An Efficient Application Searching Approach Based on User Review Knowledge Graph

Fang Li[1], Tong Li[2] *
1 Fan Gongxiu Honors College, 2 Faculty of Information Technology
Beijing University of Technology, Beijing, China
fraulifang@163.com, litong@bjut.edu.cn

Abstract—**Finding a software application that perfectly suits user needs is essential for improving user experiences, as well as contributing to the development of the application ecosystems. However, it is not an easy task regarding the huge number of existing applications that are available for use. In this paper, we propose to tackle this challenge by exploring valuable information from user reviews. In particular, we design a user review knowledge graph that consists of both functional information and user preferences in order to comprehensively and precisely characterize software applications. Based on such a review knowledge graph, our approach can support application search in an efficient and precise manner. To evaluate our proposal, we have collected a total of 4,370 applications and 4,396,950 pieces of reviews for constructing a comprehensive review knowledge graph and have illustrated how users and developers can efficiently retrieve applications and improve software functionality based on the knowledge graph.**

Index Terms—**NLP, Knowledge-graph, App searching**

I. Introduction

With the proliferation of types and quantities of mobile phone's application, it is increasingly difficult for users to find applications that perfectly meet their requirements through only the huge amount of application descriptions. There is an urgent need for efficient and accurate application search.

Many direct approaches to application searching have been proposed in the past decade, which are mainly based on application descriptions. Specifically, some researches mine application descriptions from App store and process data by extracting short textual application features or time series data to make application recommendation [1]–[3]. Another branch of research advocates on retrieving applications based on relevance or history of application usage [4]–[6]. Some other researchers further use a graph-based method to analyze the similarity between downloaded applications and search from the graph to improve the searching efficiency [7]. Although, the approaches mentioned above are considered as systematic and promising means to recommend useful application, they typically require a huge amount of textual data and time to obtain applications related to user needs. Moreover, the recommendations that are made based on application descriptions are not able to reflect the application's performance as they do not considering user feedback.

Application reviews contain a huge amount of customer's real opinions that are valuable for profiling users. Specifically, review analysis can obtain the user feedback regarding application functionality and qualities. We argue that the effectiveness of application search can be improved by considering information from both application descriptions and application reviews.

In this paper, we focused on extracting application features and corresponding user preference, as well as calculating the emotional distribution of users. Based on such analysis, we can meaningfully characterize applications to better support application recommendation. Specifically, we firstly retrieve application descriptions and user reviews via web crawling tool from Google Play Store[1]. By crawling the user reviews data and applying Natural Language Processing (NLP) methods [8], we obtain 4,370 applications and about 4,396,950 pieces of reviews in total. In order to promote the effectiveness of our approach, only applications that have a decent number of comments are considered in our study. We then extract the connections between user sentiment and app features based on a large scale of textual comments, and figure out the sentiment distribution accordingly. All the above information services as the foundation for construction a comprehensive Knowledge Graph (referred to as "KG" hereinafter), enabling efficient and effective application search.

Our work mainly leverages techniques of relation extraction and text classification. With the assistance of Sentiword Corpus [9] and an application features dictionary, we are able to extract the relationships between applications features and their corresponding user sentiment. Each relationship we mark a sentiment score to evaluate the tendency as well as the strength of emotions. Based on those relationships, we construct a KG containing user entity, application feature entity and sentiment relation for further clustering and searching. Using this Knowledge Graph, we can cluster applications via intelligent search, rendering multi-angle query results. In this way, we can deal with an enormous data set of App reviews and can search the application accurately.

This paper is organized as follows. Section II presents related work in which app searching has researched within software engineering. In section III we describe the methodology of this work, the collection and processing of data, and

DOI:10.18293/SEKE2020-119
* Corresponding author.

