is widely used for clustering along with other algorithms like Euclidean distance (ED) (Kaizhu et al. 2008) which give better results for small numbers of documents. Xuejuan et al. (2018) proposes an efficient way of merging cosine similarity (Lulwah and Mourad 2018) with spherical k-means (Liang et al. 2018) for better results with large numbers of documents. For dealing with a large number of data documents, dimension reduction can also be added as a part of pre-processing as mentioned in Fuyuan et al. (2018), Tanvir Habib and Zahid (2018).

Most of the methods used in iterative clustering use the concept of a correlation coefficient matrix or an inverse relationship matrix which describes the relationship between two or more words. A general pattern is found and the relationships between these patterns and the rest of the data are calculated. This process continues until there is sufficiently close distance between the data. However, few algorithms based on these concepts have been explained in the corresponding papers (Abla Chouni et al. 2019; Roger Alan et al. (2019; Elizaveta and Vsevolod 2018).

Divisive clustering like that described in Marcos Wander et al. (2018) performs iteration in a reverse manner. It starts with a single large cluster and, in each iteration, the algorithm divides the cluster or groups into smaller groups. The division is made in such a way that the smaller groups are more dissimilar compared to other groups. The division process is performed using the variance of the data. There is larger variance between the resulting divided groups. Few papers (Marcos Wander et al. 2018) have mentioned how to divide a larger cluster into few small sets. Some concepts (Marcos Wander et al. 2018) work by transferring a few patterns from one cluster into another cluster iteratively, thus gaining couplings within items.

Other related works include clustering with many algorithms such as in Lloyd-Max algorithm (Mangi et al. 2018), Forgy approach Ryan and Jeff (2018), and so on, which works well when the number of documents is large.

3 Problem statement

In this study, we address the problem of assigning appropriate topics to a document in a corpus. We focus on representing a document in a bag-of-words model and then finding the topic correlation of the document iteratively.

We introduce our problem more formally as follows.

We consider a corpus C of N documents ($D_1, D_2, ..., D_N$). Let the topic word distribution be a list of tuples of the form $< W_i, T_i, P_i >$, $1 \leq i \leq M$, where W_i is a word, and its topic correlation is T_i with the probability P_i, and M is the number of entries in the topic word distribution. The topic-word distribution changes over successive iterations.

We aim to solve the two key problems defined below.

Problem 1 To assign approximate topics for all documents in the corpus and create a document-topic distribution. This distribution is the list of tuples of the form $< D_i, T_i, P_i >$, $1 \leq i \leq N$

Problem 2 Starting with very few records in a topic-word distribution (semi-supervised) and ending with a supervised set of records in a topic-word distribution.

To address the above two problems, we propose an iterative algorithm that modifies the traditional topic-word modeling algorithm to cluster the documents more accurately. Our algorithm works for both small and large numbers of documents.

4 Document topic distribution function

In this section, we introduce the process of iterative topic-word clustering (TWC), which assigns topic distributions for the text document. Next, we show how efficiency is improved by multi-probability normalization.

4.1 Iterative topic-word modeling

We propose an iterative clustering algorithm that modifies the TWC algorithm to produce better clustering of text documents. Practically, it is impossible to specify all of the word-topic distribution in existing clustering methods, so only a subset of a word-topic distribution can be given as input to TWC for topic mining. After topic mining, only a set of words from the document is assigned a topic distribution. Many unknown words (words which do not have any topic distribution) result. The unknown words are then given an estimated probability distribution based on the calculated probability distribution of the corresponding document. Our algorithm starts with semi-supervised input and moves towards supervised input during each iteration.

Whenever a word W is retrieved from a document D, its topic correlation is found in β. If a topic is known (if the topic exists in β), then the corresponding topic (Z) is returned. Using the topics of all known words, the topic distribution α is found by merging all the topic distributions; for example, if a document T1 has three words each with the distribution (70% A, 30% B), (80% A, 20% C), and (100% B), then the topic distribution of the document is 42.86% for topic A, 51.43% for topic B, and 5.71% for topic C. Since the number of entries in β (input distribution which contain initial probability distribution) is less than number of unique

words in the corpus, obtaining topics for all of the words is not possible, so there may be more unknown words (words which have no topics). Unknown words are represented as W′, and U represents the total number of unknown words. For each unknown word, the topics are estimated and added to the estimated distribution β'. In the above example, if an unknown word UW exists in T1, then its topics are estimated based on α; that is, the UW′ topic distribution is 0.4286 for topic A, 0.5143 for topic B, and 0.0571 for topic C. At the end of each iteration, β' is added to β. Algorithm 1 explains how iterative TWC works. The iterative TWC process is shown in Fig. 1 as a flowchart.

