Fine-Grained Mobile Application Clustering Model Using Retrofitted Document Embedding

Yeo-Chan Yoon, Junwoo Lee, So-Young Park, and Changki Lee

In this paper, we propose a fine-grained mobile application clustering model using retrofitted document embedding. To automatically determine the clusters and their numbers with no predefined categories, the proposed model initializes the clusters based on title keywords and then merges similar clusters. For improved clustering performance, the proposed model distinguishes between an accurate clustering step with titles and an expansive clustering step with descriptions. During the accurate clustering step, an automatically tagged set is constructed as a result. This set is utilized to learn a high-performance document vector. During the expansive clustering step, more applications are then classified using this document vector. Experimental results showed that the purity of the proposed model increased by 0.19, and the entropy decreased by 1.18, compared with the K-means algorithm. In addition, the mean average precision improved by more than 0.09 in a comparison with a support vector machine classifier.

Keywords: Document embedding, Text clustering, Deep learning.

Manuscript received Dec. 21, 2016; revised Apr. 19, 2017; accepted May 8, 2017. This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (R0118-16-1005, Digital Contents In-House R&D).

Yeo-Chan Yoon (ycyoon@etri.re.kr) and Junwoo Lee (leejw@etri.re.kr) are with the SW & Content Research Laboratory, ETRI, Daejeon, Rep. of Korea.

So-Young Park (corresponding author, ssoya@smu.ac.kr) is with the Department of Game Design and Development, Sangmyung University, Seoul, Rep. of Korea.

Changki Lee (leeck@kangwon.ac.kr) is with the Department of Computer Science, Kangwon National University, Chuncheon, Rep. of Korea.

I. Introduction

Smartphones have achieved considerable success with an average penetration rate of 70% in 50 countries around the world. In addition, more than two million mobile applications have been registered in both the Google Play Store and the Apple App Store [1]. In these stores, mobile apps are classified in a coarse-grain fashion into four to 50 different categories. However, the number of categories is too small to classify the millions of applications available, and the classification criteria are often ambiguous. For example, both scanning and job-search applications are generally classified into the business category, whereas real-estate applications providing similar functions are classified into lifestyle, finance, or business categories according to the developer's determination.

Certain approaches have recently been proposed to automatically classify mobile applications by analyzing their user logs and metadata. Some of them classify mobile applications into dozens of predefined categories with a good level of performance [2]–[6]. For fine-grained classification, the authors in [7] classify mobile applications into 250 clusters. However, for such approaches, dozens of predefined categories cannot be sufficient. Even for fine-grained classification, 250 clusters cannot be deemed a reasonable number for preparing a personalized, referral, or analysis service.

In this paper, we propose a fine-grained mobile application clustering model for automatically determining clusters and their numbers without any predefined categories. Considering that mobile application titles precisely identify their characteristics—for example, job search applications are identified as "job search - snag ***" and "** free jobs"—the proposed model initializes clusters based on the title keywords. The initialized clusters are used as a training set to classify applications without any title keywords.

https://doi.org/10.4218/etrij.17.0116.0936

pISSN: 1225-6463, eISSN: 2233-7326

To classify applications with an automatically constructed training set, we propose a method for learning document vector representations. Unlike conventional methods for predicting the words in a document from the input document, the proposed model learns how to predict the documents within the same cluster with distance constraints between the input document and the output documents on the object function. Because synonyms with different keywords have mutually separated clusters, it then merges certain similar clusters. For improved performance, the proposed model uses word embedding and retrofitted document embedding.

The major contributions of this paper are summarized as follows. First, we propose a fine-grained and semi-supervised clustering method for a very large mobile application set. Second, we propose a method for learning a document vector using an automatically constructed training set. Finally, we exploit both word embedding and document embedding to merge similar clusters.

II. Related Works

1. Fine-Grained Document Clustering

Word and phrase-based clustering approaches utilize salient words or phrases as keywords for document clustering [8]–[15]. They have the following advantages. The label of each cluster can be easily obtained from the keywords of the clustering results. In addition, these approaches do not require predefined categories or their numbers. Moreover, they can choose either the fine-grained clustering results or the coarse-grained clustering results by adjusting the keyword selection method applied. Product-feature-based clustering approaches [12]–[15] have recently focused on sentences describing the characteristics of a product, such as its functions. The product features are recognized from such sentences and then clustered according to what the sentence describes. The performance depends on how the feature keywords are selected from the sentences. An association rule [16] is widely used for such tasks, and probability-based methods are used to remove inappropriate words by distinguishing between words that tend to frequently occur in certain product fields [12]–[14].

To obtain useful keywords from millions of mobile applications, the proposed model extracts keywords from the titles, rather than long descriptions. The proposed model considers there to be much less noise in a title than in a description because creators tend to carefully name their mobile applications.

2. Text Representation through Deep Learning

Word-embedding approaches can change from a high dimensional discrete vector to a low-dimensional continuous vector. CBOW and Skipgram have been proposed as neural-network based word-embedding models [17]. CBOW learns the weight matrix to predict the target word from the input context words, whereas Skipgram learns the weight matrix to predict the context words from the input word. The learned weight matrix is used to encode words into a low-dimensional word vector.

Word-embedding approaches have been applied to various problems, such as text classification, text clustering, and textual similarity tasks. Text similarity can be measured through document or word embedding [18], [19]. Recently, word-embedding approaches have been mainly used for generating the features of the classifier, such as the support vector machine (SVM), the convolutional neural network (CNN), or the recurrent neural network (RNN) [20]–[23]. To classify texts, the authors of [20] and [21] use word-embedding as an input layer of a CNN. The average word vector is used as a feature [22], [23]. In these approaches, the classification performance highly depends on the size of the tagged corpus. However, in a fine-grained clustering problem, it is difficult to construct an adequately large tagged corpus per cluster because too many clusters exist. Although the relationships among many clusters can follow a hierarchical system for certain tasks, such as mobile application clustering, these approaches are generally suitable for a flat classification system, rather than for a hierarchical system.

