Journal of Theoretical and Applied Information Technology
15th September 2020. Vol.98. No 17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

# A COMPARISON STUDY OF DOCUMENT CLUSTERING USING DOC2VEC VERSUS TFIDF COMBINED WITH LSA FOR SMALL CORPORA

**[1]AMALIA AMALIA, [2]OPIM SALIM SITOMPUL, [3]ERNA BUDHIARTI NABABAN, [4]TEDDY MANTORO**

[1]Department of Computer Science, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Medan, Indonesia
[2,3]Department of Information Technology, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Medan, Indonesia
[4]Department of Computer Science, Universitas Sampoerna, Jakarta, Indonesia

E-mail: [1]amalia@usu.ac.id

## ABSTRACT

The selection of a suitable word vector representation is one of the essential parameters in document clustering because it affects the performance of clustering. The excellent word vector representation will generate a good clustering result, even only using the simple clustering algorithm like K-Means. Doc2Vec, as one of word vector representations, has been extensively studied in large text datasets and proven outperforms the performance of traditional word vector representation in document categorization. However, only a few studies analyze word vector representations of small corpora. As appropriate, learning observation in a small corpus is also crucial because, in some cases, a large corpus was not always available, particularly in some low-resources languages like Bahasa Indonesia. Moreover, the clustering of the small datasets also plays essential roles in pattern recognition and can be an initial step to implement the analysis result in a more significant corpus. This study is an experimental study that aims to explore more in-depth exploration to compare document clustering using Doc2Vec versus TFIDF-LSA for small corpora in Bahasa Indonesia. In this study, the quality of word vector representation is measure by the cluster performance using intrinsic and extrinsic measurements. The study also considers measuring word representation based on time and memory consumption. This study also concerns with getting an optimal word vector representation by tuned appropriate hyper-parameter. The word vector representations were tested to various sizes of the small corpora using the K-Means algorithm. The result of this study, a TFIDF-LSA gets a better cluster performance; meanwhile, the Doc2Vec model gets a better time and memory usage efficiency.

**Keywords:** *Clustering, Word Vector Representation, Word Embedding, Clustering Comparison, Small Corpora*

## 1. INTRODUCTION

Along with the growth of the internet makes the growth of data text on the internet has explosive as well. Therefore, automation of document categorization, such as classification and clustering is an essential task for nowadays. The drawback of supervised text categorization like classification is the necessity of annotated linguistic resources. This is a challenge for languages that do not have adequate available annotated linguistic like Bahasa Indonesia. The task to manually annotate the linguistic resources from scratch like an annotated corpus requires many times and human labors. Therefore, text categorization using a clustering approach can be the best solution. Document clustering is an unsupervised machine learning techniques to automatically group documents into clusters based on document similarity [1], [2]. Clustering is one of the most important tasks in data analysis [3]. Clustering does not need pre-defined labels for each group by human labor. Many algorithms for document clustering have been proposed, for instance, K-Means [4], K-Means++ [5], Self Organizing Map [6] et cetera. In addition to the choice of cluster algorithm, the other essential aspects of generating good clustering performance are selecting appropriate word vector representation. Word vector representation is a feature representation for data input in textual form. Word

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

vector representation, also known as text representation, in this study, we used these two terms interchangeably. Text representation is a pre-processing stage in machine learning for data input in a textual form so that the data able to further processed in machine learning. One of the simple methods in word vector representation is Bag of Words (BoW). The instances of BoW methods are *one hot encoding* and term-frequency inverse-document-frequency (TFIDF) [7]. The BoW is a count-based text representation method that treats documents as a set of distinct atomic words; therefore, this method cannot preserve the semantic and syntaxis information. It means that relationships between words, such as synonyms, are not incorporated [1]. Another drawback of BoW is sparsity and generate high dimensional vectors. Despite many deficiencies, the BoW is still widely used because of its simplicity and surprising accuracy. Many previous studies tried to anticipate the lack of BoW. For instance, a study by [8] implemented Latent Semantic Analysis (LSA) in TFIDF to capture semantic and dimension reduction. Implementation LSA in TF-IDF proven to increase cluster performance. Another previous study by [9] added a synonym vocabulary checked in the pre-processing stage to anticipate the synonym deficiency in TFIDF representation. Both of these studies were implemented in small corpora and gained Purity about 75%.

Beside BoW, another alternative concept of word vector representation is word embeddings. A word embedding is a low-dimensional, dense, and real-valued vector representation [10]. A word embedding is generated using a prediction-based by neural network approach based on learning representation in a large text corpus. This process generated word vector representation that captures syntactic and semantic aspects [11]. Therefore, words that have similarity meaning and close correlation will have similar word vector representation as well. Many previous studies in Natural Language Processing (NLP) stated word embeddings representation increases NLP task performance, including document categorization [11]. Using word embedding certainly added advantage for the document clustering process, because the same words in documents with similar topic tend to have similar word vector representation. Word embedding models have been researched in previous studies by [12] [13] and proven to outperform BoW for various NLP tasks. The instances of word embedding models are word2vec for word-level representation and Doc2Vec for document-level representation. In document categorization, the Doc2Vec yields higher classification accuracy than other document representation methods in various domains, such as sentiment classification, news categorization, and forum question duplication [14]. However, most studies that stated word embedding outperforms BoW using a large text corpus [15] to implement the learning process. Compare to the BoW model; word embedding certainly has the advantage if implemented in a large dataset, since word embedding does not consume as much memory as some classic methods like TFIDF and LSA [15] make many researchers implemented as much as data corpus for training. Moreover, the assumption that more data is better for catch semantic and syntactic information makes learning representation trained from a large corpus containing about billions of tokens. However, it is still unclear how many documents in the corpus does the embedding model needs in generating a good word embedding representation. There are only a few studies analyze semantic representations of small corpora [15]. It should be more observation of small corpora, particularly in some cases, large corpora not be always available [16], for examples in some low-resources languages [17] or in domain-specific like medical corpus [18]. Moreover, the clustering of the small datasets also plays important roles in pattern recognition [19], where we can predict the behavior of the unseen data from data training. This task is also referred to as the learning process [20]. Observation in the small dataset also can be an initial step to implement the analysis result to a bigger corpus. Even though clustering does not need labeled data for training, but the ground truth label is still needed to measure the cluster performance accuracy; therefore, to analyze first in a small corpus is the right decision.

The research question, is word embeddings outperform TF-IDF and LSA representation for small corpora? It is still not clear whether word embedding outperforms classical models in the small corpus. A previous study by [21] found LSA produced better performances than word embedding models in small to medium size of the training corpus. However, a previous study by [22] concluded that the corpus size is not always the main parameter in generating right word embedding. This study also revealed that the word2vec model could extract linguistic information from a small domain-specific corpus and get a satisfactory result.

Based on these two contrary statements, we want to explore more in-depth exploration in which text representation methods are better for small corpora. Unfortunately, to obtain the best suitable word

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

representation is not easy; there is no universally good representation, the choice of representation also determined by domain knowledge [20]. However, we can measure the quality of word representation based on the performance of the clustering result. If the word vector representation is right, even the simple clustering algorithm like K-Means will find the data cluster pattern and generate a good clustering result [20].

