ISSN 2007-9737

RelEmb: A Relevance-based Application Embedding for Mobile App Retrieval and Categorization

Shubham Krishna*, Ahsaas Bajaj*, Mukund Rungta, Hemant Tiwari, Vanraj Vala

Samsung R&D Institute, Bengaluru,
India

{shubham.k1, ahsaas.bajaj, mukund.r, h.tiwari, vanraj.vala}@samsung.com

Abstract. Information Retrieval Systems have revolutionized the organization and extraction of Information. In recent years, mobile applications (apps) have become primary tools of collecting and disseminating information. However, limited research is available on how to retrieve and organize mobile apps on users' devices. In this paper, authors propose a novel method to estimate app-embeddings which are then applied to tasks like app clustering, classification, and retrieval. Usage of app-embedding for query expansion, nearest neighbor analysis enables unique and interesting use cases to enhance end-user experience with mobile apps.

Keywords. Information systems and retrieval, mobile applications, application embedding.

1 Introduction

Recent years have seen tremendous increase in usage of mobile applications, a.k.a. *apps*, mainly due to the ever rising popularity of smart phones. There are millions of apps [26] freely available on Google Play Store and Apple App Store. With the abundance of data available in form of applications, it is important to build efficient and effective retrieval engines around them. When a user queries for an application, it is crucial to bring the most relevant application at the top. In a mobile environment, the user expects search operation to be rapid and relevant. This is a challenge as the number of available applications are readily growing these days. Most of the recent work in the field of Information Retrieval has been focused on web-search scenarios and improving the ranking of web results. Little work is focused on information retrieval centered around mobile applications. Intelligent retrieval methods are required to make sense of this large amount of app data and also keep them organized in users' devices. Since most of the data related to applications are in the form of its description, it is important to mine this source of information. Very recently, neural word embedding has found its use in the field of Information Retrieval. A novel method of learning word embedding from app description is proposed in this paper. The paper is organized as follows.

An overview of related work in the domain of information retrieval is presented in Section 2. The details of estimating the embeddings are discussed in section 3. Section 4 elaborates experiments and results using the proposed embeddings for various tasks. Finally section 5 recapitulates the proposed approach with current applications and scope of future extensions.

2 Related Work

In past few years, word embeddings have found a major role for solving different tasks in multiple domains like Natural Language Processing and Machine Learning. Word embedding is basically a representation of a word in a vector space and there are different ways to learn this representation- to model semantic or syntagmatic relationships. Traditionally, techniques like Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) were applied to

* indicates equal contribution of the authors.

generate dense word representations. With the advent of neural networks, it became possible to learn more enhanced word representations. Different variants of word embeddings such as word2vec [16], GloVe [20] have been proposed for learning dense distributed representation of words. These window based methods count the number of co-occurrences of terms in the training corpus and suggest how likely is a term to occur with other terms. Such methods have been highly effective in solving different problems such as Text Summarization, Sentiment Analysis, and Machine Translation. Later, the idea of neural word embeddings was extended to learn document-level embeddings [13] which opened further interesting use-cases. Information Retrieval is one such domain.

In 2015, a unified framework for monolingual and cross lingual information retrieval was formulated using word embeddings [28]. Dual Embedding Space model using word embeddings was proposed for the task of document ranking [17]. Also, there are methods for solving the problem of term weighting and query interpretation using distributed representation of words [31]. Word and document embeddings have also found their use in applications such as query expansion and estimation of accurate query language models in the task of retrieval [29]. In general, query expansion is based on pseudo relevance feedback [22] to enhance the performance of ad-hoc retrieval. Initial work in query expansion utilized word2vec (continuous bag-of-words) embedding approach [12].

Currently, researchers are experimenting to build robust word embeddings to analyze their usage in different problems of information retrieval. The novelty of locally-trained word embeddings are nicely highlighted in [7]. They have shown that these locally trained word embeddings outperform the globally trained ones (like word2vec, GloVe, etc.) for retrieval tasks. It isn't necessary that what a word means globally also means the same in a local context. Building on the same line of work, relevance-based word embeddings [30] were explored to propose two new cost functions to learn word embedding specifically for retrieval tasks like query expansion and classification.