On the other hand, some embedding approaches have been used for sentences, documents, or semantic units [24]–[29]. A document-embedding method called *doc2vec*, which is similar to CBOW and Skipgram, was proposed [24]. Unlike CBOW and Skipgram, however, the document-embedding method represents a document as a vector. For a document similarity task, a document-embedding-based approach performs better than an *n*-gram-based approach or a word-embedding-based approach [18]. A topic tag can be used to learn the document vector [27]. However, for untagged data, a document vector cannot be learned in the same way as the tagged data.

Some word-embedding approaches have used knowledge resources for improving the performance. Similar to the pairwise ranking approach, the authors in [30] consider the constraints in which the distance between two antonyms is longer than the distance between

https://doi.org/10.4218/etrij.17.0116.0936

two random words, whereas the distance between two synonyms is closer to the distance between two random words. Similarly, the authors in [31] consider the constraint in which the distance between two antonyms is longer than the distance between two synonyms according to WordNet. In addition, the authors of [28] consider the constraint in which a word vector can be generated by combining the sense vectors. To fine-tune the learned weight matrix, the authors in [32] consider the distance between two related words as being closer to the distance between random words according to language resources, such as WordNet and the Paraphrase Database.

For improved performance, the proposed model adopts semi-supervised learning. The model learns the document embedding based on a conventional *doc2vec* algorithm and then retrofits it using an automatically tagged training corpus. As a constraint, we use the assumption that the distance between two documents in the same cluster should be close.

III. Fine-Grained Mobile Application Clustering Model

The proposed model consists of the following steps, as shown in Fig. 1. First, the keyword selector extracts keywords as the initial cluster labels from the mobile application titles. To cluster similar mobile applications, the cluster initializer generates the initial clusters based on the title keywords. By using document embedding, the cluster expander allows the initial clusters to include more mobile applications without any title keywords. Finally, the cluster merger combines similar clusters by using word and document embedding.

1. Initializing Clusters with Title Keywords

Considering that a mobile application title precisely identifies its characteristics, the proposed model initializes clusters based on the keywords in the title. The proposed model utilizes an association rule [16] to extract keywords from the titles. For example, the keywords “nail art,” “nail design,” and “home design” can be extracted from the titles.

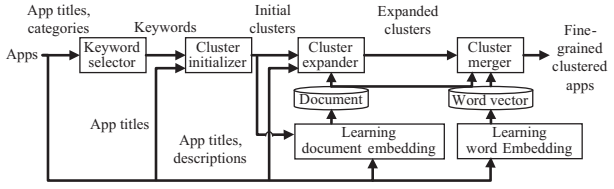

Fig. 1. Proposed mobile application clustering model.

On the other hand, many mobile application titles can include some stop words, such as “iPhone” and “Android.” Although these stop words are not useful for clustering, they can be extracted as keywords because they are frequently included in many mobile application titles. To filter these stop words from the keywords, the proposed model considers that the mobile application stores classify the mobile applications into 40 to 50 categories according to their characteristics.

Assuming that more frequent words in only certain categories can indicate the important keywords to cluster for the mobile applications, the proposed model gives a high weight to these important keywords by calculating the relative category frequency (RCF) as follows:

$$\mathrm{RCF}(word,c)=\frac{\sum_{d\in c} tf(word,d)}{|c|}\times\frac{cf(word,c)/|c|}{\sum_{c'\in C} cf(word,c')/|c'|} \quad (1)$$

where *word* is a candidate keyword, c is a mobile application category, d is a document in category c, $tf(word,d)$ is the term frequency of *word* in document d, $|c|$ is the number of total documents in the category, $cf(word,c)$ is the term frequency of *word* in category c, and C is the set of categories.

The extracted keywords are used to classify the applications, including the keywords in the title, into the initial clusters. For example, the titles “nail art tutorial,” “nail art salon,” and “candy nail art” are classified as the “nail art” cluster. In order to improve the document embedding, the proposed model utilizes these initial clustering results as the cluster tagged training corpus, as described in Section IV.

2. Expanding Clusters with Document Embedding

To overcome the limitation in which the initial cluster cannot classify many mobile applications with no title keyword, the proposed model expands the initial clusters using document embedding. First, it generates a document vector from both the title and the description of each mobile application in the initial clusters. It then calculates the average of the document vectors per initial cluster. It calculates the cosine similarity between the new cluster centroids and the document vector of every mobile application with no title keyword. The cluster can additionally accept new mobile applications with a higher cosine similarity than threshold ε. Although the titles “French manicure,” “my beautiful nails,” and “nail salon” do not include the keywords “nail art,” they are classified into the “nail art” cluster by the document embedding of the application description.

3. Merging Clusters with Word and Document Embedding

To merge similar clusters, such as “nail art” and “nail design,” the proposed method exploits label similarity, cluster similarity, and cluster overlapping, as described in Fig. 2. It thus does not require the resource construction cost for a synonym dictionary because it utilizes the values of label similarity, cluster centroid similarity, and the overlapped number. When the linear combination of these three values exceeds the threshold, two clusters are merged, as described in Fig. 2. First, the label similarity is the cosine similarity between the word vectors of the keywords representing two clusters, and each word vector is calculated using word embedding [17]. Whenever two clusters are merged, the word vector of the merged cluster estimates the average of the previous two word vectors. Second, the cluster similarity is the cosine similarity between two average document vectors, indicating the average of all document vectors in each cluster. Third, the overlapped number is the number of mobile applications overlapped between two clusters.

4. Time Complexity of the Algorithm

The time complexity of the keyword selection is $O(n)$ because the complexity of the association rule [16] is $O(n)$, where n is the number of documents. The complexity of initializing and expanding the clusters is $O(n)$ and $O(c \cdot n)$, respectively, where c is the number of initialized clusters. The time complexity of merging clusters takes $O(c^2)$ because it computes the similarity for each pair of clusters. Because c is much smaller than n, it is feasible to apply the algorithm to a large data.

```
1   while merged = true
2     merged = false
3     for clusterA in all_clusters
4       for clusterB in all_clusters
5         label_similarity = cos (clusterA.label, clusterB.label)
6         cluster_similarity = cos (clusterA.centroid, clusterB.centroid)
7         overlap = # of apps overlapped between clusterA and clusterB
8         if αlabel_similarity + βcluster_similarity
            + (1 – α – β)cluster_overlap > threshold
9           clusterC = merge(clusterA, clusterB)
10          Insert clusterC to all_clusters
11          delete clusterA, clusterB from all_clusters
12          merged = True
13          go to line 1
```

Fig. 2. Cluster merge algorithm.