[1]https://play.google.com/store

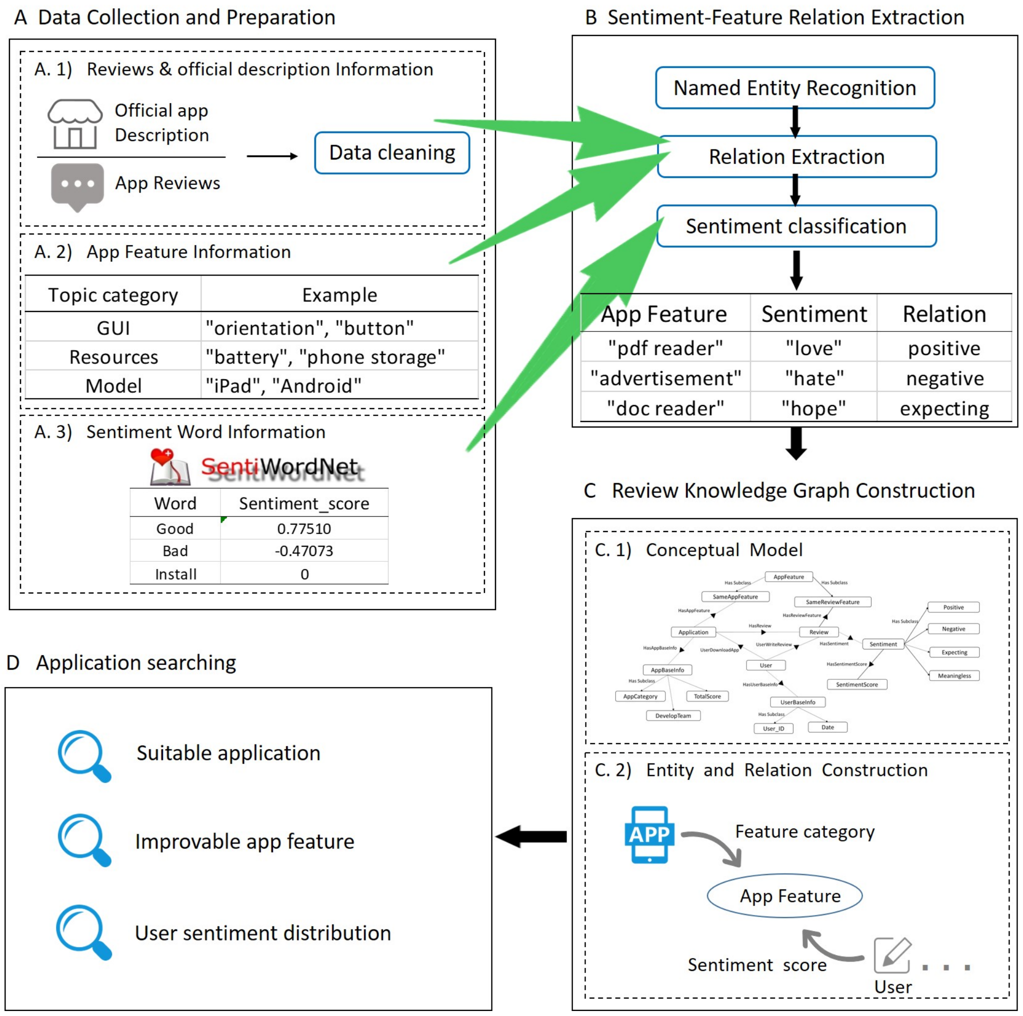

A Data Collection and Preparation
A. 1) Reviews & official description Information
Official app Description
App Reviews
Data cleaning
A. 2) App Feature Information
Topic category | Example
GUI | "orientation", "button"
Resources | "battery", "phone storage"
Model | "iPad", "Android"
A. 3) Sentiment Word Information
SentiWordNet
Word | Sentiment_score
Good | 0.77510
Bad | -0.47073
Install | 0
B Sentiment-Feature Relation Extraction
Named Entity Recognition
Relation Extraction
Sentiment classification
App Feature | Sentiment | Relation
"pdf reader" | "love" | positive
"advertisement" | "hate" | negative
"doc reader" | "hope" | expecting
C Review Knowledge Graph Construction
C. 1) Conceptual Model
C. 2) Entity and Relation Construction
APP
Feature category
App Feature
Sentiment score
User
D Application searching
Suitable application
Improvable app feature
User sentiment distribution

Fig. 1. The Framework of our paper. The solid rectangle in the figure represents a working step, the dashed rectangle represents the sub-steps in each step, the blue rectangle with rounded corners represents the technical process. And the tables and sin the figure are examples of each step.

the Knowledge Graph construction phase. In section IV we provide evaluation of our work. In section V we provide some discussion. Section VI is our conclusion.

II. Related Work

Instead of using a formal App description, App reviews are a key driver of application clustering. On the one hand, it can learn the true user's feedback for the App. On the other hand, it assists developers in finding the novel functions that can be improved. To date, the smartphone operating systems: Android and iOS have the whole worldwide smartphone shipment market share [10]. A large scale of application reviews is provided to researchers for review analysis. The history of application clustering and the use of KG in application research is discussed in this section.