These days, mobile applications form a large chunk of data which is available for consumption. Some work has been done to mine user-reviews on mobile applications [27]. Mobile app retrieval has been experimented by few researchers to learn application representation using topic modeling [19] and intention in social media text [18]. Very recent work has been done to build a unified mobile search framework using neural approaches to learn high-dimensional representations for mobile apps [1]. Also, there are recent attempts made to use Learning-to-Rank algorithms for optimizing retrieval of mobile notifications triggered by installed apps [3]. However, application embeddings have been rarely utilized to perform query expansion, nearest neighbor analysis and other tasks related to mobile applications.

This paper proposes a novel method to learn dense word embeddings from app descriptions. The learned word embeddings are then used to compute relevance-based application embeddings which are suitable for retrieval and categorization of Mobile Apps.

3 Proposed Method

The following section details the method to extract word embeddings from app descriptions. As discussed in section 2 embedding techniques like word2vec and Glove are not suitable for information retrieval tasks as they are based on co-occurrence and fail to capture the associated relevance. It is observed that task specific embeddings generally perform better. To improve performance in tasks related to retrieval, it is important to learn these representations which carries the notion of relevance. With this motivation, the following subsection 3.1 discusses the process of learning this representation (word embedding) for each unique word in the vocabulary.

Once this sparse representation is calculated, it is used for learning a distributed vector representation using neural networks. This reduces noise, amplifies patterns and is suitable for mobile devices as computations involving dense vectors are faster and requires less space for storage. The approach proposed in this paper is language-independent as while formulating the

method it is assumed that application descriptions can be of any language and is seen as collection of words. Therefore, same techniques can be applied if datasets for different languages are available for training the models. For the methods discussed below, the terms - application embedding and document embedding are used in the same context.

3.1 Relevance-Based Word Representation

As it is said in linguistic theory [8]: "You shall know a word by the company it keeps". For the domain of information retrieval, it can be said that a word is known by the documents (or other words) it retrieves. In order to capture the relevance associated with each word, the information contained in the top retrieved documents are utilized when the same word is used as a query to retrieval engine. Apache Lucene [2], an open source search library is used to test this hypothesis and get the relevant results for each word in the vocabulary. Lucene uses Okapi-BM25 score [21] to rank the retrieved documents. Using the app descriptions for all applications, Vector Space Model (VSM) representation is constructed using the term frequency [14] and inverse document frequency [25] (tf-idf) scores.

In this manner, every application is represented by a vector of dimension $1 \times N$, where N is the vocabulary size of the corpus. The weight vector for application j is defined as [24]:

$$VSM_j = [w_{1,j}, w_{2,j}, ..., w_{N,j}], \tag{1}$$

$$w_{ij} = tf_{ij} \cdot \log \frac{TotalDocs}{\mid \{j' \epsilon Doc \mid i \epsilon j'\} \mid}, \tag{2}$$

where, $tf(i,j)$ is the term frequency of term i in application j. $TotalDocs$ is the total number of documents (applications) in the corpus. $\mid \{j' \epsilon\ Doc \mid i\ \epsilon\ j'\} \mid$ is the total number of applications containing the term i. $\log \frac{TotalDocs}{\mid \{j' \epsilon\ Doc \mid i\ \epsilon\ j'\} \mid}$ is inverse document frequency. For a query t with words $t_1, t_2, .., t_{max}$, $BM25$ score of document (application j) is computed as:

$$BM25(t,j) = \sum_{i=1}^{max} idf(t_i) \cdot \frac{tf(t_i,j) \cdot (k_1+1)}{f(q_i,j) + k_1 \cdot (1 - b + b \cdot \frac{|j|}{avgdl})}, \tag{3}$$

$\mid j \mid$ is the length of document Doc in words, k_1 and b are hyper-parameters and $avgdl$ is the average document length.