Algorithm 1 Iterative TWC

```
1: procedure ITERATIVETWC
2:     C ← Corpus
3:     PD ← Topic Word Distribution
4:     TW ← Topic Word Distribution of the current document
5:     begin:
6:      DocProb ← List <topic,prob>
7:     for each document D in c do
8:         int N
9:         List< String > unknown
10:        for each word W in D do
11:            P=PD(W)
12:            if P= null then
13:                add W to unknown
14:            else
15:                if P already exists in TW then
16:                    Q=P union TW(W)
17:                    Remove duplicate words and topics by keeping only the highest-
   probability items
18:                    Add Q to TW
19:                else
20:                    Add P to TW
21:        for each word W in the list of unknown words do
22:            for each entry E in TW do
23:                List <Word,Topic,Prob> un = W, getTopic(E) , getProb(E)
24:                add un to TW
25:        N=unique words in TW
26:        for each Topic T in TW do
27:            float Prob=0
28:            for each word W belonging to T in TW do
29:                Prob = Prob + (1/N) * getProbInTW(W,T)
30:            add <D,T,Prob> to DocProb
```

4.2 Multi-probability normalization

In this subsection, we introduce an optimization method that increases the efficiency of topic assignment for a document.

A word's topic is determined by its neighbor when the word has multiple meanings. For example, consider the following two sentences: 'his favorite color is rose' and 'the garden has many rose flowers'. After pre-processing, the two strings become 'favorite color rose' and 'garden rose flower'. Here the word 'rose' can be assigned two topics: '*color*' and '*flower*'. The correct topic can be assigned only with the help of neighboring words. In the first sentence, the word 'rose' can be assigned the topic of '*color*' because its neighbor word is 'color' and the word 'rose' in the second sentence can be assigned the topic of 'flowers' because of its neighbor word, 'flower'. Thus, whenever a word has more than one topic entry in its word-topic distribution, the topic can be assigned accurately with the help of its neighboring words.

$\forall$i W_i, W_{i+1}, $1 \leq i < n$, where n is the number of words in the document if $PD(W_i) \cap PD(W_{i+1}) = < T_1, P_1 >, < T_2, P_2 >, \ldots$.

Let $P_{max} = \max(P_1, P_2, P_3, \ldots)$ and T_{max} be the associated topic for P_{max}.

Let the incremental factor (IF) be the percentage of topic contributions which have the probability P_{max}

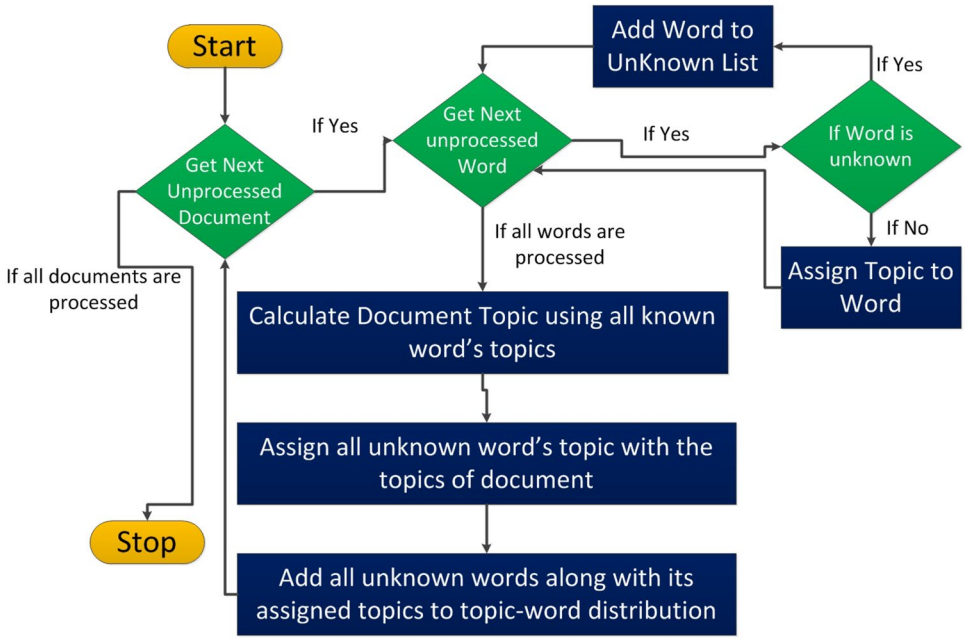

Fig. 1 Flowchat of iterative TWC for a single iteration

$$PR(W_i, T_{max}) = PR(W_i, T_{max}) + PR(W_i, T_{max}) * IF/100$$
$$PR(W_{i+1}, T_{max}) = PR(W_{i+1}, T_{max}) + PR(W_{i+1}, T_{max}) * IF/100$$
$$\forall_k PR(W_i, T_k) = PR(W_i, T_k) - PR(W_i, T_k) * (1 - IF)/100$$
$$\forall_k PR(W_i, T_k) = PR(W_i, T_k) - PR(W_i, T_k) * (1 - IF)/100$$