Previous researches mainly solve the app searching problem by text mining and semantic awareness [11]. Jiang et al. [12] use a greedy algorithm to find the semantic awareness of the user's request and design an application order for user retrieval. Datta, Kajanan and Pervin [13] provide an independent unbiased search machine for mobile apps with semantic search capabilities. Lavid Ben Lulu and Kuflik [3] use the Machine Learning method to automated analyze functional similarity on the application's description data. Al-Subaihin et al. [14] extracted App feature using information retrieval augmented with ontological analysis to characterize apps. User history and using experience can be used for app searching because user feelings are important for app searching. Costa-Montenegro, Barragáns-Martínez and Rey-López [15] used user history to select the application similar to the downloaded apps. And Krishna et al. [4], searched a similar app using the user's history and the app description information to achieve word import. Zhu et al. [16] combine the popularity of mobile Apps, personal preference, and mobile device constraints for app searching and recommendation. Such textual data is difficult to search and manage. Park et al. [17] leverage user reviews to find out important features of apps for app retrieval. Compared with using app descriptions, brief app reviews can indicate true feedback from users and comes to more detailed app features.

A more applicable method is to use both official descriptions and user reviews for information collection and select more app features to more in line with user needs.

Graph is a kind of data structure which models a set of objects (nodes) and their relationships (edges). Recently, researches of app searching with a graph representation have been receiving more and more attention. Jisha, Krishnan and Vikraman [18] construct a Knowledge Schema - a graphic model of interconnections of data that characterize any mobile app for app searching. Bae et al. [19] incorporates a graph-based technique for application recommendation. Such a method is difficult to find out relevant applications from a large app store. To concisely record the app features and achieve multi-angled searching, we construct a knowledge-based graph containing all the application's information and the user's feedback.

Based on the previous researches, we focus on the app searching based on formal descriptions and user reviews. By analyzing the content of reviews, we can extract not only the application's features but also true feedback from users. What's more, we design a Knowledge Graph-based on the extracted results for app searching. In this way, the large scale of review data can be managed well and can be searched efficiently.

III. Methodology

Our framework is shown in Fig.1. The sequence Numbers in the figure correspond to the III.Methodology part. We firstly design a web crawling tool to grasp the App reviews and official descriptions from the Google Play store. We also prepared the sentiment word information and application feature data for Named Entity Recognition. During the data preprocessing phase, we changed words in sentences to their lowercase and removed the stop words and special symbols, then we delete the irregular reviews. For example, reviews which only contain punctuation or non-English sentence. Based on these corpora, we extracted the sentences with co-occurrence of user sentiment and App feature. To figure out the sentiment of user, we classified the relation into different categories and use the sentiment score from SentimentWordNet to indicate emotional strength. At last, we build a Knowledge Graph using these relations for results cluster and retrieval. This step contains conceptual model and entity & relation construction. The conceptual model include the user and the application's basic information and the inclusion and emotional relation of them is linked in the graph.

A. Data Collection and Preparation

According to the introduction above, the relationships of user sentiment with app features are fundamental for constructing the Knowledge Graph and their information thus need to be collected and prepared beforehand.

1) App Reviews and official description Information: We choose the Google Play store as the app repositories for its large scale and high frequency of use. We design web crawling tools to catch the app reviews, which is built upon PhantomJS[2] and Selenium[3]. There are 20 app categories, including Art & Design, Augmented Reality, Auto & Vehicles, Beauty, Books & Reference, Business, Comics, Communication, Dating, Daydream, Education, Entertainment, Events, Finance, Food & Drink, Health & Fitness, House & Home, Libraries & Demo, Lifestyle, and Game.

We extract user reviews from App Store as records and save the content, user's ID, review date and other detailed information of each review. Before we extract the user sentiment information, the textual data need to be cleaned. We changed words in sentences to their lowercase and removed the stop words such as ". , !" and special symbols. Then we dismiss the irregular reviews which may only consist of punctuation, number or reviews with messy code.

2) Application feature Information: Fine-grained categorization of app features are considered in our research. Here we create a list of 13 different topics [20]. The topics were proposed by Di S. et al., which can be a great classification category for different review topics, for example, *"I like the GUI of Crazy Bird"* is a comment on application's GUI. Table I illustrates the definition of each review topic. These topic are concluded from user reviews and they indicate the categories that user comments most frequently. Nearly all mentioned application feature in user reviews can be classified into a specific feature category. The classify tools create a new category once the review does not belong to the existing categories, and then they summarize each category into a topic. This cluster has shown to achieve a classification accuracy of 0.76.