3.1.1 Method

Firstly, the most relevant documents ($topDocs$) are retrieved after querying the system with each term in the vocabulary. The number of top documents is an input provided to the system and is given by the variable $len(topDocs)$. For the experiments in this paper $len(topDocs)$ is set to 10. Using the constructed VSM and $BM25$ scores for these $topDocs$, a high-dimensional word representation is computed by the following equation:

$$WordRepr_k = \frac{\sum_{i=0}^{len(topDocs)} BM25_{i,k} \cdot VSM_i}{\sum_{i=0}^{len(topDocs)} BM25_{i,k}}, \tag{4}$$

where, VSM_i is the Vector Space Model representation for the i^{th} application and $BM25_{i,k}$ is the $BM25$ score of the i^{th} document for term k. In this manner, word embeddings of all words in the vocabulary are computed. This word embedding takes into account the relevance of word in accordance with the context of data.

3.2 Dense Word Embedding

Curse of Dimensionality is a well known phenomenon in the field of Machine Learning and Data Mining. After a certain point, the increase in number of dimensions hurts the performance of algorithms. Inferring and performing computations on sparse vectors and matrices are costly operations as they may contain noise and irrelevant information. Even the vector given by equation 4 is sparse in nature which makes its usage limited. Hence, dimensionality reduction is used to remove noise and extract useful patterns. Different algorithms like Principal Component Analysis, Singular Value Decomposition, and Neural Networks are being used for performing the task of dimensionality reduction.

Firstly, to learn word embeddings from word representations, a simple yet effective feed-forward neural network architecture is used. The architecture is shown in Figure 1.

The input fed to the network is a N dimensional one hot encoded vector, where N is the vocabulary size and the output layer is N dimensional word representation calculated in equation 4. The activation functions used in the hidden and the output layer are ReLU and softmax respectively. Adam Optimizer has been used for updating the parameters of the neural network and the loss function used is cosine distance. The neural network tries to learn latent patterns present in the word representation while reducing dimension, making it computationally faster and cheaper for performing different retrieval tasks. For training the word embeddings, unique words from the app descriptions are used and passed as one hot encoded vector (input vector). The corresponding word representation is fed as the output of neural architecture. The dimension of the hidden layer is set to 300. Tensorflow framework [10] is used for the training process. For vocabulary size N, let the two weight matrices be W_1 and W_2 and t_j be the one-hot encoding of term j (input vector):

$$WordEmb_j = ReLu(t_j \times W_1), \quad (5)$$

where W_1 has the dimension of $N \times 300$ and $ReLu(x) = max(0, x)$ as given in [9]. The output layer is given by:

$$softmax(WordEmb_j \times W_2) + bias, \quad (6)$$

where W_2 has the dimension of $300 \times N$ and $softmax$ is the activation function [5]. $bias$ is a vector of dimension $1 \times N$. The cosine distance loss is computed against the values from equation 6 and the representation stored in $WordRepr_j$. After training, the hidden layer gives the word embedding $WordEmb$ for term j. This relevance based embedding has been developed keeping in mind the notion of relevance. This embedding is not intuitively built for machine learning tasks such as classification and clustering. To tackle problems such problems, a vanilla auto-encoder is trained to learn the word embeddings from the word representations proposed in equation 4. The input and output to this auto-encoder are the word representations given by equation 4. The embeddings learned in the hidden layer of the auto-encoder are denoted by $WordEmbAE$ for the rest of this paper.

With the discussed $WordEmb_j$ representation for each term j in vocabulary, an embedding matrix is constructed.

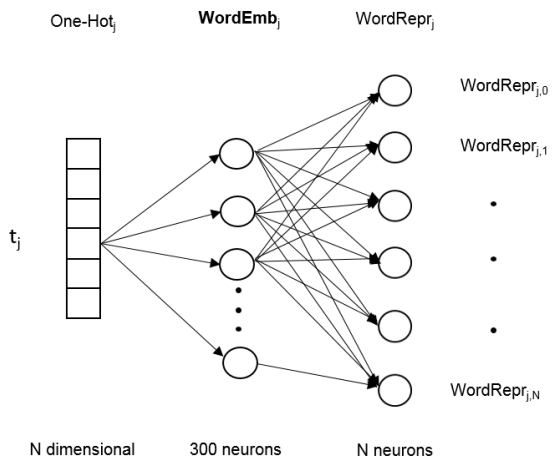

Fig. 1. Neural architecture to generate Word Embedding $WordEmb$ from Word Representation $WordRepr$