IV. Learning Document Embedding with Training Set

The document embedding method represents documents as vectors. Most of the previous methods learn the document vector in an unsupervised manner, as described below in Section IV-1, whereas the proposed method learns the vector in a semi-supervised manner, as described in Section IV-2.

1. Skipgram and *doc2Vec*

As a word embedding method, shown in *word2vec* in Fig. 3, the authors of [17] proposed Skipgram, a neural network predicting the context words w_j, w_{j+1}, and w_{j+2}, from the input word w_i based on the respective input, projection, and output layers. In (3), V_{w_i} is a word vector of input word w_i based on the weight matrix, whereas V'_{w_j}, is a word vector of the output word w_j.

$$\frac{1}{T}\sum_{t=1}^{T}\sum\nolimits_{j\in cw(i)}\log \mathrm{P}(w_j|w_i), \tag{2}$$

$$\log \mathrm{P}(w_j|w_i) = \frac{\exp {V'_{w_j}}^{T} V_{w_i}}{\sum_{j'\in \mathrm{Vocab}} \exp {V'_{w_{j'}}}^{T} V_{w_i}}. \tag{3}$$

Given the training corpus T, Skipgram maximizes the log probability in (2) and learns the weight matrix. Because (3) requires several computations to calculate the denominator, the authors of [33] utilize a negative sampling method to decrease the number of computations. Finally, each word is represented as a vector based on the weight matrix.

Considering the characteristics of a multi-word text, such as a document, sentence, or paragraph, the authors of [24] proposed a document embedding method, called *doc2vec*, by modifying the word-embedding methods of Skipgram (PV-DBOW) and CBOW (PV-DM).

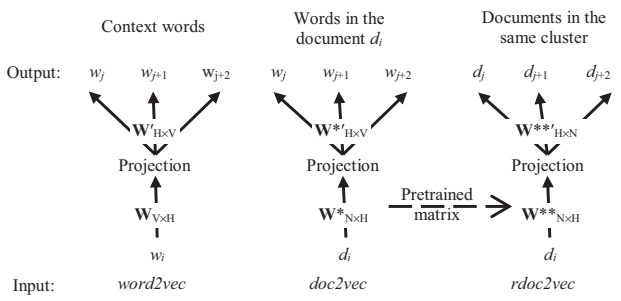

Fig. 3. Proposed retrofitted document embedding architecture.

https://doi.org/10.4218/etrij.17.0116.0936

For a forum question duplication task and semantic textual similarity task, document-embedding-based approaches perform better than word-embedding-based approaches, while a PV-DBOW-based approach performs better than PV-DM [18]. Therefore, the proposed model utilizes the document-embedding method PV-DBOW to obtain a pre-trained document vector. Given the input documentd_i, it predicts the context words w_j, w_{j+1}, and w_{j+2} in the document, as described in *doc2vec* of Fig. 3.

2. Retrofitting Document Embedding with Tagged Corpus

The proposed model uses the initial clustering results as a cluster tagged training corpus to retrofit the pre-trained document vector. Specifically, the proposed model utilizes the pre-trained weight matrix, which is learned using the *doc2vec* method, as the initial weight matrix. It then updates the weight matrix using the documents in the same cluster as the output layer. As described in *doc2vec* of Fig. 3, the conventional document embedding method predicts the words in the document from the input document. On the other hand, the proposed model predicts the documents d_j in the same cluster from input document d_i, as described in *rdoc2vec* of Fig. 3.

The weight matrix is updated according to the constraint in which the distance between two documents in the same cluster should be close within the embedding space. Inequality (4) represents the constraint in which threshold δ indicates the distance between two random documents. In inequality (4), the document vectors of two documents D_i and D_j are represented as $\mathrm{V}_{\mathrm{D}_i}$ and $\mathrm{V}_{\mathrm{D}_j}$.

$$\mathrm{dist}(\mathrm{V}_{\mathrm{D}_i}, \mathrm{V}_{\mathrm{D}_j}) < \delta \text{ if } \mathrm{D}_i \text{ and } \mathrm{D}_j \text{ are in the same cluster} \quad (4)$$

The proposed model maximizes the log probability (5) with a constraint and learns the weight matrixes. In log probability (5), D is the document set, D_i is the input document, D_j is the output document, and $\mathrm{sc}(\mathrm{D}_i)$ is the index set for the documents within the same cluster.

$$\sum_{i\in \mathrm{D}} \sum_{j\in \mathrm{sc}(\mathrm{D}_i)} \log \mathrm{P}(\mathrm{D}_j|\mathrm{D}_i) \quad (5)$$

$$\text{Subject to } \mathrm{distance}(\mathrm{D}_j, \mathrm{D}_i) < \delta,$$

$$\log \mathrm{P}(\mathrm{D}_j|\mathrm{D}_i) = \frac{\exp^{\mathrm{V}'_{\mathrm{D}_j}{}^{T}\mathrm{V}_{\mathrm{D}_i}}}{\sum_{j'\in \mathrm{D}} \exp^{\mathrm{V}'_{\mathrm{D}_{j'}}{}^{T}\mathrm{V}_{\mathrm{D}_i}}}. \quad (6)$$

The proposed model learns the document vector according to the semi-supervised learning method. The pre-trained document embedding is fine-tuned using the cluster-tagged training corpus, where the corpus is automatically generated from the initial clustered results. To solve the constraint optimization problem, the proposed model utilizes the constraint as a penalty term of the object function [31]. In (7), $\eta$is a control parameter used to balance the contribution of the penalty term.

$$O' = O - \eta \mathrm{Dist}(\mathrm{D}_i, \mathrm{D}_j). \quad (7)$$

The distance between two documents is estimated based on the cosine distance. A closer distance between two documents D_i and D_j in the same cluster incurs a smaller penalty value, whereas a longer distance incurs a greater penalty value.

$$\mathrm{Dist}(\mathrm{D}_j, \mathrm{D}_i) = 1 - \cos(\mathrm{V}_{\mathrm{D}_i}, \mathrm{V}_{\mathrm{D}_j}). \quad (8)$$

The proposed model conducts normalization using a negative sampling method. The negative sampling method randomly extracts certain documents from the other clusters without document D_i. When the weight matrix is updated, the negative samples are used in the opposite way as a positive sample; if the document vector is close to the document vector in the negative samples, it incurs a greater penalty value.