TABLE I
Definitions of Application features

Topic	Definition
App	sentences related to the entire app, e.g., generic crash reports, ratings, or general feedback
GUI	sentences related to the Graphical User Interface or the look and feel of the app
Contents	sentences related to the content of the app
Pricing	sentences related to app pricing
Feature or Functionality	sentences related to specific features or functionality of the app
Improvement	sentences related to explicit enhancement requests
Updates/ Versions	sentences related to specific versions or the update process of the app
Resources	sentences dealing with device resources such as battery consumption, storage,etc.
Security	sentences related to the security of the app or to personal data privacy
Download	sentences containing feedback about the app download
Model	sentences reporting feedback about specific devices or OS versions
Company	sentences containing feedback related to the company/team which develops the app
Other	sentences not treating any of the previous topics

To extract the entity word from the review sentence, we construct an entity dictionary which contains all clusters words,

[2]http://phantomjs.org/
[3]http://www.seleniumhq.org/

e.g., “battery, phone storage” belong to cluster “Resources” and “Android, iOS” belong to the cluster “Model”. We create a list of entity words manually which contained 134 original words identifying the topics. To make the dictionaries more exhaustive we used Word-Net [21] to generate synonyms for all the feature words. Synonyms word were then appended to the dictionary to extract the topic entity word from a different user, such as ”price, expense and cost” all mean the amount of money for which something is sold.

3) Sentiment Word Information: For the sentiment information, we adopt the method of Gatti, Guerini and Turchi [9]. We adopt the sentiment corpus containing approximately 155,000 sentiment words called “SentiWords”, which has both high precision and coverage. We use this sentiment lexicon as our emotional words dictionary to extract user’s sentiment from reviews. Each sentiment word from lexicon has a sentiment score, ranging from -1 to +1. The absolute value of the sentiment score indicates the emotional tendency (like the application or dislike) and how strong the emotion is. When a score is a positive number, the higher the score is, the more positive the user comment is. On the contrary, the more negative it is. For example, positive word such as “Good” has the sentiment score 0.77510, negative word such as “Bad” has the score -0.47073 and the subjective word such as “install” has the sentiment score 0.

B. Sentiment-Feature Relation Extraction

In this phase, we mainly introduce how to extract the relationship of user sentiment with app features from review contents, which may have the complex grammar and unstructured sentence pattern. We utilize NLP methods to process the comments. During this process, Name Entity Recognition (NER) is significant for getting a high performance of relation extraction. We construct both user sentiment corpus and app feature dictionary. Then we use NLTK (Natural Language Toolkit)https://www.nltk.org/ to figure the sentences with co-occurrence of user sentiment and app features.

Using the sentiment information and app feature information, we dealt with the sentence relation extraction task as a classification problem. We divided the relation between the app feature and user sentiment into four main categories, positive (+), negative (-), expecting (1), and meaningless (0). More specifically, the relations are classified as “expecting” when sentiment word is in the expecting word list (which contains words expressing expected meaning, such as hope, wish, expect, etc.). Sentiment word with a score greater than 0 will be classified as “positive”, and a score less than 0 is classified as “negative”. If the above three conditions do not exist, it will be classified as “none”. For instance, if the user likes the application, We labeled it as positive to the relationship of the entity with user sentiment. If the user dislikes the application, We labeled it as negative to the relationship. If a user expects an improvement of the app feature, we labeled it as expecting to the relationship. And if the user comments without real meaning, we labeled it as meaningless to it. The example of those four categories is shown in Fig.1, which contains the user sentiment word, comment app feature word and the relation category.

Relation Extraction is a popular topic in NLP, many novel methods have been proposed with high performance. To simplify the model as well as avoid complex grammatical and semantic analysis, we here adopt the text CNN (Convolutional Neural Network) for sentence classification. In our study, we manually label part of the reviews and then utilize Text-CNN [22] to classify others automatically. The marked sentences are divided into two parts: training data and validation data. We then trained the CNN model on the marked corpus. In this way, we extract all sentences that contain co-occurrence of sentiment words and entity words. For each relationship, a sentiment score corresponds to the user’s emotion.

C. Review Knowledge Graph Construction

1) Conceptual Model: Fig.2 shows the conceptual model of the review knowledge graph, including the concept of the application, user, and some attributes. Specifically, we defined the sentiment of reviews into four basic emotions: positive, negative, expecting, and meaningless. At the same time, we also defined app features that applications and reviews have. The detailed concept definition is illustrated as follows.

For each application we choose from App Store, we collect the basic information including the app’s category, app’s total score, and app’s development team.