where W_i is a word in a document PD(W) is the probability distribution function of a word and returns the list of topics along with its probability in a tuple $< Topic, Probability >$ IF is the increment factor and PD(Word,Topic) returns or sets the probability of the topic for a word.

If two consecutive words have more than one probability assigned and if both words have the same set of probability distributions, then the highest-probability topic t_{max} is calculated and that particular topic's probability is increased by IF, where IF is the percentage of t_{max} in the corresponding document.

If any two continuous words W_i, W_j where i, j ∈ {1,..., N} have the same set of topics, then the topic with the largest probability is increased. The increase factor is equal to the percentage of the topic with the greatest probability that contributes to document M; for example, if the highest probability is 0.5, its topic contributes 37.5% and the new probability is then 0.6875.

After incrementing the topic probability by the IF, the rest of the topic probabilities are decreased by 1-IF so that the sum is always 1. In this multi-probability normalization, the probabilities of words are changed to obtain a more accurate topic assignment. First, the continuous words are checked for having common probabilities; then the maximum topic is increased by IF and the rest of the topics are decreased by (1-IF). Algorithm 2 shows the process of multi-probability normalization when checking the topic match for two consecutive multi-probability words. If a match is found, then the probabilities are adjusted by IF.

Algorithm 2 Multi-probability normalization

```
procedure MULTIPROBNORMALIZATION
    Wi ← Word
    Wj ← Word
    begin:
    List<topic,prob> pd1 = getTopicDist(Wi)
    List<topic,prob> pd2 = getTopicDist(Wj)
    List<topic,prob> common=pd1 intersect pd2
    if common != null then
        pmax = maximum probability in common
        tmax = corresponding topic for pmax
        IF = obtain topic distribution from iterative TWC (algorithm 1)
        for each item in pd1 do
            if getTopic(item)=tmaxl then
                update prob. p=p + p *IF / 100
            else
                update prob. p = p - p * IF / 100
        for each item in pd2 do
            if getTopic(item)=tmax then
                update Prob. p = p+ p * IF / 100
            else
                update Prob. p=p-p* (1-IF) / 100
```

4.3 Removing noise from iterations

The advantage of using iterations in clustering is that we obtain more accurate clustering results in each iteration. The results from previous iterations can be used to cluster the document in the current iteration more efficiently as long as the results are reliable. Data that are not reliable are called noise, which decreases the efficiency of the iteration and thus leads to more iterations. Therefore, to obtain an optimal clustering result, noise should be removed so that it does not propagate into future iterations and result in lower performance.

During the implementation of the above two algorithms, we found that some words with multiple topics lead to faster noise generation. Noise in the above algorithm is defined as topics with very low probabilities. These noisy words must be found and their corresponding noisy topic distributions removed so that future iterations are safe from noise.

To eliminate noise, a significant threshold value is normally used. If the topic distribution values fall below this threshold, then the topic is removed completely from the distribution. The probability of the rest of the entities is adjusted (increased) so that the summation is always 1.

Whenever a probability falls below the threshold value (TH), the probability is considered to be zero. This prevents noise from propagating to higher iterations. Whenever the probability is reduced to zero, the other probabilities belonging to the same topic are increased so that the summation of probabilities is always equal to 1.

5 Experiments and results

We implemented a hybrid algorithm of all three above algorithms: Algorithm 1 (iterative TWC), Algorithm 2 (multi-probability normalization), and Algorithm 3 (noise removal). Three benchmark datasets were used for evaluation: 20 newspaper, Patent, and Reuters. We compared our algorithm's results with those of two other algorithms (K-Means and cosine similarity) and found that the proposed method has better clustering results. That is, the average distance between documents within a cluster is reduced by

Table 1 Data set description

Dataset	Unique documents	Unique words	Number of topics used
20 newspaper	8000	23,642	4
Patent	28,395	179172	5
Reuters	5000	20,050	5

Table 2 Initial word probability distribution details and maximum number of iterations

Dataset	Number of topics	Average number of topic-word entries	Number of iterations
20 newspaper	5	18	5
Patent	6	21	5
Reuters	5	20	5

Topic A	Topic B	Topic C	Topic D	Topic E
molecules	Protein	sampling	data	government
nucleus	Vaccines	clocks	transfer	invention
material	Cancer	frequency	games	rights
sodium	Tumor	Heat	card	Institutes
phosphate	prevention	Shock	Fragment	agents

Fig. 2 Sample word-topic correlation—words are taken from 20 newspaper dataset

using our algorithm in comparison with using K-Means and cosine similarity.