In this study, to explore more in-depth exploration in which text representation methods are better for small corpora, we compare two model word vector representations which are classical model: TF-IDF combined with LSA versus word embedding model: Doc2Vec. To simplify, we used the terms TFIDF-LSA to mention TF-IDF combined with LSA. We generated a small corpus that contains only 500 articles in Bahasa Indonesia. We also split the corpus into three smaller sub-corpora that contain 60, 125, and 250 articles. Further, to find the performance of clustering, the K-Means algorithm was implemented for each model. To measure the cluster performance, we used intrinsic and extrinsic cluster evaluation measurements. We implemented the Silhouette Coefficient evaluation for intrinsic measurement. For extrinsic cluster performance measurement, we implemented Purity and adjusted random index (ARI) evaluation. Besides the clustering performance, in this study, the comparison is also concern with the clustering processing time and memory consumption for each model. This study also concerns to tune various parameter initialization for each model to get the best model.

Our contribution is the comparison result based on a quantitative experiment that can consider other researchers in choosing the suitable word representation for a small corpus, particularly for Bahasa Indonesia based on cluster performance, time, and memory consumption. The small dataset observation can also reveal the quality of word representation for low-resources language like Bahasa Indonesia. As far as our knowledge, there is still no previous observation to compare word vector representation in the small corpus for Bahasa Indonesia.

The rest of this paper is organized as follows: In Section 2, we described the literature review. In Section 3, the related works by previous studies were described. In Section 4, we explain the methodology of this study, parameter initialization, and cluster performance measure. The experimental results are discussed in Section 5. Finally, in Section 6, we conclude the current work with a few future research directions.

## 2. LITERATURE REVIEW

Word vector representation for data text in machine learning is a process to transform input objects into numbers or vector. In the NLP task such as classification and clustering, this process is an essential process because it affects the performance and result. In this study, we compare two kinds of feature representations in the document clustering task, which are TFIDF-LSA and Doc2Vec model.

### 2.1 TFIDF-LSA

TFIDF [7] is one method to represent text documents into a vector. TFIDF contains two calculation which is *TF* and *IDF*. *TF* or term frequency is the number of times that the term *t* occurs in document *d*. The more frequencies, the more value of *TF*. Meanwhile, *IDF* or inverse document frequency is a calculation of the number of document *D* that contain the term *t* in a whole corpus. The formula of TFIDF describes in formula 2.

$$tfidf(t, d, D) = tf(t, d). idf(t, D) \quad (1)$$

The TFIDF generates high value for the essential terms in a document and filters out the common terms that occur in many documents. The feature representation in this method comes in feature matrix with dimensional as many as the total unique words number in a whole corpus. Some pre-processing stages are implied to reduce the dimension of the vector. For instance, to reduce dimensionality, a threshold cut-off is used to use only those words with high values. TFIDF is one of the BoW which treats a word as an atomic unit without considering the relationship of the word to others in the corpus. Therefore, this method cannot handle the meaning representation, such as synonyms are not incorporated. To account for the meaning representation, the TFIDF can be combined with Latent Semantic Analysis (LSA) [23] [24]. LSA is one of the most used methods for word meaning representation in BoW representation. Besides handling meaning representation, LSA can also be used to reduce the dimensionality of TFIDF matrices. A dimensionality reduction is implemented by a truncated Singular Value Decomposition, SVD, which projects every word in a subspace of a pre-defined number of dimensions. Once the vectorial representation of words is obtained, the semantic similarity between two terms is typically computed by the cosine of the angle between them [15].

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

### 2.2 Doc2Vec

Word embedding is a method to transform the text into a dense vector in real numbers using a neural network approach. A study by [10] in the form of a feed-forward neural network language model as one of the pioneers the word embedding. A study by [25] proposed a simpler method that produces high-quality vectors called word2vec. Word embedding by word2vec begins with the learning process in a collection of text documents or corpus. The learning process aims to exploits the statistical properties of the textual structure in a vectorial space. Words with similar meanings tend to be located close to each other in vectorial space. This hypothesis relies on the idea that words with similar meanings tend to occur in similar contexts [26]. The learning process has collected the information about neighbor words (context words) of each word (target word) in the corpus.

Further, fully connected feed-forward neural networks are implemented to predict context words based on the target word or vice versa. The aim is to get the optimal prediction based on the information on the learning corpus stage. Words embedding are trained using stochastic gradient descent, and the gradient is obtained via backpropagation. The weight is adjusted to get the optimal prediction. Word embeddings are these adjusted weights. Once the neural network has been trained, the learned linear transformation in the hidden layer is considered the word representation [15]. There is two architecture in word2vec, which are Continuous bag-of-words (CBOW) and Skip Gram. CBOW architecture was implemented to predict the target word based on the context word. Meanwhile, Skip Gram is an architecture to predict context words based on the target words. The result of the word embedding the words with a similar meaning is mapped to a similar position in the vector space [27]. Word2vec is a tool to provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words [28]. This tool learns to projects words into a latent d-dimensional space with n-window size. Window size is a parameter to determine the number of context words. Also, some researchers try to build model embedding at the document level to extend the level of feature representation in some NLP tasks. However, when using word embedding models to create document-level representations, the word vectors need to be aggregated somehow. A general approach to calculate document embedding is to simply estimate the word vectors' mean for all terms in the document [29]. Another method to train the document level is using the Doc2Vec algorithm. Doc2Vec algorithm is extended to the word2vec algorithm by (Le & Mikolov, 2014). Doc2vec was built based on paragraph vector, an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents [30].

The primary task of document embedding is to determine an appropriate distributed representation for a single document by learning a neural network with the word's information and its surrounding words in the document. The vector representation is trained to predict words in a paragraph. The Doc2Vec algorithm concatenates the paragraph vector with several word vectors from a paragraph and predicts the following word in the given context. In the Doc2Vec approach, each document has its own vector values in the same space as for words. Thus, the distributed representation for both words and documents are learned simultaneously. There are two algorithms of Doc2vec, which are the Distributed Memory Model of Paragraph Vectors (PV-DM) and the Distributed Bag of Words version of Paragraph Vector (PV-DBOW). PV-DM is a document-embedding algorithm that randomly takes sequence words from a document and tries to predict a target word from the randomly sampled set the context words as input. Meanwhile, PV-DBOW is an algorithm to generate document embedding without considering word order in documents. There is some hyper-parameter that needs to be tuned to generate a good vector for document embedding, such as the window size, dimension, and minimum words. With the right parameter, feature representation for the document will increase document clustering.