Each application has a lot of reviews. The user’s ID and the comment date would be selected as basic information for uniquely identifying a comment. Specifically, each review containing user sentiment for app features. Here we particularly add sentiment score for each comment to transform the textual emotion to a digitized score, which is defined as sentiment score ranging from -1 to +1. This score can directly indicate the relationship including positive, negative, meaningless and expecting. In this way, the KG can display the distribution of the user’s opinion about the app. The review is also linked with the app features that users comment on. The app feature entity belongs to a different perspective of the application, the detailed definition of app features is shown in Table I. Each category contains feature words with specific meaning. For example, ”button” belongs to the topic ”GUI”. In the review KG, the app feature entity indicates comment objects around a specific app.

2) Entity and Relation construction: The relationships of user sentiment and app features were then embedded into the Knowledge Graph. Each relationship is displayed in the KG through the one-to-one correspondence of the concepts including user entity, app feature entity and the sentiment score. The attributes information such as app category and app feature topic are also linked to the app.

In this way, our Knowledge Graph can express the relations of users and the Apps. For each user entity and App feature entity, there exists a relation line with a sentiment score to indicate the meaning and the strength of the relation, such as “Good (0.77510)”, “Bad (-0.47073)”. We use Neo4j[4], an

[4]https://neo4j.com/

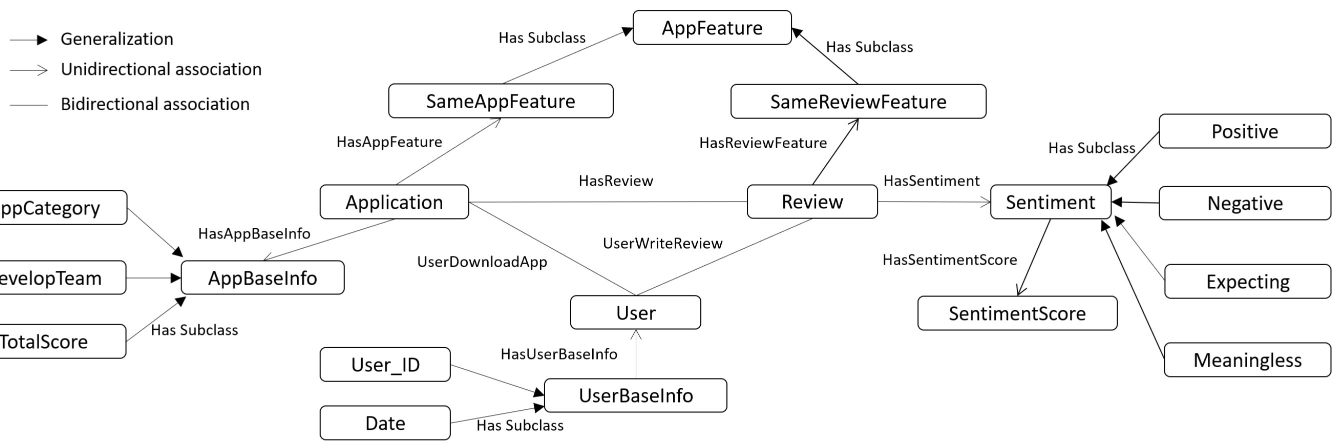


Fig. 2. The Conceptual model of Review Knowledge Graph

online database, to construct the Knowledge Graph of the application's review.

D. Application searching based on Knowledge Graph

Based on the application's knowledge graph, application and its features can be searched accurately. In addition, the distribution of the user's emotions can be shown according to the sentiment score. Several searching problems as follows can be solved efficiently.

1) Suitable Application Retrieval: For a specific application searching, the knowledge graph performs a clustering algorithm to group the applications that have the same attributes, e.g. app category, app development team, app total score, etc.

2) Improvable Application Feature Retrieval: The development team can check the expecting relations in the application knowledge graph to know the most needed functions. In this way, the knowledge graph can achieve the application's multi-angled retrieval.

3) User Sentiment Distribution Retrieval: For selecting a specific function or standard of applications, the knowledge graph can give the results based on app features, which were linked with positive sentiment scores ranging from +1 to -1. It can also show the distribution of users' experience feelings for a certain app by ordering sentiment relations.

IV. Evaluation

A. Experimental Data Set

We utilize crawling tools designed by ourselves to catch the application's description and user reviews from Google Play Store[5]. To increase the accuracy and reliability of review analysis, we only choose the application whose total score is bigger than 4.0/5.0 and the number of reviews is bigger than 1,000. For those categories that take less than 1% of the total apps we only choose the applications with score bigger than 3.5/5.0 and the number of reviews bigger than 100. We get a total of about 4,370 applications and about 4,396,950 pieces of reviews. The proportion of the reviews number from each app category is shown in Fig. 4, in which all app categories and the sentiment distribution of reviews are illustrated. In this picture, each row represents an application category. For each application category, the figure identifies the total number of user comments in that category, the proportion of comments in the total number of experimental comments, and the review sentiment distribution in each category of comments. The proportion of different sentiment is distinguished by different color.