3.3 RelEmb: Application Embedding

The intention behind any user query is to get the most relevant applications. Application title or category is not sufficient to extract relevant applications. But, each application has an associated description, which carries the most useful information about the application and its features. This description can be considered as bag of words and can be used to generate word embeddings as discussed in sections 3.1 and 3.2. The learned word embeddings can be extended for calculating application-level embedding by aggregating the word embedding of each term in the application's description. Let an application k (document) be represented as $description(k) = \{ w_1, w_2, w_3, ..., w_i, ..., w_n\}$ where each w_i is the word from vocabulary.

$$RelEmb_k = \frac{\sum_{i=0}^{len(description(k)} WordEmb[w_i]}{len(description(k))}. \quad (7)$$

Similarly, when the word embeddings are learned using vanilla auto-encoder, application embed-

dings are given by:

$$RelEmbAE_k = \frac{\sum_{i=0}^{len(description(k)} WordEmbAE[w_i]}{len(description(k))}. \tag{8}$$

4 Experimental Details

To evaluate the performance of the proposed method, extensive experiments are performed for different tasks related to application retrieval and categorization. A publicly available apps dataset is used for testing the performance of the embeddings. The dataset includes the data for query-application relevance judgment, which is useful to test the retrieval task of Query Expansion. A qualitative experiment in terms of nearest neighbor analysis is also elaborated in this paper. This shows the capability of app embedding for tasks like application recommendation. The dataset also includes a category tag for each application which can be used to analyze the performance of application embeddings for the task of multi-class app classification. Apart from supervised learning, these embedding vectors can also aid in unsupervised tasks like app clustering. The evaluation of embeddings has been discussed in this section. Results indicate superior performance as compared to existing state-of-the-art methods for various tasks.

4.1 Dataset

Data Set for Mobile App Retrieval [19] (TIMAN dataset) is used for evaluation of the proposed methods. This data include information about 43,041 mobile apps including the title, description, category, package name, user-reviews, query-document relevance pairs, etc. To trim down the vocabulary size, only English words from the app descriptions are selected. Moreover, minimum permissible length for each word was set to two. With the above mentioned preprocessing, the number of unique apps comes down to 42,895 with a vocabulary size of 37,213.

4.2 Application Retrieval

In mobile apps scenario, user generally queries with short-text with mostly 1-2 terms. It becomes a challenge for any retrieval engine to bring the most relevant results at the top ranks. Solution like query expansion helps to re-formulate the user query which eventually improves the retrieval accuracy. But, it is also problematic to bring the useful expanded terms which increases the relevance of the results. The word embeddings which are discussed in sections 3.1 and 3.2 are based on pseudo-relevance feedback (BM25) and are trained in a manner suitable for retrieval tasks. Authors believe that these embeddings are capable of finding the best expansion terms for user-query.

4.2.1 Evaluation with Query Expansion

Query expansion is a standard retrieval task which has its practical advantages. It boosts the precision of a retrieval engine using a two-pass method. Query expansion approach is also an appropriate methodology to validate the performance of a retrieval system. Using the word embeddings proposed in section 3, query expansions tasks are evaluated. TIMAN dataset also provides the query-application relevance data with 4533 instances. The relevance score is a real-valued number in the range of 0-2. Since, it is a floating point (and not just binary) relevance label, $NDCG$ metric [11] is used to judge the effectiveness of query expansion. The number of expanded terms is given by the variable k. To find the expansion terms for a given query, the following methodology is used:

— Calculation of input query vector using multiple terms in the user-query $query = \{w_1, w_2, ...\}$.

$$Q = \frac{\sum_{i=0}^{len(query)} WordEmb[w_i]}{len(query)},$$

where, $WordEmb[w_i]$ are calculated in equation 5.