### 2.3 Document Clustering with K-Means

Clustering is an unsupervised process to groups a set of objects based on the similarity between the objects. A good cluster will split the collection of data objects into *k* clusters. The data objects that are similar to one another will be grouped in one cluster, and the different data objects will be put in a different cluster. One of the methods to document clustering is using the K-Means algorithm [20]. K-means clustering is one of partitioning hard clustering methods that split a given dataset into a fixed number (*k*) of clusters. In K-Means, we have to set *k* value (number of clusters) in the first process. Based on the number of *k*, the K-Means will choose *k* numbers randomly as centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids. The process of *centroid* adjustment is repeated until the values of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the *centroids* stabilize. The final *centroids* will be used to produce the final clustering of the input data, each with a class identity.

### 2.4 Cluster Evaluation Measure

Cluster evaluation measurement of a clustering algorithm is essential as the algorithm itself. A clustering evaluation demands an independent and reliable measure for assessing and comparing clustering experiments and results. However, cluster evaluation not as trivial as classification evaluation. Evaluating a clustering algorithm's performance is not just as counting the number of correct clustering like precision and recall measurement of a supervised classification algorithm. In clustering, the good cluster based on an internal and external criterion. The text document that grouped in the same cluster or intra-cluster should have high similarity. On the other hand, documents in different clusters or inter-cluster should be dissimilar documents. In cluster evaluation, any evaluation metric should not take the cluster labels' absolute values into account. However, the cluster evaluation is more to define separations of the data similar to some ground truth set of classes. Members belonging to the same class are more similar than members of different classes according to some similarity metric [31]. There are two kinds of methods in document clustering, namely intrinsic and extrinsic measures. Intrinsic or internal measures of quality such as distortion or log-likelihood to indicate how well an algorithm optimized a particular representation. Intrinsic comparisons are inherently limited by the given representation in other words dependent on the feature representations, therefore intrinsic can not compare between different representations [32]. Intrinsic measures, calculate the cluster separation, and cohesion. The advantage of this method, it does not require a ground truth label. The example of this method is the Silhouette coefficient. The silhouette coefficient calculates how similar an object to its own cluster (cohesion) compared to the different clusters (separation). In other words, the silhouette coefficient (S) is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample in the cluster (C). The best value of the silhouette coefficient is 1, and the worst value is -1. The formula of silhouette can be described in formula 2.

$$s = \frac{b-a}{\max\{a,b\}}, if\ |C| > 1 \qquad (2)$$

On the other hand, the extrinsic or external measure is able to compare the clustering result from different feature representation methods. This method measures of quality compare a clustering to an external knowledge source such as ground truth labels. Examples of extrinsic measures are Purity, Random Index, and Adjusted Random Index (ARI). Purity is a simple and transparent evaluation measure. Purity's formula is to calculate the total number of the most frequent object class in each cluster and then divided by the total number of objects (N). The formula of Purity can be seen in formula 3. Where $M$ is a set of clusters and $D$ is a set of classes.

$$P = \frac{1}{N}\sum_{m\in M}\max_{d\in D}|m \cap d| \qquad (3)$$

The highest purity score is 1, which means each document in its cluster, and the worse purity score is 0. Purity cannot be used to calculate the quality of clustering against the number of clusters.

Random Index is a method to calculate quality clusters based on the benchmark classifications. On the other hand, *RI* is the percentage of correct decisions made by the algorithm. The formula of *RI* described in formula (4).

$$RI = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}} \qquad (4)$$

Where, $n$ is a set of elements, $a$ is a number of pairs of elements in *S* that are in the same subset in *X* and the same subset in *Y*. Meanwhile, $b$ represents the number of pairs of S elements in different subsets of *X* and different subsets of *Y*. The value $c$ is several pairs of elements in *S* that are in the same subset of *X* and different subsets of *Y*, and $d$ represents the number of pairs of elements in *S* that are in different subsets of *X* and the same subset of *Y*. The *RI* score between 0 and 1. The drawbacks of *RI* measurement are that the *RI* does not has a constant baseline, implying that these measures are not comparable across clustering methods with different numbers of clusters. The raw *RI* score is then adjusted for chance into the ARI score using the following scheme:

$$ARI = (RI - Expected\_RI) / (\max(RI) Expected\_RI) \qquad (5)$$

The *ARI* values between 0 and 1, with 1 representing identical partitions, and is adjusted for the number of partitions in X and Y

## 3. RELATED WORKS

A study by [8] experiments by trying various pre-processing processes and various clustering

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

algorithms to increase clustering accuracy. This study is an experimental study to observe the suitable clustering process for Bahasa Indonesia with a small corpus that contains only 100 documents. The experiments were conducted for several cluster algorithms, such as K-Means, K-Means++, and Agglomerative, with various pre-processing stages. The results of this study indicate that the K-Means and K-Means++ algorithms are superior. This study also concludes the TFIDF-LSA pre-processing produces the best cluster purity for $k = 2$, and followed by $k = 3$ and $k = 4$. It is shown that the more clusters to be formed, the cluster purity value will decrease. The study implemented TFIDF word vector representation combined with various reduction percentages with LSA.

Another previous related study by [9] implemented document clustering using the BoW representation approach that aims to anticipate the BoW drawbacks, unable to identify synonym words. The study proposed an additional pre-processing stage, which is the dictionary synonym checking function. The study also implemented LSA to reduce the dimension of BoW representation. The experiment process results indicated that the addition of the synonym checking function could increase the quality cluster up to 13%.

A study by [14] aims to categorize documents using a classification approach. This study also focuses on various document representation methods such as LSA, Latent Dirichlet Allocation (LDA), and Doc2Vec. The study implemented semi-supervised learning (SSL) to improve classification performance, particularly in the labeling process. The study used five popular English corpora such as 20 newsgroup, Reuters, semEval and OhSumed.
The LSA requires the highest dimension, follow by LDA and Doc2Vec. The results of this study Doc2Vec yields the highest classification performance.

Another previous related research by [1] evaluated several document clustering and topic modeling models for the Online Social Network. The study used a large dataset corpus collected from social media such as Twitter and Reddit social. This study implemented word vector representation like TFIDF, LDA, and word embedding. This research shows word embedding combined with *k*-means clustering delivered the best performance. This study also concerns observing several hyper-parameter settings for word embedding, particularly for short text. This study found that 100 dimensions are sufficient for short text representation like *twitter*, the size of the context window is 5 and the minimum word count is 1. This study also shows K-Means is the best algorithm.

A study by [15] proposed a comparison study to analyze semantic representations of small corpora. This study compared the LSA representation versus SkipGram word2vec. The finding of this study LSA showed better performance than Skip-gram in a small size training corpus. This study also stated LSA could capture relevant words associations in the training corpus, even in a small number of low-frequency words. The study investigated the optimality of different methods to achieve reliable semantic mappings when only medium to small corpora are available for training.

A study by [21] investigates corpus specificity and corpus size in a word embedding. This study investigates the suitable number of words embedding dimensions. In this study, the observation was done to a whole corpus and sub-corpus. The sub-corpus was generated from the primary corpus that was split into a half size corpus, a third size corpus, and a quarter size corpus. The study found word2vec obtained its best performance when it is trained with the whole corpus. In this study, the contrary fact also found that the specialization (removal of out-of-domain documents) of the training corpus, accompanied by a decrease of dimensionality, can increase LSA word- representation. From a cognitive-modeling point of view, the study points out that LSA representation acquisitions may not be efficiently exploiting higher-order co-occurrences and global relations, whereas word2vec does.