Similar to a conventional document embedding approach, the weight matrix is learned according to the stochastic gradient descent algorithm. For the back propagation algorithm, the penalty term is derived as follows:

$$\frac{\partial \mathrm{Dist}(\mathrm{D}_i, \mathrm{D}_j)}{\partial \mathrm{V}_{\mathrm{D}_i}} = \frac{\partial\left(1 - \cos\left(\mathrm{V}_{\mathrm{D}_i}, \mathrm{V}_{\mathrm{D}_j}\right)\right)}{\partial \mathrm{V}_{\mathrm{D}_i}}. \quad (9)$$

Cosine similarity can be partially differentiated as follows:

$$\cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}\cdot\mathbf{b}}{\sqrt{\mathbf{a}^2\mathbf{b}^2}}, \quad (10)$$

$$\begin{aligned}\frac{\partial \mathrm{cossim}(\mathbf{a}, \mathbf{b})}{\partial_{\mathrm{a}_1}} &= \frac{\partial}{\partial_{\mathrm{a}_1}} \frac{\mathrm{a}_1\cdot\mathrm{b}_1 + \cdots + \mathrm{a}_n\cdot\mathrm{b}_n}{|\mathbf{a}|\cdot|\mathbf{b}|} \\ &= \frac{\partial}{\partial \mathrm{a}_1} \mathrm{a}_1\cdot\mathrm{b}_1\cdot(\mathrm{a}_1^2 + \mathrm{a}_2^2 + \cdots + \mathrm{a}_n^2)^{-\frac{1}{2}}\cdot|\mathbf{b}|^{-1} \\ &= \frac{\mathrm{b}_1}{|\mathbf{a}|\cdot|\mathbf{b}|} - \frac{\mathrm{a}_1\cdot\mathrm{b}_1}{|\mathbf{a}|\cdot|\mathbf{b}|}\cdot\frac{\mathrm{a}_1}{|\mathbf{a}|^2}, \end{aligned} \quad (11)$$

$$\therefore \frac{\partial \cos(\mathbf{a}, \mathbf{b})}{\partial \mathbf{a}} = \frac{\mathbf{b}}{|\mathbf{a}|\cdot|\mathbf{b}|} - \cos(\mathbf{a}, \mathbf{b})\cdot\frac{\mathbf{a}}{|\mathbf{a}|^2}. \quad (12)$$

In (11), a_i and b_i are the i-th values of vectors **a** and **b**, respectively.

V. Evaluation

1. Experimental Dataset

For the evaluation dataset, 2.1 million mobile alications were crawled from the Google Play Store [34], and 1,000 mobile applications were selected for each of the following five categories: lifestyle (Life), education (Edu), travel and local (Travel), tools (Tool), and entertainment (Ent). Then, three annotators manually classified 5,000 mobile applications with a fine-grained hierarchical classification structure. When the annotators obtained different results for the same mobile application, the different results were changed into a single result according to the mutual agreement. The hierarchical classification structure enables both a lower-level class (such as "make up") and a top-level class (such as "beauty") for the same mobile application.

The final number of mobile applications classified per category was less than 1,000 because some mobile applications were excluded from the test set when the annotators failed to agree. Table 1 shows the details of the test set and the crawled set. To learn the document embedding, we exploited the crawled set as the training corpus, which was automatically clustered using the cluster initializer of the proposed model, as described in Fig. 1.

2. Experimental Setup

The proposed model shown in Fig. 1 is implemented as follows. First, the keyword selector extracts certain keywords based on the association rule [16] and RCF, as described in (1). Then, the cluster initializer classifies some mobile applications based on the keywords included in each mobile application title. Additionally, the cluster expander classifies other mobile applications with no keywords by calculating the cosine similarity between each cluster centroid and every mobile application. The classification is based on the document vector and is learned using the proposed retrofitted document embedding method shown in Fig. 3.

Finally, the cluster merger combines similar clusters according to the cluster merge algorithm shown in Fig. 2 based on the document vector and word vector. Moreover, it is learned using the word embedding [17]. The irace package [35] is used to tune the hyper-parameter of the algorithm. All experiments in this paper were run on a PC with an Intel 4.00 GHz CPU and 64 GB of RAM

Table 1. Test set and crawled set information.

		Life	Edu	Travel	Tool	Ent
Test set	# Apps	802	806	890	840	760
	# Class	67	35	19	52	30
	# Class (top level)	43	23	15	38	24
Crawled set	# Apps	79,122	96,393	42,858	86,784	91,564
	# Class	3,865	4,399	2,019	3,302	4,411

3. Comparable Clustering or Classification Methods

We compare the clustering performance of the proposed method with two renowned and efficient algorithms: the K-means and LDA algorithms. We also compare our algorithm with agglomerative hierarchical clustering against various merging metrics. To compare the algorithms in a fair manner, we set the number of clusters as the golden standard and use the irace package [35] to tune the hyper-parameter of each algorithm with the lifestyle category. The details of the comparable methods are as follows:

- The K-means and LDA algorithms are respectively learned by Sklearn [36] and the Gensim tool [37]. The performances are measured on the average of ten experimental results.
- For agglomerative hierarchical clustering, we use complete linkage (CHC), average linkage (AHC), and ward metric with Sklearn [36].
- The proposed model is described in Fig. 1. In a linear combination for a cluster merge, the weights of the label similarity, the cluster centroid similarity, and the cluster overlap are optimized for each vector representation. Similar clusters are merged until the number of clusters reaches the golden standard.

To analyze the effects of the proposed document representation method, the following methods are compared.

- TF indicates the term frequency vector of a document. For the proposed model, α, β, and threshold ε for an expanding cluster is optimized as 0.47, 0.22, and 0.52, respectively.
- RCF indicates a relative category frequency vector of a document, as described in (1). For the proposed model, α, β, and ε are optimized as 0.51, 0.19, and 0.63, respectively.
- The LDA algorithm can be used to represent the document as a topic-score based vector [38]. The number of topics is set to 124. For the proposed model,

https://doi.org/10.4218/etrij.17.0116.0936

α, β, and ε are optimized as 0.35, 0.46, and 0.4, respectively.