For each review, we save it as one record with review_id, review_app, review_content, review_date, and review_user to indicate the review details. The part of the database we crawled from the website is shared on another website[6].

B. Research Question

This research aims to use previous user comments to provide a more practical result for users. The most important evaluation standard of the results would be the user's satisfaction degree since there are much more fine-grained feature categories than the original searching method. So we have the following research questions to guide our evaluation.

- RQ1: Does this Knowledge Graph improve the algorithm generated searching results?
- RQ2: Does this Knowledge Graph deal with the application information in the same way for all app platforms?

C. Experimental Setting

The app reviews data is collected from the Google Play Store, which is a world-wide used platform for the mobile application's downloading. This app information has also been analyzed for app retrieval before. Since we only choose the app with a high total score and accept conspicuously rich searching terms than ever before, the searching efficiency would improve a lot. Which can explain RQ1. Using the sentiment corpus and app feature dictionary designed before, we manually mark 4000 sentences and achieved an agreement of 92%. If we hold a different opinion on the sentence, we

[5]https://play.google.com/store/apps

[6]https://github.com/fraulifang/Google-app-reviews

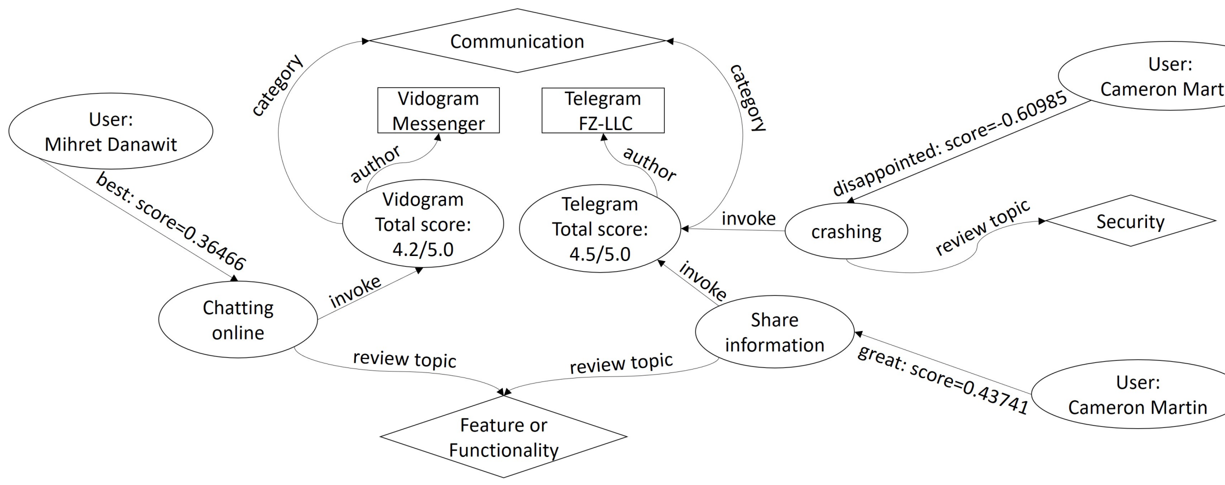


Fig. 3. Example of Knowledge Graph

discussed and then labeled it. We collected 1802 positive relations, 1326 negative relations, 628 improve relations and 244 useless relations. Then we use the Text-CNN [22] model to automatically classify the other reviews. The parameters of the model are: filter size are 3,4 and 5, the number of filters is 100, and three convolution layers depending on three different window sizes. There are also three max-pooling layers, and an output layer with a softmax function to get the label. The performance of our model is presented in Table II.

D. Results and Analysis

For now, we get about 4,395,000 relation pairs of user and app features, including 1,923,038 positive relations, 1,174,434 negative relations, 874,480 improvement relations, and 423,048 meaningless relations. The more detailed distribution of user sentiment is shown in Fig.4. Then we construct the knowledge graph based on the relations and the application description information.

A simplified example of KG is shown in Fig.3, which contains user entity, app feature entity, relation score and entity attribute. In this figure, rectangles represent the Apps from the App Store, ovals represent the App features and rhombus denote the attributes such as App category and Topic category. In this figure, two apps, Telegram and Vidogram, with its total score from Google Play are shown in the graph. The lines with arrows denote the relationship between apps and their entities and users. The label of the solid line in the figure is the type of relationship between the entities.