A study by [22] observed the applicability of word2vec to extracting similar words from small, domain-specific data. The study found the corpus's specificity has much more influence on word2vec results than the corpus size. This study concluded the specificity of the corpus is more important than the size of the corpus. word2vec was used to extract domain-specific related terms from very small corpora. The study gets satisfactory results with small data for domain-specific words using a word embedding approach.

A study by [2] observed document clustering on a large public domain. This study used various algorithms and various word vector representation. This study obtained the Doc2Vec algorithm to get the best result.

A study by [16] proposed a method to generate effective word embedding from a limited dataset. This study expanded the small text corpus by generated multiple versions for sentences in the corpus.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Other previous studies by [17], [18], and [19] stated observation of a small corpus is important.

According to these previous studies, our study proposes an experimental study to compare TFIDF-LSA versus Doc2Vec algorithm in small corpora. Different from the most previous studies that are using available popular publicly English corpus, in our study, the observation language is Bahasa Indonesia. We generated the corpus from Bahasa Indonesia newspapers. Our study also focuses on tuning the hyper-parameter for each word vector representation to get the optimal result. We also concern in some criterion cluster performance using various cluster evaluations. We observed the quality of intra-cluster and inter-cluster using the silhouette coefficient.

## 4. METHODOLOGY

The methodology of this study described in Figure. 1

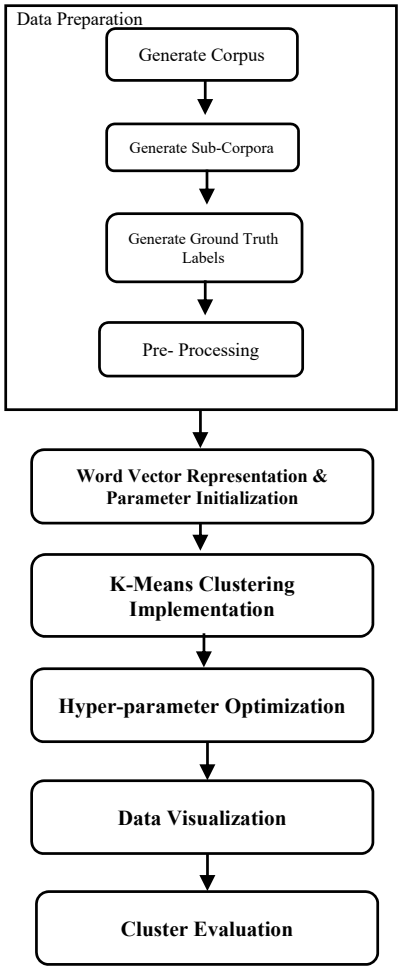


Figure 1. Methodology

### 4.1 Data Preparation

#### 4.1.1 Generate a Corpus

The first step of data preparation is to prepare a corpus. In this study, a corpus is a set of documents. We used a partial corpus of the works by [33]. The corpus was generated from Indonesian online newspapers. However, the corpus by [33] is already transformed into one big file in txt format. Meanwhile, we need the corpus that contains various topics in separate files. Therefore, we took the original form after the crawling process, which is still in JSON format. Each JSON file results from the crawling process from many articles in one topic from one newspaper. There are seven newspapers and five topics which are economics, politics, law, health, and technology. Each JSON file contains some metadata like URL, author, date, title, and articles content. In this study, we only took articles content. The number of articles in these JSON files reaches thousands of articles. The purpose of this study is to compare word vector representation in small corpora; therefore, we collected 500 articles. The corpus with 500 articles can be assumed as a small corpus compared to billions of articles in a big corpus from previous studies.

#### 4.1.2 Generate Sub-Corpora

The purpose of this study is to compare the cluster performance of two words vector representation in small corpora. Therefore, in addition to process the whole corpus with 500 documents, we also split this corpus into another 3 sub-corpora. To ease the corpus identification, we named the corpus with corpus-500, corpus-250, corpus-125, and Corpus-60. The detail of each corpus can be seen in Table 1.

*Table 1: The Corpus Information*

| Corpus | Number of articles | Total number of words | Total number of unique words |
|---|---|---|---|
| Corpus-500 | 500 | 110521 | 9629 |
| Corpus-250 | 250 | 57773 | 6615 |
| Corpus-125 | 125 | 46559 | 5770 |
| Corpus-60 | 60 | 14346 | 2639 |

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

#### 4.1.3 Determine Ground Truth Labels

In this study, determine ground truth labels is a process to annotate each document with an appropriate news category. Some of the evaluation cluster algorithms, such as Purity and ARI, need ground truth labels to calculate the cluster performance. We determined the ground truth labels for each document based on the category determined by the newspapers. To ease ground truth labels determination, we collected the exact same amount number for each category for each corpus. For example, corpus-500 contains 100 Health articles, 100 law articles, 100 economics articles, 100 technology articles, and 100 sports articles.

#### 4.1.4 Pre-Processing

Pre-processing is a step to transform raw data into an understandable and efficient format. In this stage, we implemented tokenization, data cleaning, stopwords removal, and stemming. Tokenization is a process to split documents in a corpus into word by word. Data cleaning is a process to remove unwanted symbols like HTML tags and unwanted symbols. Stopwords removal is the step to exclude unimportant words from a language. This study analyzes Bahasa Indonesia; therefore, we implemented a stopword list for Bahasa Indonesia by Tala [34]. Stemming is a process to get the root base of the words. In this study, we implemented a stemming algorithm for Bahasa Indonesia by studying [35].

### 4.2 Word Vector representation and Initialize parameter

This stage is a process to transform the cleaning corpus into an observed word vector representation, which is TFIDF-LSA and Doc2Vec. To get an optimal word vector representation, we have to choose a set of optimal hyper-parameters for a learning algorithm. There are some parameters to be tuned for each text representation. For instance, one of the parameters that affect TF-IDF quality is *n* number initialization in n-grams. An n-gram is a contiguous sequence of *n* items, and *n* is the parameter to set the maximum number of words in text sequence that be converted into a token, meanwhile, the instances of hyper-parameters of Doc2Vec model including the choice of Doc2Vec algorithm, window size, dimensions, and minimum word count and the number of epoch iteration.

### 4.3 K-Means Clustering

Each model from the previous stage hereafter implemented by the K-Means algorithm. In this study, we implemented the K-means algorithm in the Scikit-learn library [36] in python. Since in this study, we focus on word representation, not in the clustering algorithm, so we used default values provided by the Scikit-learn python package for all experiments. We used 300 iterations and n-init = 10. We set clusters number as 5; this number is consistent as news categories number in a training corpus.