The Table 1 shows the three data sets and their associated characteristics used for our experiment.

5.1 Pre-processing

All three datasets underwent the pre-processing step. Four levels of pre-processing were performed: (1) stop-word removal, (2) stemming, (3) weblink removal, and (4) file removal (Uysal and Gunal 2014).

Stop word removal includes removal of common words like 'the', 'that', 'she', 'of' , and so on. Stemming involves the conversion of original words into their root forms; for example, 'sleeping' is converted to 'sleep'. Both the Patent and Reuters datasets contain numerous weblinks such as URLs, email addresses, and other time-based information like dates and times of news articles, last edited times, etc., and these formations are also removed from the dataset. The last step is file removal, in which some files not used for clustering are removed. For example, some files less than 20 KB in the Patent dataset were removed automatically through pre-processing, and these files were insufficient for topic assignment.

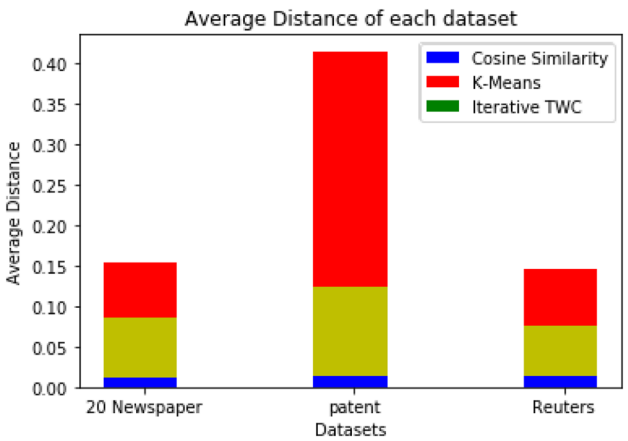

Fig. 3 Average distances for all the three datasets with the three algorithms: cosine similarity, K-Means, and iterative TWC

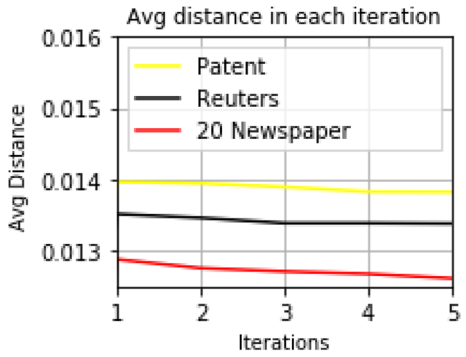

Fig. 4 Average distance between the documents from the three datasets in each iteration

In the 20 newspaper dataset, we considered 8000 random files from 20 clusters. Pre-processing tasks included the removal of headers, footers, and words that contain dates, email addresses, contact information, and addresses. Next, all files were pre-processed using the stop word removal and stemming algorithms.

The patent dataset contained a total of 48,213 files, out of which only 29,847 files were in the English language. 28,395 files were considered suitable for clustering because their sizes were greater than 20 KB. These files were in XML format, so the first step was to convert them from XML into text by removing all tags and meta information. After this step, the final step was to perform stop word removal and stemming.

The Reuters dataset contains numerous news articles, from which we first removed all links and kept only the text content. We then performed stop word removal and stemming.

5.2 Initial topic-word probability function

For each dataset, we used different topic-word probability functions for the purpose for training. The training data consisted of a very low number of entries, as shown in the Table 2. As iteration proceeded, the number of entries in the probability function grew and became stable at higher iterations.

Figure 2 shows the sample word-topic correlation used in our clustering process.

5.3 Measuring distance

The ultimate goal of a cluster is to group the documents into one or more groups. The cluster is considered to be efficient when the distance between the documents and the document head (the cluster point) is very small. In our experiment, we measured the distance between the documents and the cluster point as follows:

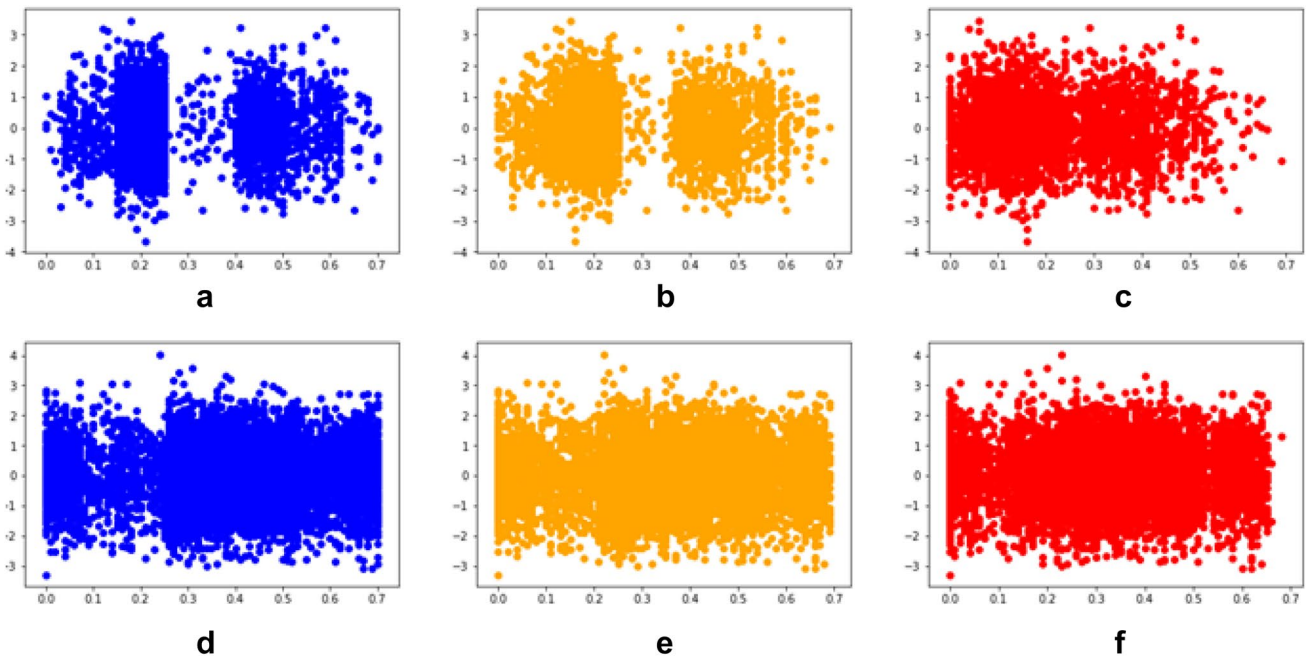

Fig. 5 **a**–**c** show the last three iterations (the third, fourth, and fifth iterations) of topic A and **d**–**f** show the last three iterations (third, fourth, and fifth iterations) of topic B using iterative TWC in the Patent dataset. The figure shows that, as iterations proceed, the documents move closer to the cluster point (0,0). Random values are taken for the Y axis for the purpose of illustrating the spread of data and the X-axis denotes the original distance. For example, if a document had 0.7 probability of a topic, then its distance was 0.3