- *Doc2vec* represents a document as a document vector using the *doc2vec* algorithm, and it is learned using the Gensim tool [37]. The vector dimension is set to 300. The length of the window is set to eight, and the number of negative examples is set to five. For the proposed model, α, β, and ε are optimized as 0.31, 0.45, and 0.43, respectively.
- *Rdoc2vec* represents a document as a document vector using the retrofitted weight matrix. Up to 50 positive examples are randomly selected from the documents within the same cluster. The same number of negative examples is also randomly selected from the other cluster. The value of η is set to 0.34. For the proposed model, α, β, and ε are optimized as 0.32, 0.49, and 0.41.

In addition, we evaluate the performance in classifying the applications using the training set automatically constructed from the crawled set. For the classification performance evaluation, the proposed document vector similarity based approach is compared with naive Bayes, SVM, and the one-class SVM classifier.

- The naive Bayes classifier is learned using Sklearn [36]. We set the alpha as one.
- An SVM classifier is learned using Lee's SVM tool [39] with a linear kernel and 1,000 as the cost value.
- One-class SVM [40] is learned with a linear kernel. We set the cost value as 100. The main difference of a one-class SVM is that it learns from only positive examples.

4. Experimental Results

A. Clustering Performance

To compare clustering performance between the proposed model and the baseline algorithms, we use purity and entropy as evaluation measures. Because the baseline algorithms are designed for a nonhierarchical output, the top-level classes are used as the correct class without the lower level in the test set.

Purity is an external evaluation criterion of the cluster quality. Moreover, it is the ratio of the number of correctly clustered mobile apps from the number of total clustered mobile apps.

In (13), N is the number of total clustered mobile apps, k is the number of clusters, c_i. is a cluster, and t_j. is the class tag that most of the mobile apps select in cluster c_i.

$$\text{Purity} = \frac{1}{N}\sum_{i}^{k} \max_j |c_i \cap t_j|. \qquad (13)$$

Entropy measures the unexpectedness. For each resulting cluster, we can measure its entropy as follows. In (14), $\text{P}(t_j)$ is the probability of classifying a mobile application as t_j in cluster i, N is the total number of clustered mobile apps, and N_w is the number of mobile apps in cluster i.

$$\text{Entropy} = \frac{N_w}{N}\sum_{i}^{k}\sum_{j} - \text{P}(t_j)\log \text{P}(t_j). \qquad (14)$$

Table 2 shows the clustering performances based on the combinations of clustering algorithms and the document representation methods. Purity takes a value of zero to one, where a higher purity value indicates a better performance. Entropy takes a value of more than zero, where a lower entropy value describing less unexpectedness indicates a better performance.

In Table 2, the "travel and local" category shows a better performance than the other categories because the number of clusters in this category is less than those of the other categories, and the keywords are easily found in the title and description.

The proposed model shows a better performance than the K-means algorithm in a comparison of most of the document representation methods. For the average performance of the five categories, the document representation with *rdoc2vec* increases by 0.19 in terms of purity, and decreases by 1.18 in terms of entropy when using the proposed model instead of the K-means algorithm.

Table 2 shows that the LDA algorithm performs better than any instance of the K-means algorithm. However, the proposed algorithm tends to overwhelm the LDA algorithm except for frequency-based document representation methods, such as TF or RCF. In addition, the proposed model using LDA as the document representation shows a better performance than using the document topic weight from LDA for direct clustering.

The performance difference between the two document representation methods, TF and RCF, is not significant. With the K-means algorithm, the document representation method RCF shows a better performance than the document representation method TF in terms of both purity and entropy. On the other hand, with the proposed model, the document representation method RCF shows a better performance than the document representation method TF for purity, whereas the document representation method TF shows a better performance than the method RCF for entropy.

The performance of the hierarchical clustering is generally poor except for the ward metric. With this

Table 2. Clustering performance estimated based on purity and entropy.

Category	Purity						Entropy						Avg Time (s)	
	Life	Edu	Travel	Tool	Ent	Avg	Life	Edu	Travel	Tool	Ent	Avg	Train	Test
K-means (TF)	0.39	0.39	0.70	0.28	0.32	0.42	3.25	2.67	1.26	3.85	3.12	2.83	202.87	35.93
K-means (RCF)	0.42	0.41	0.70	0.37	0.40	0.46	3.13	2.56	1.27	3.34	2.98	2.66	309.64	31.27
K-means (LDA)	0.47	0.49	0.76	0.39	0.38	0.50	2.52	2.37	1.05	2.90	2.74	2.32	10718.5	19.93
K-means (d2v)	0.44	0.51	0.76	0.39	0.44	0.51	2.93	2.29	1.10	3.25	2.63	2.44	1951.02	46.32
K-means (rd2v)	0.52	0.56	0.74	0.43	0.46	0.54	2.41	2.09	1.16	2.73	2.39	2.15	3128.41	53.58
LDA	0.58	0.56	0.79	0.51	0.49	0.59	1.79	1.84	0.89	2.13	2.04	1.74	316.78	33.43
CHC (rd2v)	0.39	0.42	0.49	0.34	0.35	0.40	2.88	2.45	2.18	3.34	3.04	2.78	3128.41	46.63
AHC (rd2v)	0.28	0.28	0.47	0.20	0.22	0.29	4.15	3.38	2.38	4.43	3.69	3.60	3128.41	45.83
Ward (rd2v)	0.57	0.55	0.75	0.50	0.49	0.57	2.02	2.03	1.12	2.42	2.21	1.96	3128.41	47.16
Proposed (TF)	0.47	0.52	0.85	0.51	0.55	0.58	1.97	1.91	0.47	2.2	**1.19**	1.55	202.87	112.1
Proposed (RCF)	0.54	0.46	0.73	0.67	0.57	0.59	1.87	1.98	1.21	1.37	1.42	1.57	309.64	111.27
Proposed (LDA)	0.55	**0.68**	**0.89**	0.63	0.64	0.68	1.50	1.29	0.45	1.25	1.48	1.19	10718.5	122.5
Proposed (d2v)	0.64	0.61	0.81	0.69	0.62	0.67	1.25	1.32	**0.43**	1.14	1.35	1.1	1951.02	90.76
Proposed (rd2v)	**0.72**	0.67	0.86	**0.73**	**0.65**	**0.73**	**0.99**	**1.18**	0.48	**0.94**	1.28	**0.97**	3128.41	102.58

algorithm, using the *rdoc2vec* method shows the highest performance on average.