### 4.4 Hyper-parameters Optimization

Despite many parameters to be tuned that can be implemented for each word vector representation and for each corpus, in this study, the hyper-parameters' initialization is tuned only in corpus-60. With consideration, corpus-60 is the smallest corpus size. Therefore, training time will be shorter. For each tuning of hyper-parameters, the best model is taken that generates the best Purity, silhouette, and ARI score. This model is then a representative of word vector representation.

#### 4.4.1 Tuning Hyper-parameters of TFIDF-LSA model

For TFIDF-LSA, we tested variations of *n* (range 1-3 ) in n-grams and implemented them with the K-means algorithm. The result can be seen in Table 2.

*Table 2: The Result of TFIDF-LSA Model*

| Model | Silhouette | Purity | ARI |
|---|---|---|---|
| TFIDF-LSA ( n = 1 / unigram) | 0.51 | 0.5 | 0.45 |
| TFIDF-LSA ( n = 2 / bigram) | 0.52 | 0.5 | 0.518 |
| TFIDF-LSA (3-gram) | 0.46 | 0.48 | 0.49 |

Based on the result in Table 2, we can conclude that the optimal hyper-parameters of TF-IDF combined with LSA is the bigram model or n = 2.

#### 4.4.2 Tuning Hyper-parameters of Doc2Vec Model

For the Doc2vec model, we tested various parameters to look for the best performance. We implemented 2 algorithms of Doc2Vec, which are PV-Dbow and PV-DM. We tested context window sizes for each algorithm ranging to values 5, 8, and 15 and dimension with values 50, 100, and 300. We set epoch number = 300 and min-count = 5. For another hyper-parameters initialization with default values provided by Gensim. The result can be seen in Table 3.

Based on Table 3, the best performance of Doc2Vec model is with implemented parameters: algorithm PV-DBOW, dimensions or vector size = 300,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

context window size = 8, epoch = 300 and window size = 5.

Further, the best model for each word representation in the hyper-parameters optimization stage will be tested using the corpus's different size. To ease the identification, we named the process with different codes. The feature representation for TFIDF-LSA with different corpus is coded as TF1 – TF4. The feature representation for Doc2Vec with different corpus is coded as DV1 – DV4.

TF1 = 2 grams, TFIDF-LSA, corpus-60
TF2 = 2 grams, TFIDF-LSA, corpus-125
TF3 = 2 grams, TFIDF-LSA, corpus-250
TF4 = 2 grams, TFIDF-LSA, corpus-500

PV1 = algorithm pv-dbow, vector size = 300, min_count = 5, epoch = 300 and window size = 8, Corpus-60

PV2 = algorithm pv-dbow, vector size = 300, min_count = 5, epoch = 300 and window size = 8, Corpus-125

PV3 = algorithm pv-dbow, vector size = 300, min_count = 5, epoch = 300 and window size = 8, Corpus-250

PV4 = algorithm pv-dbow, vector size = 300, min_count = 5, epoch = 300 and window size = 8, Corpus-500

The experiment results for various processes for TFIDF-LSA can be seen in Table 4. Meanwhile, the result for Doc2Vec can be seen in Table 5.

*Table 4: Evaluation Cluster for TF-IDF combined with LSA*

| Model | Silhouette | Purity | ARI | Time Proc (in second) | Memory Usage (in MiB) |
|---|---|---|---|---|---|
| TF1 | 0.52 | 0.50 | 0.51 | 0.6 | 243.17 |
| TF2 | 0.49 | 0.51 | 0.56 | 0.6 | 248.82 |
| TF3 | 0.43 | 0.54 | 0.61 | 0.8 | 269.14 |
| TF4 | 0.51 | 0.51 | 0.59 | 1.13 | 357 |

*Table 5: Evaluation Cluster for Doc2Vec*

| Model | Silhouette | Purity | ARI | Time proc(in a second) | Memory Usage (in MiB) |
|---|---|---|---|---|---|
| PVI | 0.60 | 0.37 | 0.012 | 0.4 | 238.09 |
| PV2 | 0.63 | 0.39 | 0.090 | 0.46 | 239.68 |
| PV3 | 0.57 | 0.32 | 0.014 | 0.46 | 244.54 |
| PV4 | 0.58 | 0.29 | 0.019 | 0.5 | 249 |

*Table 3. Hyper-parameters of Doc2Vec Model*

| | | PV-DBoW | | | | PV-DM | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | WS = 5 | WS = 8 | WS =15 | | WS = 5 | WS = 8 | WS =15 | |
| Dimension = 50 | Purity | 0.37 | 0.35 | 0.37 | | 0.37 | 0.37 | 0.32 | |
| | Silhoutte | 0.56 | 0.56 | 0.56 | | 0.59 | 0.52 | 0.55 | |
| | ARI | 0.024 | 0.025 | 0.020 | | 0.010 | 0.04 | -0.01 | |
| Dimension = 100 | Purity | 0.35 | 0.35 | 0.35 | | 0.37 | 0.33 | 0.30 | |
| | Silhouette | 0.56 | 0.54 | 0.55 | | 0.56 | 0.55 | 0.56 | |
| | ARI | 0.025 | 0.02 | 0.02 | | 0.03 | 0.012 | -0.02 | |
| Dimension = 300 | Purity | 0.33 | **0.37** | 0.33 | | 0.33 | 0.32 | 0.32 | |
| | Silhouette | 0.59 | **0.60** | 0.61 | | 0.61 | 0.52 | 0.56 | |
| | ARI | -0.0004 | **0.012** | 0.003 | | 0.004 | 0.020 | -0.006 | |

### 4.5 Data Visualization

Data visualization is an essential parameter to describe the clustering pattern. To adjust with the limitation of human eyes, we can only see 2 dimensions; therefore, we have to reduce the word vector representation into x and y layer. The original TFIDF representation will generate a sparse matrix that has dimensions as much as total words in the corpus. However, in this study, we implemented LSA to get meaning representation, and this implementation is also impacted by the reduction dimension of the TFIDF-LSA model. For the Doc2Vec model, the word vector representation dimension is 300. In this study, we implemented the Principal Component Analysis (PCA) to reduce the model's dimension. PCA is also can emphasize

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

variation and bring out a strong pattern in the corpus. Cluster result visualization for each model can be seen in Figure 2. There are many aspects to determine the quality of the cluster. Based on table 4 and Table 5, for cluster performance, each model has 3 kinds of cluster evaluation scores, which are silhouette, Purity, and ARI.