— Given the query vector Q, the most semantically similar terms are found by calculating cosine similarity on the $WordEmb$ matrix.

Table 1. Evaluation results on Query Expansion

Metric	Okapi-BM25	Expansion Techniques (k=5)			
		SVD	PCA	WordEmbAE	WordEmb
NDCG@3	0.569	0.499	0.467	0.572	**0.577**
NDCG@5	0.542	0.502	0.469	0.548	**0.563**
NDCG@10	0.535	0.505	0.477	0.542	**0.556**

— Top k terms from the $WordEmb$ matrix are selected based on their values of cosine similarity with respect to the input query vector Q.

— Example: $query = \{music, stream, airplay\}$ retrieves top five expansion terms as $\{sonos, bose, dlna, player, listen\}$. It can be seen that these terms are closely related to the initial user query and all of them deal with music and streaming services.

Query expansion results are shown in Table 1. Okapi-BM25 scores are used as baseline by considering the input query $\boldsymbol{Q}$ as the direct query to the retrieval system (without expansion). The novelty of the proposed word embeddings can be seen from the increase in NDCG scores after query expansion.

$WordEmb$ (equation 5) performed better than $WordEmbAE$ as the intuition behind learning these embedding was the notion of relevance for the tasks in information retrieval. Query expansion is the right measure to validate this claim and the results fully support it. Even then, $WordEmbAE$ was able to out-perform the baseline BM25 scores, proving the benefits of the sparse embeddings discussed in section 3.1.

4.2.2 Nearest Neighbor Analysis

Not only quantitatively, but also qualitatively, the proposed application embeddings have proved its worth. It is not always the case that user provides a query term for app retrieval, a user may want to get some recommendations based on a particular app. Similar to [4], nearest neighbor analysis acts as recommender tool to get the closest match from specific application. This recommendation system being retrieval based, $RelEmb$ can be employed as application embedding. Qualitative results for nearest neighbor analysis using $RelEmb$ (equation 7) are shown in the Table 4. From the results it can be seen that for an application of a particular category (query), the closest matched apps mostly belong to the same category. For these selected applications, $RelEmb$ embeddings are plotted in a two-dimensional space using t-SNE visualization [15] (shown in Figure 2). The visualization also shows accurate grouping for different categories of applications. This section shows the effectiveness of application embedding for extracting similar apps.

Fig. 2. Visual representation for applications and their categories given in Table 4

4.3 Application Categorization

Under Application categorization, classification and clustering are the two subdomains, which are most useful. Both classification and clustering use application embeddings as the feature vector for training.

Table 2. Evaluation results on App Classification (41 categories)

Classifier	F1 Score (Percentage)		
	Doc2Vec	RelEmb	RelEmbAE
Decision Tree	8.536	10.351	**21.332**
Multi-class SVM	32.550	40.557	**43.866**

Table 3. Evaluation results on Clustering Techniques

Metric	Different Methods		
	Doc2Vec	RelEmb	RelEmbAE
Silhouette Score	-0.0903	-0.061	**0.289**
Davies Bouldin Score	4.84	4.376	**0.8947**

Classification is a supervised learning algorithm which uses the document (or application) embedding as feature vector and the application category as the output variable. Trained application classifier can act as black box to predict the category of any new application. On the other hand, clustering is an unsupervised approach based on just the feature vectors. It can be used for grouping similar applications together, which has many practical use-cases like folder creation for grouping similar apps in mobile.

4.3.1 App Classification

Decision tree and multi-class SVM are used to train the classifiers with features being application embedding and labels as application category. The considered TIMAN dataset has application data divided into 41 categories. To evaluate the performance of the proposed $RelEmb$ and $RelEmbAE$, the dataset is divided into training and test splits using K-Fold cross validation (k = 5). Multi-class classification using Doc2Vec embedding approach [13] is used as baseline for comparison. Evaluation results are presented in Table 2 and the reported F1-score is computed by averaging F1-score from all 5 groups (after K-Fold). For multi-class SVM classifier, the F1-score (in percentage) for $RelEmbAE$ is 43.866 outperforming the baseline Doc2Vec, which gives the F1-score of 32.55. Similarly, for Decision tree the F1-score (in percentage) for $RelEmbAE$ is 21.332 whereas for baseline Doc2Vec it is 8.536. The results indicate the better performance of $RelEmbAE$ over Doc2Vec and also $RelEmb$.