Compared with frequency-based document representation methods, the LDA and *doc2vec* based methods show better performances regardless of the clustering algorithms. This indicates that the way in which the document characteristics are represented is important. Although the difference in performance between LDA and *doc2vec* is not significant, *doc2vec* shows less performance deviation per category than LDA.

Compared with other document representation methods, the proposed document representation method *rdoc2vec* performs better regardless of the clustering algorithms. For the average performance level based on the five categories, the document representation *rdoc2vec* increases by 0.06 in terms of purity, and decreases by 0.13 in terms of entropy in comparison with the document representation *doc2vec*. This indicates that the performance can be improved by retrofitting the document embedding using the training corpus and constraint.

Table 3 shows the top-five nearest neighbor mobile applications for each of the three mobile applications. The bold type indicates truly similar mobile applications, which indicates that the proposed retrofitted document embedding method *rdoc2vec* can find more truly similar mobile applications than the conventional document embedding method *doc2vec*.

Figure 4 shows the change in performance of the proposed method according to α and β. The difference between the minimum and maximum purities is 0.13, whereas the difference between the minimum and maximum entropy is 0.52 according to the values of α and β, respectively. The clusters merged by both the word similarity and cluster similarity together showed a better

Table 3. Top-five nearest neighbor apps with *doc2vec* and *rdoc2vec*.

App	Similar apps (*doc2vec*)	Similar apps (*rdoc2vec*)
Nail Fashion	• Nigerian Fashion Gallery • App Uninstaller Mechanics • Art of Nail Decoration • Monster Jump Free • Colorful and Fashion • Helper	• **Art of Nail** • **Decoration Nail Art Trend** • **Beautiful Nail Designs** • **Nail Decoration** • **Nail 2 Go**
Horoscope	• Strobe Light • Living Room • Easy to Use Calculator • Love Calculator • **My Daily Horoscope Positive**	• **Horoscope** • **Daily Horoscope** • **The Horoscope** • **Horoscope** • **Cancer Horoscope**
My Diary	• GENIUSGAME • FREE Anxiety & Panic Relief • Should I Date You? • New Year Gift	• **WePhoto: Diary** • Baby Sounds • **Diary Q** • **Picture Diary** • **Contact Diary**

https://doi.org/10.4218/etrij.17.0116.0936

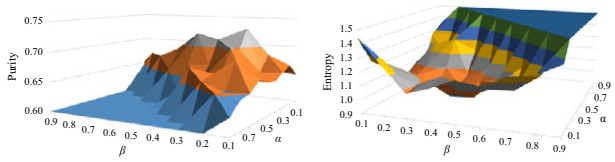

Fig. 4. Results of parameter optimization.

level of performance than the clusters merged by only one of them (when α or β is zero).

The average training time of *rdoc2vec* is approximately 3,128 s. Considering that it learns the vector from tens of thousands of documents, this is a reasonable time frame for large data. It requires less than 2 min to cluster approximately 1,000 mobile apps during the test set, which means *rdoc2vec* is suitable for a real environment.

B. Classification Performance

The proposed method automatically constructs the initial cluster and uses it as the training set. It then classifies the documents by comparing their similarity with the clusters. However, there is a well-known method for using the training set, namely, exploiting machine learning classifiers such as an SVM or naive Bayes classifier. The document vector similarity based approach is compared with other classification algorithms, such as the above two classifiers.

To evaluate the performance in classifying applications using the automatically constructed training set, the mean average precision is used as an evaluation measure. The mean average precision (MAP) is the mean of the average precision scores for each cluster. The mean average precision computes the mean of the average values of precision as follows.

$$\mathrm{MAP} = \frac{1}{|c|}\sum_{c}\frac{\mathrm{P}(k)\times \mathrm{rel}(k)}{\text{number of relevant documents}} \quad (15)$$

In (15), P(k) is the precision at cut-off k in the ranked list; rel(k) is an indicator function, which takes one if the item at rank k is relevant, or zero otherwise; and $|C|$ is the number of test clusters. To measure the mean average precision (MAP), the similarity score or classification scores are used to rank the similarity between documents and clusters. Table 4 shows the classification performances according to the classification algorithm applied. MAP takes a value of zero to one, where the higher value indicates a better performance.

The proposed document representation method *rdoc2vec* shows the best MAP for the average of five categories. It improves by more than 0.09 in MAP in a comparison with the three classifiers: naive Bayes, the one-class SVM, and SVM. This indicates that the proposed model is more suitable for the fine-grained classification problem than a classifier.

Table 4. Classification performance estimated by MAP.

Category	Life	Edu	Travel	Tool	Ent	Avg	Train time	Test time
TF	0.42	0.40	0.41	0.55	0.44	0.45	202.87	74.21
RCF	0.46	0.28	0.37	0.55	0.38	0.41	309.64	83.14
LDA	0.57	0.45	0.61	0.60	0.54	0.55	10718.5	78.36
doc2vec	0.64	0.49	0.56	0.72	**0.59**	0.60	1951.02	56.93
rdoc2vec	**0.70**	0.54	**0.67**	**0.74**	0.58	**0.64**	3128.41	63.67
Naive Bayes	0.54	0.58	0.46	0.60	0.55	0.55	795.99	11079.3
One-class SVM	0.50	0.49	0.51	0.39	0.46	0.47	348.72	142.13
SVM	0.58	**0.63**	0.52	0.54	0.53	0.55	26394.2	264.31

The classifiers perform better than the frequency-based document representation methods, TF and RCF. On the other hand, the classifiers show a lower performance level than the neural-network-based document representation methods, *doc2vec* and *rdoc2vec*, for the following reasons. Because the fine-grained clustering problem for mobile applications requires a large number of clusters, it is difficult to prepare a sufficiently large training corpus for every cluster. In the initial clusters used for the training corpus, some clusters include only 10 to 30 mobile applications. In addition, the hierarchical classification structure can be unfavorable to the classifiers. The one-class SVM classifier is not affected by the hierarchical classification structure because it uses only positive examples. However, the performance of one-class SVM is generally lower than that of a conventional SVM, and it cannot overcome the limitation of a small-sized training set.