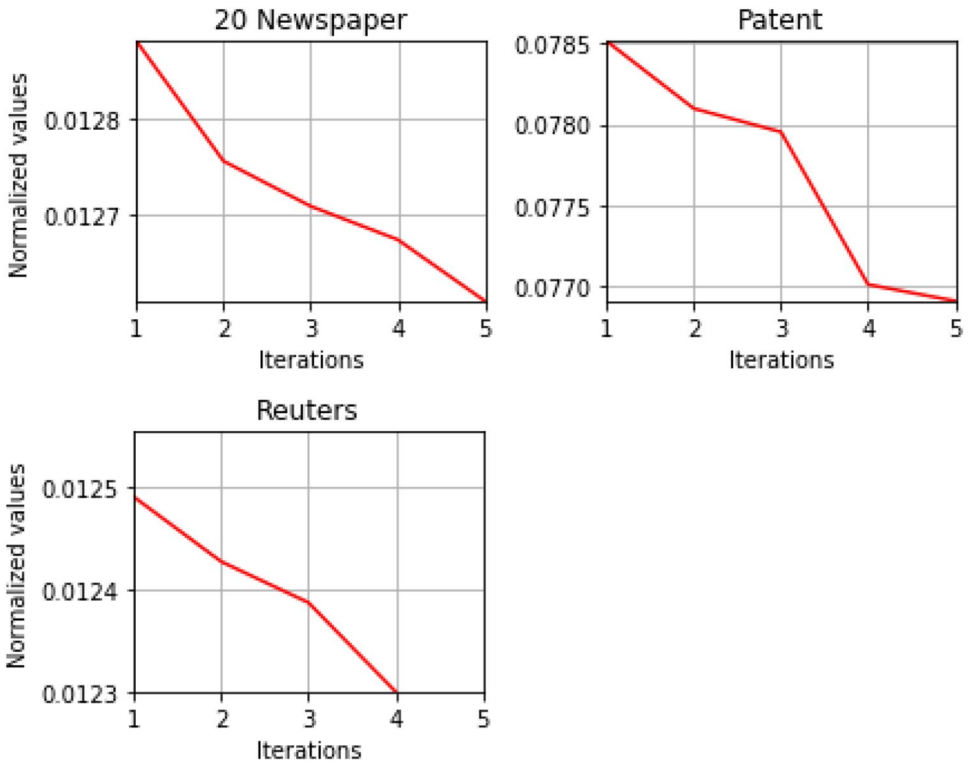

Fig. 6 A topic convergence graph showing the reduction of multiple topic assignments as iterations proceed

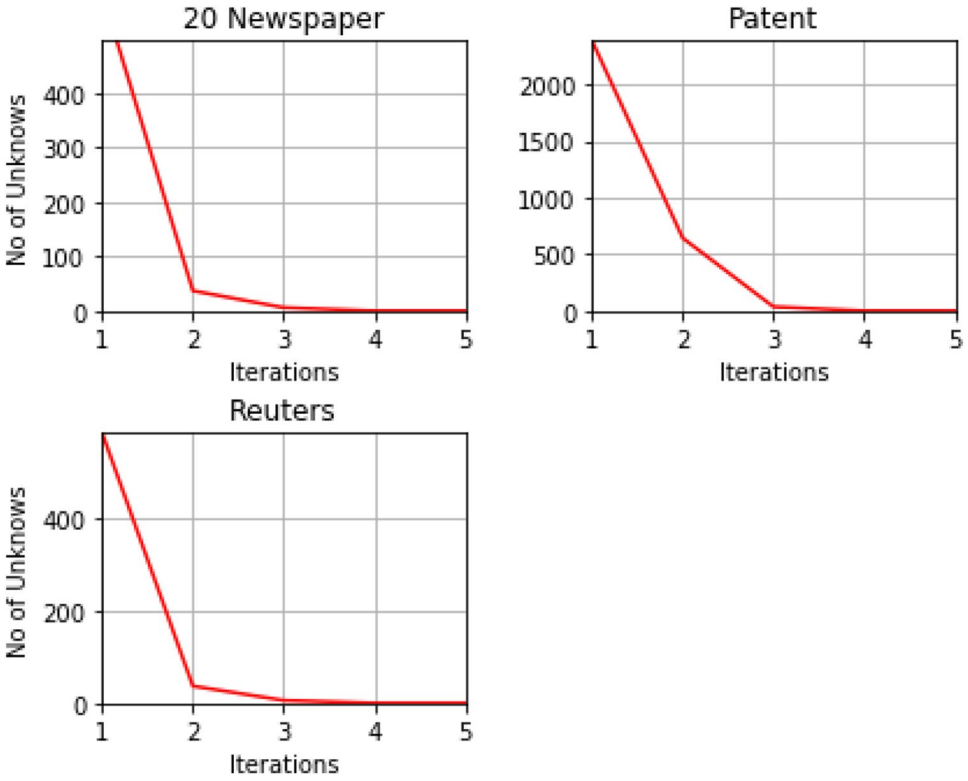

Fig. 7 Number of unknown words in each iteration

- In the cosine similarity method, we obtained the similarity value between each pair of documents and took the average of all the values to calculate the overall average distance.
- In iterative TWC, we measured the distance using the probability value, i.e., if a document had 70% (0.7) topic A content, then we took the distance as 1–0.7 = 0.3. That is, the document was 70% similar to (or 30% away from) topic A.

Figure 3 shows the overall clustering efficiency for all three datasets. The graph shows that the average distance of our algorithm (at iteration 5) was much less than that for the other two algorithms.

In the graph in Fig. 3, the values represent the results obtained in the fifth iteration, and Fig. 4 visualizes the results at each iteration (Fig. 5).

One of the advantages of our algorithm is the learning process used; that is, it starts from a semi-supervised state and ends in a supervised state. The word-topic convergence is a good measure for testing this property and is illustrated by the graph in Fig. 6. The values in the graph show the average number of multiple topics associated with each word at each iteration. As shown in Fig. 6, when iterations proceed, the topic convergence decreases; that is, words with excessive topics are removed and, with further iteration, incorrect topics are removed.

A supervised state is one in which there is no uncertainty; that is, there are no words with zero topics. Put differently, there are no unknown words in the dictionary of the topic-word distribution. Our algorithm's learning process and noise reduction process performs this step by eliminating the number of unknown words at each iteration. At one point, the unknown word count becomes zero, and a supervised state is achieved. Figure 7 shows the decrease of the number of unknown words with iteration of the algorithm.