Based on such a huge graph, application searching can be achieved. Apps have the same attributes that can be easily grouped and the category of the app is also shown clearly in the graph. Furthermore, users can simply search the KG to find the application they want and know the real experience feelings from others. For example, a user wants to find an app for communication. He or she can search the user sentiment distribution of the app and the result would be the app's sequential arrangement of sentiment score, e.g."Telegram-4.5, Vidogram-4.2" in Fig3. Or someone wants to find an application that can read both .doc and .pdf format files, then they can search the app feature with those keywords. What's more, app developers can also check the expecting relations to see whether the application needs to be improved in some directions from the user's aspect. In this way, the multi-angled and user-friendly app searching can be achieved.

As for RQ2, we test our method on other application platforms- Uptodown[7], which contains Android apps in 6 categories, including Communication, Games, Lifestyle, Multimedia, Productivity, and Tools. We only choose the apps with a total score bigger than 4.5/5.0 and the number of reviews is bigger than 1,000. We get a total of about 700 applications. By dealing with the reviews as illustrated before, we get about 680,000 relationships. Then the Knowledge graph is expanded using this new application information. Since the app feature entities and user sentiment information are the same, this method can include more platforms application for more precise searching results.

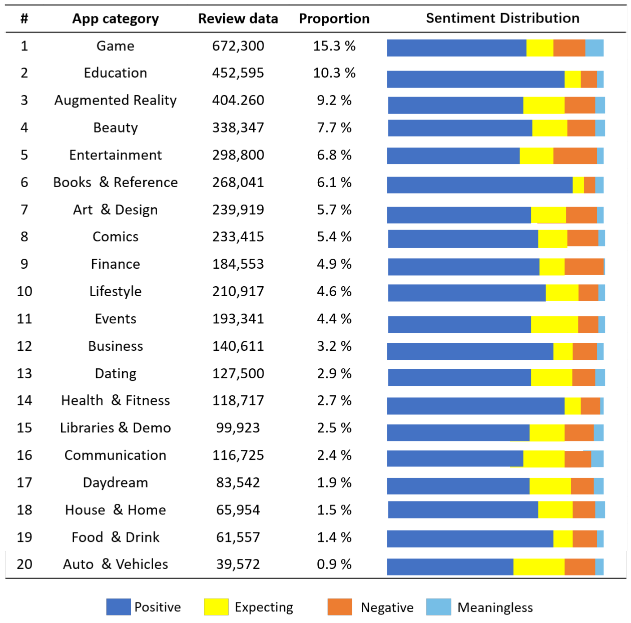
TABLE II
PERFORMANCE OF OUR TEXT-CNN MODEL

category	Precision	Recall	F1 value
positive	0.75	0.71	0.70
negative	0.74	0.72	0.71
improve	0.76	0.76	0.73
useless	0.81	0.85	0.83
total	0.78	0.76	0.78

V. DISCUSSION

There are still many potential directions need to be explored. A meaningful direction is to refine the classification of the review category. For example, dividing the content of comments into version-related, platform-related and operation-related can help eliminate differences and improve the Knowledge Graph.

[7]https://en.uptodown.com/



#	App category	Review data	Proportion	Sentiment Distribution
1	Game	672,300	15.3 %	
2	Education	452,595	10.3 %	
3	Augmented Reality	404.260	9.2 %	
4	Beauty	338,347	7.7 %	
5	Entertainment	298,800	6.8 %	
6	Books & Reference	268,041	6.1 %	
7	Art & Design	239,919	5.7 %	
8	Comics	233,415	5.4 %	
9	Finance	184,553	4.9 %	
10	Lifestyle	210,917	4.6 %	
11	Events	193,341	4.4 %	
12	Business	140,611	3.2 %	
13	Dating	127,500	2.9 %	
14	Health & Fitness	118,717	2.7 %	
15	Libraries & Demo	99,923	2.5 %	
16	Communication	116,725	2.4 %	
17	Daydream	83,542	1.9 %	
18	House & Home	65,954	1.5 %	
19	Food & Drink	61,557	1.4 %	
20	Auto & Vehicles	39,572	0.9 %	

Fig. 4. The proportion of app category and the sentiment distribution

The size of the database is equally important and determines the accuracy and comprehensiveness of the knowledge map. We tend to continue the collection work and generate more knowledge on the data while achieving automatic real-time updates of some data sources. The resulting knowledge contributes to further development.