To confirm the classifier performance differences according to training corpus size, we separated the initial clusters used for the training corpus based on the number of mobile applications in each cluster. Figure 5 shows that the MAP of the classifiers depends on the training corpus size. Specifically, the naive Bayes classifier learned from less than 100 mobile applications shows a 0.30 MAP,

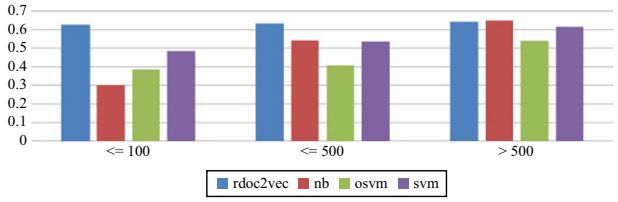

Fig. 5. Performance based on the training corpus size.

whereas the same classifier learned from more than 500 mobile applications shows a 0.65 MAP.

The proposed *rdoc2vec* method shows a stable performance regardless of the training corpus size owing to its characteristics. The classification performance highly depends on the quality of the document vector and the centroid vector, which averages the cluster document vectors. The proposed method shows good performance because each cluster centroid vector is appropriately learned, albeit the cluster is small.

VI. Conclusion

In this paper, we proposed a fine-grained mobile application-clustering model retrofitting the document embedding. The proposed model has the following characteristics. First, the proposed semi-supervised clustering algorithm clusters a huge number of mobile applications. The proposed model initializes the clusters based on the title keywords with less noise. The initialized clusters are used as a training set to learn the document embedding. The experimental results showed that the proposed model increased by 0.19 in terms of purity, and decreased by 1.18 in terms of entropy, compared with the K-means algorithm. In addition, it improved more than 0.09 for MAP in a comparison with the classifiers.

Second, we proposed a method for learning a document vector using a training set. By adding a constraint to the object function, the performance of document embedding is improved compared to the conventional *doc2vec*. Third, to merge similar clusters, we consider the label similarity of the clusters, the cluster similarity, and the cluster overlap together. Through the proposed approach, it is possible to avoid constructing costly resources, such as a synonym dictionary. In future work, we will apply the proposed fine-grained mobile-application clustering model for retrofitting the document embedding to other domains, such as an emotional analysis.

References

[1] Number of Apps Available in Leading App Stores, June 2016, Retrieved from https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

[2] H. Zhu et al., "Exploiting Enriched Contextual Information for Mobile App Classification," *Proc. ACM Int. Conf. Inform. Knowl. Manag.*, Maui, HI, USA, Oct. 29–Nov. 2, 2012, pp. 1617–1621.

[3] H. Zhu et al., "Mobile App Classification with Enriched Contextual Information," *IEEE Trans. Mobile Comput.*, vol. 13, no. 7, 2014, pp. 1550–1563.

[4] M. Lindorfer, M. Neugschwandtner, and C. Platzer, "Marvin: Efficient and Comprehensive Mobile App Classification through Static and Dynamic Analysis," *IEEE Annu. Comput. Softw. Applicat. Conf.*, Taichung, Taiwan, July 1–5, 2015, pp. 442–433.

[5] G. Berardi et al., "Multi-store Metadata-Based Supervised Mobile App Classification," *Proc. Annu. ACM Symp. Appl. Comput.*, Salamanca, Spain, Apr. 13–17, 2015, pp. 585–588.

[6] J.M. Heo and S.Y. Park, "Word Cluster-Based Mobile Application Categorization," *J. Korea Soc. Comput. Inform.*, vol. 19, no. 3, Mar. 2014, pp. 19–24.

[7] V. Radosavljevic et al., "Smartphone App Categorization for Interest Targeting in Advertising Marketplace," *Proc. Int. Conf. Companion World Wide Web.*, Quebec, Canada, Apr. 11–15, 2016, pp. 93–94.

[8] J.D. Rose, "An Efficient Association Rule Based Hierarchical Algorithm for Text Clustering," *Int. J. Adv. Eng. Techol.*, vol. 7, no. 1, Jan.–Mar. 2016, pp. 751–753.

[9] F. Beil, M. Ester, and X. Xu, "Frequent Term-Based Text Clustering," *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Alberta, Canada, July 23–26, 2002, pp. 436–442.

[10] S.S. Bedi, H. Yadav, and P. Yadav, "Categorization, Clustering and Association Rule Mining on WWW," *Multimedia, Signal Process. Commun. Techol.*, Aligarh, India, Mar. 14–16, 2009, pp. 173–177.

[11] A. Kongthon, C. Haruechaiyasak, and S. Thaiprayoon, "Constructing Term Thesaurus Using Text Association Rule Mining," in *Proc. ECTICON 2008*, Krabi, Thailand, May 14–17, 2008, pp. 137–140.

[12] S. Das et al., "Opinion Based on Polarity and Clustering for Product Feature Extraction," *Int. J. Inform. Eng. Electron. Bus.*, vol. 8, no. 5, Sept. 2016, pp. 36–43.

[13] K. Bafna and D. Toshniwal, "Feature Based Summarization of Customers' Reviews of Online Products," *Procedia Comput. Sci.*, vol. 22, 2013, pp. 142–151.

[14] S. Homoceanu et al., "Will I Like It? Providing Product Overviews Based on Opinion Excerpts," *IEEE Conf. Commerce Enterprise Comput.*, Luxembourg, Sept. 5–7, 2011, pp. pp. 26–33.

[15] Z. Zhai et al., "Clustering Product Features for Opinion Mining," *Proc. ACM Int. Conf. Web Search Data Mining*, Hong Kong, China, Feb. 9–12, 2011, pp. 347–354.

[16] M. Hegland, "The Apriori Algorithm–a Tutorial", in *Mathematics and Computation in Imaging Science and Information Processing*, Singapore: World Scientific, 2005, pp. 209–262.