Our algorithm has the ability to learn from mistakes. That is, if any word-topic distribution is added incorrectly to our algorithm, the algorithm produces a false positive; but as iteration proceeds, the wrong word's probability decreases. At one stage, the probability falls below the threshold value and the entry is removed from the table, producing the correct result. The table shows the distance between the documents with the correct topic at each iteration. Few words are assigned wrong probabilities; that is, few words are inserted into the topic-word distribution in such a way that the correct topic has less probability while the wrong topics have greater probability. The Algorithm 2 decrements the wrong topic's probability and, at each iteration, the correct topic probability is increased.

Figure 5 shows the documents (in the Patent dataset) plotted in two topic graphs. The first row represents a total of 3151 documents associated with topic A in the last three iterations. The second row represents a total of 8028 documents associated with topic B in last three iterations. The figure clearly shows the documents moving towards the cluster point in each iteration (i.e., reducing the average distance).

6 Conclusion

In this study, we focused on developing an optimized algorithm that clusters documents iteratively and aims to efficiently assign topics to documents such that the distance between the topic head or cluster head and the documents is as low as possible so that there is minimal error in speech to text conversion. We stores all the speech to text data into a document and then we developed an iterative algorithm that starts in a semi-supervised state at the first iteration and learns the semantics of the documents automatically, reaching a supervised state at future iterations. Moreover, our algorithm learns to remove noise during intermediate iterations without allowing noise to propagate to subsequent iterations. We implemented this algorithm with three real-world benchmark datasets and compared the results with other existing algorithms such as cosine similarity and K-Means. Our algorithm outperforms alternative methods in terms of clustering efficiency.

In future work, it will be interesting to reduce the learning time (number of iterations required) and cluster documents in a single pass. This can be done by generating a universal pre-learned model that can be used directly to cluster documents from any data set.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Abla Chouni, B., Asmaa, B., & Imane, B. (2019). A survey of clustering algorithms for an industrial context. *Procedia Computer Science*, *148*, 291–302. https://doi.org/10.1016/j.procs.2019.01.022.

Al-Zoghby, A. M., & Khaled, S. (2018). Ontological optimization for latent semantic indexing of arabic corpus. *Procedia Computer Science*, *142*, 206–213. https://doi.org/10.1016/j.procs.2018.10.477.

Atanu, D., Mamata, J., & Jitesh, J. (2018). Senti-N-Gram: An n-gram lexicon for sentiment analysis. *Expert Systems with Applications*, *103*, 92–105. https://doi.org/10.1016/j.eswa.2018.03.004.

Berna, A., & Murat, C. G. (2018). Semantic text classification: A survey of past and recent advances. *Information Processing & Management*, *54*(6), 1129–1153. https://doi.org/10.1016/j.ipm.2018.08.001.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.

Blei, D. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77–84.

Blei, D. M., Ng, A. Y., & Jordan, M. (2003). I: Latent Dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Bridgid, F., Ruthann, T., & Katherine, A. (2018). Learning more from feedback: Elaborating feedback with examples enhances concept learning. *Learning and Instruction*, *54*, 104–113. https://doi.org/10.1016/j.learninstruc.2017.08.007.

Daniel Carlos, G. P., Ying, W., Alexandro, B., & Chaohuan, H. (2019). Semi-supervised and active learning through Manifold Reciprocal kNN graph for image retrieval. *Neurocomputing*, *340*, 19–31. https://doi.org/10.1016/j.neucom.2019.02.016.

Elizaveta, K. M., & Vsevolod, I. T. (2018). Text clustering as graph community detection. *Procedia Computer Science*, *123*, 271–277. https://doi.org/10.1016/j.procs.2018.01.042.

Fahd Saleh, A., & Vishal, G. (2018). A cognitive inspired unsupervised language-independent text stemmer for Information retrieval. *Cognitive Systems Research*, *52*, 291–300. https://doi.org/10.1016/j.cogsys.2018.07.003.

Fuyuan, C., Joshua Zhexue, H., Jiye, L., Xingwang, Z., Yinfeng, M., Kai, F., et al. (2018). An algorithm for clustering categorical data with set-valued features. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(10), 4593–4606. https://doi.org/10.1109/TNNLS.2017.2770167.

Kaizhu, H., Haiqin, Y., Irwin, K., & Michael, R. (2008). Maxi-min margin machine: Learning large margin classifiers locally and globally. *IEEE Transactions on Neural Networks*, *19*(12), 260–272. https://doi.org/10.1109/TNN.2007.905855.