4.3.2 App Clustering

K-mean and DBSCAN clustering algorithms are used to evaluate the performance of application embedding for the task of clustering. The results with proposed $RelEmb$ and $RelEmbAE$ are compared with the baseline Doc2Vec embeddings. Two different metrics, such as Silhouette [23] and Davies Bouldin [6] scores are used to compare the results and the evaluation results on App Clustering are shown in Table 3. It is known that the value of silhouette score should be more closer to 1 for an accurate clustering. The scores for $RelEmbAE$ are positive indicating better clustering in comparison to Doc2Vec for which the scores are negative, indicating wrongly assigned clusters. For Davies Bouldin score, a value closer to zero means a better separation between the clusters indicating superior performance of clustering. The numbers for both the clustering techniques indicate that $RelEmbAE$ outperforms existing Doc2Vec embedding by significant margins.

As discussed in section 3.3, $RelEmbAE$ is more suitable for categorization tasks whereas $RelEmb$ performs well for retrieved-based tasks such as query expansion. For classification and clustering tasks, $RelEmb$ still beats the state-of-the-art

Table 4. Qualitative results with Nearest Neighbors Analysis using $RelEmb$

Application	**Predicted Nearest Applications - Name (Category)**				
cops robbers jail break **(Action)**	survival hungry games (Action)	dead target zombie (Action)	wanted survival games (Action)	cube duty ghost blocks (Action)	orange block prison break (Action)
real piano **(Music)**	drum (Music)	real guitar (Music)	congas bongos (Music)	tabla (Music)	hip hop beatz (Music)
relax rain nature sounds **(Lifestyle)**	relax night nature sounds (Lifestyle)	relax forest nature sounds (Lifestyle)	white noise (Health & Fitness)	pain depression (Medical)	melodies sleep yoga (Health & Fitness)
love phrases images **(Social)**	top good night images (Social)	top good morning images (Social)	themes classic (Personal-ization)	status messages (Social)	video chat friendcaller (Communi-cation)
kroger **(Shopping)**	ralphs (Shopping)	smith (Shopping)	king soopers (Shopping)	fred meyer (Shopping)	dillons (Shopping)

Doc2Vec approach showcasing the significance of initial relevance-based word representations learned in section 4 and denoted by $WordRepr$.

5 Conclusion and Future Work

In this paper, word embeddings are learned with a neural network architecture using the description of mobile applications. These embeddings are developed by keeping in mind the increasing usage of mobile applications and the difficulties faced to find out relevant ones from a large collection. The learning of word embeddings is carried out based on the notion of relevance. The results show that the learned word embeddings are effective for query expansion task eventually making the search experience more user friendly. In addition, the learned word embeddings are also aggregated to find RelEmbAE and RelEmb- distributed and dense representations of mobile application. These embeddings have outperformed Doc2vec on tasks like app classification and clustering.

In future, other parameters like application reviews, ratings, etc. can be used to make the application embedding much more rich and descriptive. Another extension of this work can be to analyze the use of application embeddings in the field of query intent detection, query classification which are current topics of research in Information Retrieval. Although the work in this paper is focused on tasks related to mobile applications, the same techniques can be applied to any generic scenarios of Information Retrieval.

References

1. **Aliannejadi, M., Zamani, H., Crestani, F., & Croft, W. B. (2018).** Target apps selection: Towards a unified search framework for mobile devices. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, ACM, New York, NY, USA, pp. 215–224.
2. **Apache (2019).** Lucene.