### 4.6 Cluster Evaluation

#### 4.6.1 Performance Cluster Evaluation

**Intrinsic Evaluation**

The silhouette coefficient score is one of an intrinsic measurement to indicate how well an algorithm optimized a particular representation. The TF-IDF model for every size of the corpus gained about 0.5 silhouette score. Meanwhile, Doc2Vec got about 0.6

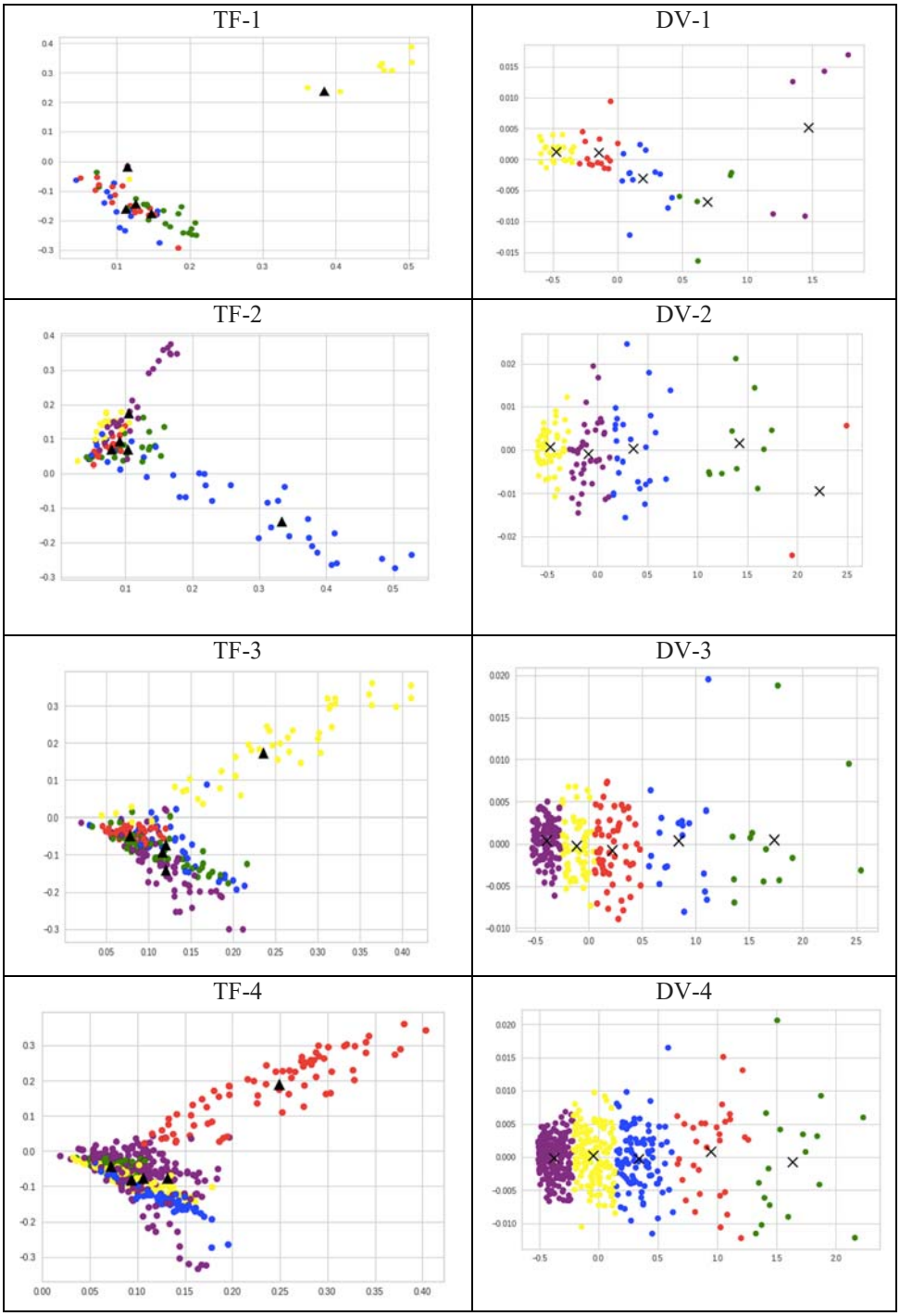


*Figure 2. Data Visualization for Each Model of Word Vector Representation*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

silhouette score. Even Though we cannot compare intrinsic evaluation between different representations, we can see both models, TFIDF and Doc2Vec model, only gained half silhouette score for the number of clusters = 5; meanwhile, the maximum purity score is 1. Still using $k$ = 5, we calculated the silhouette score for each model. The result tends to obtain the same value for various corpus sizes. This result indicates the clustering algorithm not optimal to cluster the object into 5 clusters. This intrinsic measurement calculates the score based only on word vector representation and does not need other external information; therefore, the experiments can be done to a various number of $k$. The result of the Silhouette score for various $k$ using the Doc2Vec model can be seen in Figure 3 and for the TFIDF-LSA model in Figure 4. Based on this experiment,

we found the highest Silhouette score was obtained for k 2, 3, and 4. This fact made us re-examine the documents in the corpus. We found some categories are so similar to each other. For instance, in our corpus, the Economics category is too similar to the Politics category and Laws category. These categories have many same words so that the boundaries between clusters are not so obvious. Because K-means clustering is a hard cluster, if we initialize the $k$ number bigger than the optimal $k$ obtained by intrinsic measurement, then the quality cluster's intrinsic score will not be optimal. The findings show the quality of clusters not only influenced by the corpus size but also by the quality of the corpus. It needs deeper observation for further research that focuses on the quality of data corpus, such as the comparison of small corpora in domain-specific and general corpora. We assumes the performance of cluster can be improved if various size of corpus and various small corpora in domain-specific is implemented in this study.