[17] T. Mikolov and J. Dean, "Distributed Representations of Words and Phrases and Their Compositionality," in

https://doi.org/10.4218/etrij.17.0116.0936

Advances in Neural Information Processing Systems, MIT Press, 2013.

[18] J.H. Lau and T. Baldwin, *An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation*, July 2016, Accessed 2016. https://arxiv. Org/abs/1607.05368

[19] M.J. Kusner et al., "From Word Embeddings to Document Distances," *Proc. Int. Conf. Mach. Learn*., Lille, France, July 6–11, 2015, pp. 957–966.

[20] B. Hu et al., "Convolutional Neural Network Architectures for Matching Natural Language Sentences," *Adv. Neural Inform. Process. Syst*., Montreal, Canada, Dec. 8–13, 2014, pp. 2042–2050.

[21] Y. Kim, *Convolutional Neural Networks for Sentence Classification*, Sept. 2014, Accessed 2016. https://arxiv.org/abs/1408.5882

[22] T. Kenter and M. de Rijke, "Short Text Similarity with Word Embeddings," *Proc. ACM Int. Conf. Inform. Knowl. Manag*., Melbourne, Australia, Oct. 18–23, 2015, pp. 1411–1420.

[23] C.B. di Chen et al., "Simcompass: Using Deep Learning Word Embeddings to Assess Cross-Level Similarity," *Proc. Int. Workshop Semantic Evaluation*, Dublin, Ireland, Aug. 23–24, 2014, pp. 560–565.

[24] Q.V. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," *Int. Conf. Machin. Learn*., Beijing, China, 2014, pp. 1–9.

[25] A.M. Dai, C. Olah, and Q.V. Le, *Document Embedding with Paragraph Vectors*, July 2015, Accessed 2015. https://arxiv.org/abs/1507.07998

[26] R. Kiros et al., "Skip-Thought Vectors," in *Advances in Neural Information Processing Systems*, MIT Press, 2015.

[27] S. Wang et al., "Linked Document Embedding for Classification," *Proc. ACM Int. Conf. Inform. Knowl. Manag*., Indianapolis, IN, USA, Oct. 24–28, 2016, pp. 115–124.

[28] R. Johansson and L.N. Pina, "Embedding a Semantic Network in a Word Space," *Proc. Conf. North American Chapter Association Computational Linguistics: Human Language Technol*., Denver, CO, USA, May 31–June 5, 2015, pp. 1428–1433.

[29] S. Rothe and H. Schütze, *Autoextend: Extending Word Embeddings to Embeddings for Synsets and Lexemes*, July 2015, Aceessed 2016. https://arxiv.org/abs/1507.01127

[30] Z. Chen et al., "Revisiting Word Embedding for Contrasting Meaning," *Proc. Annu. Meeting ACL-IJCNLP*, Bejing, China, July 26–31, 2015, pp. 106–115.

[31] Q. Liu et al., "Learning Semantic Word Embeddings Based On Ordinal Knowledge Constraints," *Proc. Annu. Meeting ACL-IJCNLP*, Bejing, China, July 26–31, 2015, pp. 1501–1511.

[32] M. Faruqui et al., *Retrofitting Word Vectors to Semantic Lexicons*, Mar. 2015, Accessed 2016. https://arxiv.org/abs/1411.4166

[33] A. Mnih and K. Kavukcuoglu, "Learning Word Embeddings Efficiently with Noise-Contrastive Estimation," in *Advances in Neural Information Processing Systems*, MIT Press, 2013.

[34] Viennot N., Garcia E., and Nieh J., "A Measurement Study of Google Play," *ACM SIGMETRICS Performance Evaluation Rev.*, vol. 42, no. 1, 2014, pp. 221–233.

[35] M. López-Ibánez et al., "The Irace Package, Iterated Race for Automatic Algorithm Configuration," Université Libre de Bruxelles, Belgium, Technical Report TR/IRIDIA/2011-004, IRIDIA, 2011.

[36] F. Pedregosa et al., "Scikit-Learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, Oct. 2011, pp. 2825–2830.

[37] R. Rehurek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," In *Proc. LREC Workshop New Challenges NLP Frameworks*, Malta, 2010, pp. 46–540.

[38] G. Peng et al., "K-means Document Clustering Based on Latent Dirichlet Allocation," In Proc. WDSI, Las Vegas, NV, USA, Apr. 5–9, 2016.

[39] C.K. Lee and M.G. Jang, "A Modified Fixed-threshold SMO for 1-Slack Structural SVM," *ETRI J.*, vol. 32, no. 1, Feb. 2010, pp. 120–128.

[40] C.K. Lee, "1-Slack One-Class SVM for Fast Learning," *J. KIISE*, vol. 19, no. 5, 2013, pp. 253–257.

Yeo-Chan Yoon received BS and MS degrees in computer science and engineering from Korea University, Seoul, Rep. of Korea, in 2004 and 2007, respectively. Currently, he is a senior researcher at the Electronics and Technology Research Institute (ETRI), Daejeon, Rep. of Korea. His research interests include digital content recommendation systems, natural language processing, machine learning, and big data analytics.

Junwoo Lee received a BS degree in resource engineering and an MS in electronic engineering from Hanyang University, Seoul, Rep. of Korea, in 1996 and 1998 respectively. Since 1999, he has been a research engineering staff member of the Smart Contents Research Laboratory at ETRI, Daejeon, Rep. of Korea. His research interests include intelligent content service recommendation, virtual and mixed reality, and HCI regarding digital contents.

So-Young Park received a BS degree in computer science and engineering from Sangmyung University, Cheonan, Rep. of Korea, in 1997, and MS and PhD degrees in computer science from Korea University, Seoul, Rep. of Korea in 1999 and 2005, respectively. She has been an associate professor in the Department of Game Design and Development at Sangmyung University since 2007. Her current research interests include natural language processing and data mining.

Changki Lee received a BS degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Rep. of Korea in 1999. He received MS and PhD degrees in computer engineering from Pohang University of Science and Technology, Rep. of Korea in 2001 and 2004, respectively. From 2004 to 2012, he was a researcher with ETRI, Daejeon, Rep. of Korea. Since 2012, he has been a professor of Computer Science at Kangwon National University. His research interests include natural language processing, machine learning, and deep learning.