3. **Bajaj, A., Tiwari, H., & Vala, V. (2019).** Enhanced learning to rank using cluster-loss adjustment. *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, ACM, pp. 70–77.
4. **Biswas, A., Bhutani, M., & Sanyal, S. (2017).** Mrnet-product2vec: A multi-task recurrent neural network for product embeddings. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, pp. 153–165.
5. **Bridle, J. S. (1990).** Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*. Springer, pp. 227–236.
6. **Davies, D. L. & Bouldin, D. W. (1979).** A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, Vol. 2, pp. 224–227.
7. **Diaz, F., Mitra, B., & Craswell, N. (2016).** Query expansion with locally-trained word embeddings. *arXiv preprint arXiv:1605.07891*.
8. **Firth, J. R. (1957).** A synopsis of linguistic theory 1930-1955 in studies in linguistic analysis, philological society.
9. **Glorot, X., Bordes, A., & Bengio, Y. (2011).** Deep sparse rectifier neural networks. *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323.
10. **Google (2019).** Tensorflow. `https://www.tensorflow.org/`.
11. **Järvelin, K. & Kekäläinen, J. (2002).** Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, Vol. 20, No. 4, pp. 422–446.
12. **Kuzi, S., Shtok, A., & Kurland, O. (2016).** Query expansion using word embeddings. *Proceedings of the 25th ACM international on conference on information and knowledge management*, ACM, pp. 1929–1932.
13. **Le, Q. & Mikolov, T. (2014).** Distributed representations of sentences and documents. *International Conference on Machine Learning*, pp. 1188–1196.
14. **Luhn, H. P. (1957).** A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, Vol. 1, No. 4, pp. 309–317.
15. **Maaten, L. v. d. & Hinton, G. (2008).** Visualizing data using t-sne. *Journal of machine learning research*, Vol. 9, No. Nov, pp. 2579–2605.
16. **Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013).** Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, pp. 3111–3119.
17. **Mitra, B., Nalisnick, E., Craswell, N., & Caruana, R. (2016).** A dual embedding space model for document ranking. *arXiv preprint arXiv:1602.01137*.
18. **Park, D. H., Fang, Y., Liu, M., & Zhai, C. (2016).** Mobile app retrieval for social media users via inference of implicit intent in social media text. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, ACM, pp. 959–968.
19. **Park, D. H., Liu, M., Zhai, C., & Wang, H. (2015).** Leveraging user reviews to improve accuracy for mobile app retrieval. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, pp. 533–542.
20. **Pennington, J., Socher, R., & Manning, C. (2014).** Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.
21. **Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., Gatford, M., et al. (1995).** Okapi at trec-3. *Nist Special Publication Sp*, Vol. 109, pp. 109.
22. **Rocchio, J. J. (1971).** Relevance feedback in information retrieval. *The SMART retrieval system: experiments in automatic document processing*, pp. 313–323.
23. **Rousseeuw, P. J. (1987).** Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, Vol. 20, pp. 53–65.
24. **Salton, G., Wong, A., & Yang, C.-S. (1975).** A vector space model for automatic indexing. *Communications of the ACM*, Vol. 18, No. 11, pp. 613–620.
25. **Sparck Jones, K. (1972).** A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, Vol. 28, No. 1, pp. 11–21.
26. **Statista (2018).** Number of available applications in the Google Play Store from December 2009 to December 2018.
27. **Vu, P. M., Nguyen, T. T., Pham, H. V., & Nguyen, T. T. (2015).** Mining user opinions in mobile app reviews: A keyword-based approach (t). *2015 30th*

IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE, pp. 749–759.

28. **Vulić, I. & Moens, M.-F. (2015).** Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, ACM, pp. 363–372.
29. **Zamani, H. & Croft, W. B. (2016).** Embedding-based query language models. *Proceedings of the 2016 ACM international conference on the theory of information retrieval*, ACM, pp. 147–156.
30. **Zamani, H. & Croft, W. B. (2017).** Relevance-based word embedding. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, pp. 505–514.
31. **Zheng, G. & Callan, J. (2015).** Learning to reweight terms with distributed representations. *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, ACM, pp. 575–584.

Article received on 21/01/2019; accepted on 04/03/2019.
Corresponding author is Shubham Krishna.