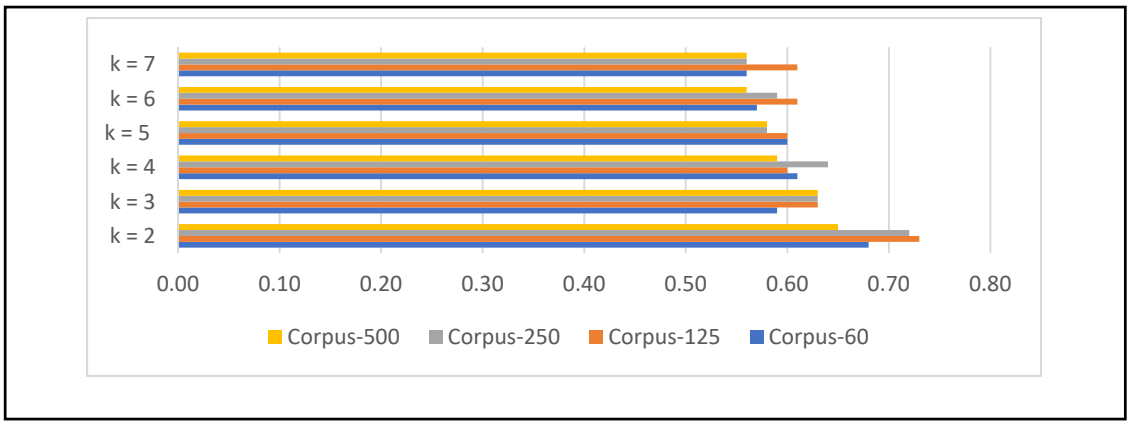


*Figure 3. Silhouette Score for Doc2Vec Representation*

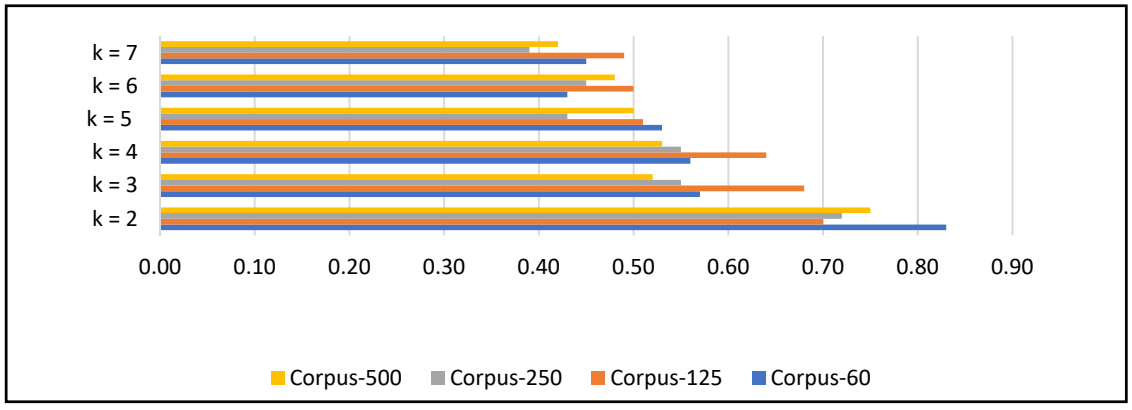


*Figure 4. Silhouette Score for TFIDF-LSA Representation*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

**Extrinsic Evaluation**

In this study, the extrinsic measurement of cluster performance is Purity and ARI. An extrinsic measurement need ground truth labels as references to indicate how well the clusters For both extrinsic evaluations, TFIDF-LSA gets a better purity and ARI score than the Doc2Vec model. ARI measurement focuses on the capability of the K-Means algorithm to separate the elements belonging to different classes. Based on the ARI score, the TFIDF-LSA representation is better in separating elements that not belong to the same class as the Doc2Vec model. Based on Table 4 and Table 5, the comparison of extrinsic cluster performance evaluation is shown in Figure 5. BesideThe performance of DocVec dep

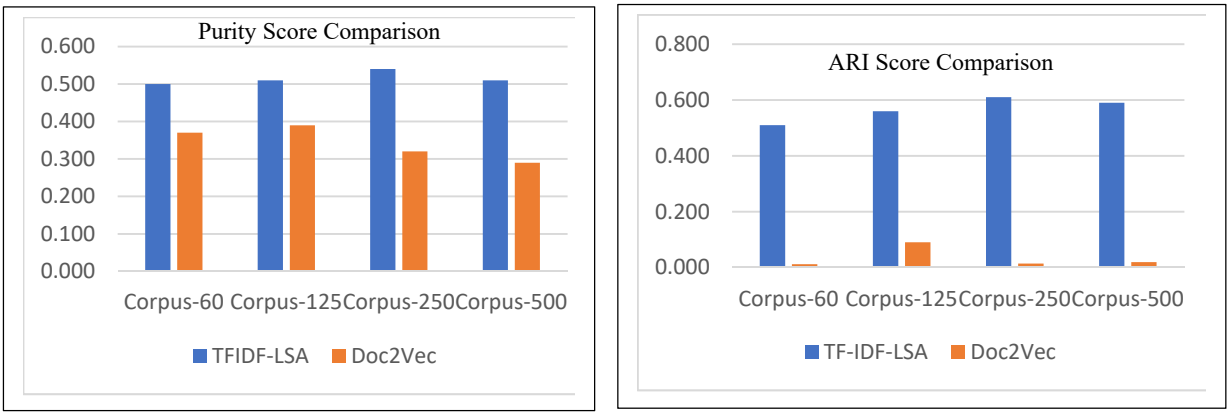


*Figure 5. Cluster Performance Comparison*

**Time and Memory Consumption Evaluation**

In addition to cluster performance, the other consideration is the efficiency of cluster processing. Based on the experiment in this study, the time and memory consumption measurements are showed in Figure 6. Doc2Vec needs less time and needs less memory in the clustering process to compare to the TFIDF-LSA model. It means the Doc2Vec model is better in time and memory usage efficiency than the TFIDF-LSA model. In the TFIDF-LSA model, time and memory usage are increasing along with corpus size increases. Meanwhile, in the Doc2Vec model, time and memory usage tend to be stable even though the corpus size increases.

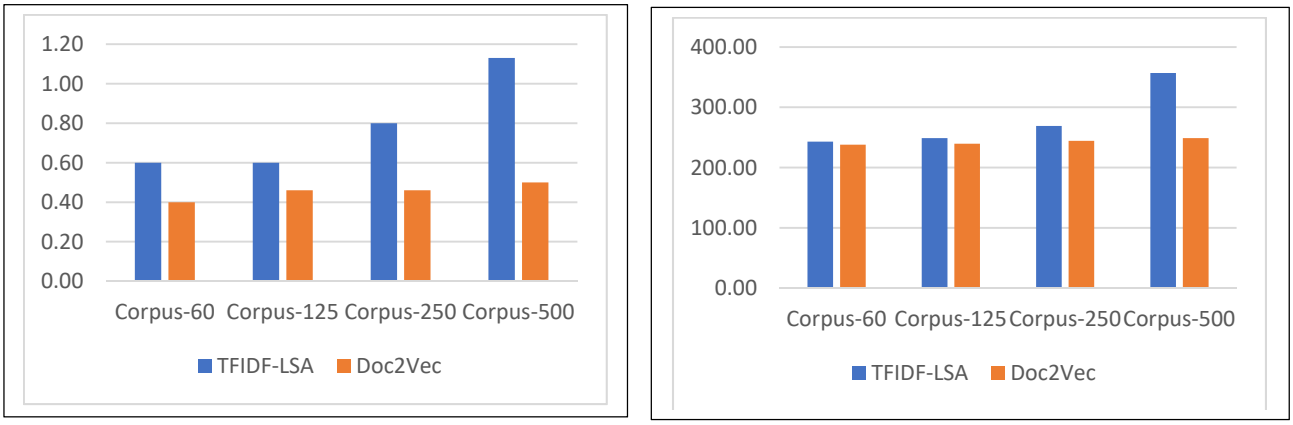


*Figure 6. Time and Memory Consumption Comparison*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

## 5. CONCLUSION

The study's purpose is to observe the suitable word vector representation in text clustering for small corpora. In this study, the comparison based on cluster performance, time, and memory consumption. For the cluster performance, which is based on extrinsic measurement, the TDIDF-LSA gets better performance than the Doc2Vec model. It means the TFIDF-LSA representation is better in separating elements that not belong to the same class as the Doc2Vec model for all the observed corpora. On the contrary, Doc2Vec is better than TFIDF-LSA in time and memory consumption. The usage of time and memory in the TFIDF-LSA model is increasing, along with corpus size increases. Meanwhile, in the Doc2Vec model, time and memory usage tend to be stable even though the corpus size increases. As further work, the same experiments should be done in a bigger corpus size but is still classified as a small corpus to find out in what points the Doc2Vecs tend to get better performance than TFIDF-LSA. Moreover, it needs deeper observation for further research that focuses on the quality of data corpus.