Leskovec, J., Rajaraman, A., & Ullman, J. D. (2011). Data mining. *Mining of Massive Datasets*,. https://doi.org/10.1017/cbo9781139924801.002.

Liang, B., Jiye, L., & Yike, G. (2018). An ensemble clusterer of multiple fuzzyk-means clusterings to recognize arbitrarily shaped clusters. *IEEE Transactions on Fuzzy Systems*, *26*(6), 3524–3533. https://doi.org/10.1109/TFUZZ.2018.2835774.

Lulwah, A., & Mourad, Y. (2018). Interest-based clustering approach for social networks. *Arabian Journal for Science and Engineering*, *43*(2), 935–947. https://doi.org/10.1007/s13369-017-2800-z.

Mane, D. T., & Kulkarni, U. V. (2018). Modified fuzzy hypersphere neural network for pattern classification using supervised clustering. *Procedia Computer Science*, *143*, 295–302. https://doi.org/10.1016/j.procs.2018.10.399.

Mangi, K., Jaelim, A., & Kichun, L. (2018). Opinion mining using ensemble text hidden Markov models for text classification. *Expert Systems with Applications*, *94*, 218–227. https://doi.org/10.1016/j.eswa.2017.07.019.

Manochandar, S., & Punniyamoorthy, M. (2018). Scaling feature selection method for enhancing the classification performance of Support Vector Machines in text mining. *Computers & Industrial Engineering*, *124*, 139–156. https://doi.org/10.1016/j.cie.2018.07.008.

Marcos Wander, R., Seiji, I., & Luiz Enrique, Z. (2018). Educational data mining: A review of evaluation process in the e-learning. *Telematics and Informatics*, *35*(6), 1701–1717. https://doi.org/10.1016/j.tele.2018.04.015.

Morteza, Z., Anteneh, A., Xing, Z., Heidar, D., & Aijun, A. (2019). A utility-based news recommendation system. *Decision Support Systems*, *117*, 14–27. https://doi.org/10.1016/j.dss.2018.12.001.

Powers, D. M. W. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, *2*, 37–63.

Roger Alan, S., Patricia, A. J., & João Francisco, V. (2019). An analysis of hierarchical text classification using word embeddings. *Information Sciences*, *471*, 216–232. https://doi.org/10.1016/j.ins.2018.09.001.

Ryan, M., & Jeff, B. (2018). Towards justifying unsupervised stationary decisions for geostatistical modeling: Ensemble spatial and multivariate clustering with geomodeling specific clustering metrics. *Computers & Geosciences*, *120*, 82–96. https://doi.org/10.1016/j.cageo.2018.08.005.

Ryosuke, M., & Tu, B. (2018). Semantic term weighting for clinical texts. *Expert Systems with Applications*, *114*, 543–551. https://doi.org/10.1016/j.eswa.2018.08.028.

Sima, S., & Omid, F. (2018). Run-time mapping algorithm for dynamic workloads using association rule mining. *Journal of Systems Architecture*, *91*, 1–10. https://doi.org/10.1016/j.sysarc.2018.09.005.

Smita, C., & Sudarson, J. (2018). Correlation based feature selection with clustering for high dimensional data. *Journal of Electrical Systems and Information Technology*, *5*(3), 542–590. https://doi.org/10.1016/j.jesit.2017.06.004.

Tanvir Habib, S., & Zahid, A. (2018). An analysis of MapReduce efficiency in document clustering using parallel K-means algorithm. *Future Computing and Informatics Journal*, *3*(2), 200–209. https://doi.org/10.1016/j.fcij.2018.03.003.

Uysal, A. K., & Gunal, S. (2014). The impact of preprocessing on text classification. *Information Processing & Management*, *50*(1), 104–112. https://doi.org/10.1016/j.ipm.2013.08.006.

Ximing, L., Ang, Z., Changchun, L., Jihong, O., & Yi, C. (2018). Exploring coherent topics by topic modeling with term weighting. *Information Processing & Management*, *54*(6), 1345–1358. https://doi.org/10.1016/j.ipm.2018.05.009.

Xuejuan, L., Jiabin, Y., & Hanchi, Z. (2018). Efficient and intelligent density and delta-distance clustering algorithm. *Arabian Journal for Science and Engineering*, *43*(12), 7177–7187. https://doi.org/10.1007/s13369-017-3060-7.

Yang, L., Wenming, Z., Zhen, C., & Tong, Z. (2018). Face recognition based on recurrent regression neural network. *Neurocomputing*, *297*, 50–58. https://doi.org/10.1016/j.neucom.2018.02.037.