VI. Conclusion

In this work, we propose an efficient and effective application search approach based on a user review knowledge graph, which contains information of application features and their corresponding user sentiment. To this end, a total of 4,370 applications and 4,396,950 pieces of reviews were collected. We leverage advanced NLP techniques for extracting the app features and identifying their corresponding user's sentiment. The 4,396,000 structured relation pairs are then embedded into a Review Knowledge Graph. Based on such a KG, application users and developers can search specific applications, review the distribution of the user sentiment and application categories to better achieve their specific requirements.

Acknowledgement

This work is supported by National Natural Science of Foundation of China (No.61902010) and Beijing Excellent Talent Funding-Youth Project (No.2018000020124G039).

References

[1] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller, "Checking app behavior against app descriptions," *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*, 2014.

[2] N. Tignor, P. Wang, N. Genes, L. Rogers, S. G. Hershman, E. R. Scott, M. ZWweig, Y.-F. Yvonne Chan, and E. E. SchadtT, "Methods for clustering time series data acquired from mobile health apps," *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, vol. 22, pp. 300–311, 02 2017.

[3] D. L. B. Lulu and T. Kuflik, "Functionality-based clustering using short textual description," *Proceedings of the 2013 international conference on Intelligent user interfaces - IUI '13*, 2013.

[4] S. Krishna, A. Bajaj, M. Rungta, V. Vala, and H. Tiwari, "Relemb: A relevance-based application embedding for mobile app retrieval and categorization," *Computación y Sistemas*, vol. 23, no. 3, 10 2019.

[5] N. Chen, S. C. Hoi, S. Li, and X. Xiao, "Simapp," *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining - WSDM '15*, 2015.

[6] D. H. Park, Y. Fang, M. Liu, and C. Zhai, "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*, 10 2016.

[7] U. Bhandari, K. Sugiyama, A. Datta, and R. Jindal, "Serendipitous recommendation for mobile apps using item-item similarity graph," *Information Retrieval Technology*, pp. 440–451, 2013.

[8] P. D. Turney, "Thumbs up or thumbs down?," *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, 2001.

[9] L. Gatti, M. Guerini, and M. Turchi, "Sentiwords: Deriving a high precision and high coverage lexicon for sentiment analysis," *IEEE Transactions on Affective Computing*, vol. 6, no. 1, pp. 409–421, 2016.

[10] IDC, "Idc - smartphone market share - os." https://www.idc.com/promo/smartphone-market-share/os, 2018.

[11] A. Datta, K. Dutta, S. Kajanan, and N. Pervin, "Mobilewalla: A mobile application search engine," *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol. 18, no. 1, pp. 172–187, 2012.

[12] D. Jiang, J. Vosecky, K. W. T. Leung, and W. Ng, "Panorama," *Proceedings of the 16th International Conference on Extending Database Technology - EDBT '13*, 2013.

[13] A. Datta, S. Kajanan, and N. Pervin, "A mobile app search engine," *Mobile Networks and Applications*, vol. 18, pp. 42–59, 10 2012.

[14] A. A. Al-Subaihin, F. Sarro, S. Black, L. Capra, M. Harman, Y. Jia, and Y. Zhang, "Clustering mobile apps based on mined textual features," *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '16*, 2016.

[15] E. Costa-Montenegro, A. B. Barragáns-Martínez, and M. Rey-López, "Which app? a recommender system of applications in markets: Implementation of the service for monitoring users' interaction," *Expert Systems with Applications*, vol. 39, no. 10, pp. 9367–9375, 08 2012.

[16] K. Zhu, Z. Liu, L. Zhang, and X. Gu, "A mobile application recommendation framework by exploiting personal preference with constraints," *Mobile Information Systems*, vol. 2017, pp. 1–9, 2017.

[17] D. H. Park, M. Liu, C. Zhai, and H. Wang, "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 - SIGIR '15*, 2015.

[18] R. C. Jisha, R. Krishnan, and V. Vikraman, "Mobile applications recommendation based on user ratings and permissions," *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 09 2018.

[19] D. Bae, K. Han, J. Park, and M. Y. Yi, "Apptrends: A graph-based mobile app recommendation system using usage history," *2015 International Conference on Big Data and Smart Computing (BIGCOMP)*, 02 2015.

[20] A. Di Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora, and H. C. Gall, "What would users change in my app? summarizing app reviews for recommending software changes," *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2016*, pp. 499–510, 2016.

[21] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 11 1995.

[22] Y. Kim, "Convolutional neural networks for sentence classification," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.