## ACKNOWLEDGMENT

The authors gratefully acknowledge that the present research is supported by Lembaga Penelitian Universitas Sumatera Utara. The support is under the research grant TALENTA USU of the Year 2020.

## REFERENCES:

[1] S. A. Curiskis, B. Drake, T. R. Osborn, and P. J. Kennedy, "An evaluation of document clustering and topic modelling in two online social networks: Twitter and Reddit," *Inf. Process. Manag.*, vol. 57, no. 2, p. 102034, Mar. 2020.

[2] F. ois Role, S. Morbieu, and M. Nadif, "Unsupervised evaluation of text co-clustering algorithms using neural word embeddings," in *International Conference on Information and Knowledge Management, Proceedings*, 2018, pp. 1827–1830.

[3] E. Rendón, I. Abundez, A. Arizmendi, and E. M. Quiroz, "Internal versus External cluster validation indexes," *Int. J. Comput. Commun.*, vol. 5, no. 1, pp. 27--34, 2011.

[4] S. ; Khaled; Ranka and V. Singh, "An efficient k-means clustering algorithm," 1997.

[5] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[6] T. Kohonen, "The Self-Organizing Map," *Proc. IEEE*, 1990.

[7] Joachims Thorsten, "A Probabilistic Analysis of Rocchio Algorithm with TFIDF for Text Categorization," 1996.

[8] A. Amalia, M. S. Lydia, S. D. Fadilla, and M. Huda, "Perbandingan Metode Klaster dan Preprocessing Untuk Dokumen Berbahasa Indonesia," *J. Rekayasa Elektr.*, vol. 14, no. 1, pp. 35–42, Apr. 2018.

[9] A. Amalia, M. S. Lydia, S. D. Fadilla, M. Huda, and D. Gunawan, "Document Clustering Optimization with Synonym Dictionary Check Function Case Study : Documents in Bahasa Indonesia," in *2017 International Conference on Electrical Engineering and Informatics (ICELTICs)*, 2017, pp. 286–291.

[10] Y. Bengio *et al.*, "A Neural Probabilistic Language Model," 2003.

[11] Q. Li, S. Shah, X. Liu, and A. Nourbakhsh, "Data Sets: Word Embeddings Learned from Tweets and General Data," 2017.

[12] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural Language Processing (Almost) from Scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, 2011.

[13] T. Mikolov, W.-T. Yih, and G. Zweig, "Linguistic Regularities in Continuous Space Word Representations," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013, pp. 746–751.

[14] D. Kim, D. Seo, S. Cho, and P. Kang, "Multi-co-training for document classification using various document representations: TF–IDF, LDA, and Doc2Vec," *Inf. Sci. (Ny).*, vol. 477, pp. 15–29, Mar. 2019.

[15] E. Altszyler, S. Ribeiro, M. Sigman, and D. Fernández Slezak, "Comparative study of LSA vs Word2vec embeddings in small corpora: a case study in dreams database," *Conscious. Cogn.*, vol. 56, pp. 178–187, Oct. 2017.

[16] A. Silva and C. Amarathunga, "On Learning Word Embeddings From Linguistically Augmented Text Corpora," in *Proceedings*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

*of the 13th International Conference on Computational Semantics*, 2019, pp. 52–58.

[17] C. Jiang, C. J. Hsieh, H. F. Yu, and K. W. Chang, "Learning word embeddings for low-resource languages by PU learning," in *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 2018, vol. 1, pp. 1024–1034.

[18] Y. Gu, G. Leroy, S. Pettygrove, M. K. Galindo, and M. Kurzius-Spencer, "Optimizing Corpus Creation for Training Word Embedding in Low Resource Domains: A Case Study in Autism Spectrum Disorder (ASD)," *AMIA ... Annu. Symp. proceedings. AMIA Symp.*, vol. 2018, pp. 508–517, 2018.

[19] P. Tao, J. Minghua, and H. Ming, "A dynamic clustering algorithm based on small data set," in *Proceedings of the 2009 6th International Conference on Computer Graphics, Imaging and Visualization: New Advances and Trends, CGIV2009*, 2009, pp. 410–413.

[20] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[21] E. Altszyler, M. Sigman, and D. Fernández Slezak, "Corpus Specificity in LSA and Word2vec: The Role of Out-of-Domain Documents," pp. 1–10, Jun. 2019.

[22] E. Dusserre, "Bigger does not mean better! We prefer specificity," in *12th international conference on computer semantics (IWCS)*, 2017, pp. 1–6.

[23] T. K. Landauer, P. W. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse Process.*, vol. 25, no. 2–3, pp. 259–284, Jan. 1998.

[24] N. E. Evangelopoulos, "Latent semantic analysis," *Wiley Interdiscip. Rev. Cogn. Sci.*, vol. 4, no. 6, pp. 683–692, 2013.

[25] T. Mikolov, I. Sutskever, K. Chen, … G. C.-A. in neural, and U. 2013, "Distributed Representations ofWords and Phrases and their Compositionality," *CrossRef List. Deleted DOIs*, vol. 1, 2013.

[26] M. Sahlgren, "The distributional hypothesis *," *Ital. J. Disabil. Stud.* , vol. 20, pp. 33–53, 2008.

[27] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," pp. 1–12, 2013.

[28] Google, "Google Code Archive - Long-term storage for Google Code Project Hosting.," *Code.Google.com*. pp. 1–6, 2013.

[29] J. Manotumruksa, C. Macdonald, and I. Ounis, "A deep recurrent collaborative filtering framework for venue recommendation," in *International Conference on Information and Knowledge Management, Proceedings*, 2017.

[30] Q. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," in *International Conference on Machine Learning*, 2014.

[31] "2.3. Clustering — scikit-learn 0.22.2 documentation," 2020. [Online]. Available: https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation. [Accessed: 10-May-2020].

[32] C. M. De Vries, S. Geva, and A. Trotman, "Document Clustering Evaluation: Divergence from a Random Baseline," 2012.

[33] A. Amalia, O. Salim Sitompul, E. Budhiarti Nababan, M. Silvi Lydia, and N. Rahmatunnisa, "BAHASA INDONESIA TEXT CORPUS GENERATION USING WEB CORPORA APPROACHES," *J. Theor. Appl. Inf. Technol.*, vol. 31, p. 24, 2019.

[34] F. Z. Tala, "A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia," *M.Sc. Thesis, Append. D*, vol. pp, pp. 39–46, 2003.

[35] M. Adriani, J. Asian, B. Nazief, and H. E. Williams, "Stemming Indonesian : A Confi x-Stripping Approach," *ACM Trans. Asian Lang. Inf. Process.*, vol. 6, no. 4, pp. 1–33, 2007.

[36] J. Montiel, J. Read, A. Bifet, and B. Kegl, "Scikit-Multiflow: A Multi-output Streaming Framework," *J. Mach. Learn. Res.*, vol. 19, pp. 1–5, 2018.