Intention-Based Destination Recommendation in Navigation Systems

Shuncheng Liu[1], Guanglin Cong[1], Bolong Zheng[2], Yan Zhao[3], Kai Zheng[1], and Han Su[1(✉)]

[1] University of Electronic Science and Technology of China, Chengdu, China
{scliu,glcong,zhengkai,hansu}@uestc.edu.cn
[2] Huazhong University of Science and Technology, Wuhan, China
blzheng@hust.edu.cn
[3] Soochow University, Suzhou, China
yanzhao@suda.edu.cn

Abstract. Taking natural languages as inputs has been widely used in applications. Since the navigation application, which one of the fundamental and most used applications, is usually used during driving, thus the navigation application to take natural language as inputs can reduce the risk of driving. In reality, people use different ways to express their moving intentions. For example, 'I want to go to the bank' and 'I want to cash check' both reveal that the user wants to go to the bank. So we propose a new navigation system to take natural languages as inputs and recommend destinations to users by detecting users' moving intentions. The navigation system firstly utilizes a wealth of check-in data to extract corresponding words, including stuff and actions, for different types of locations. Then the extracted information is used to detect users' moving intentions and recommend suitable destinations. We formalize this task as a problem of constructing a model to classify users' input sentences into location types, and finding proper destinations to recommend. For empirical study, we conduct extensive experiments based on real datasets to evaluate the performance and effectiveness of our navigation system.

Keywords: Navigation system · Destination recommendation · Intention detectection

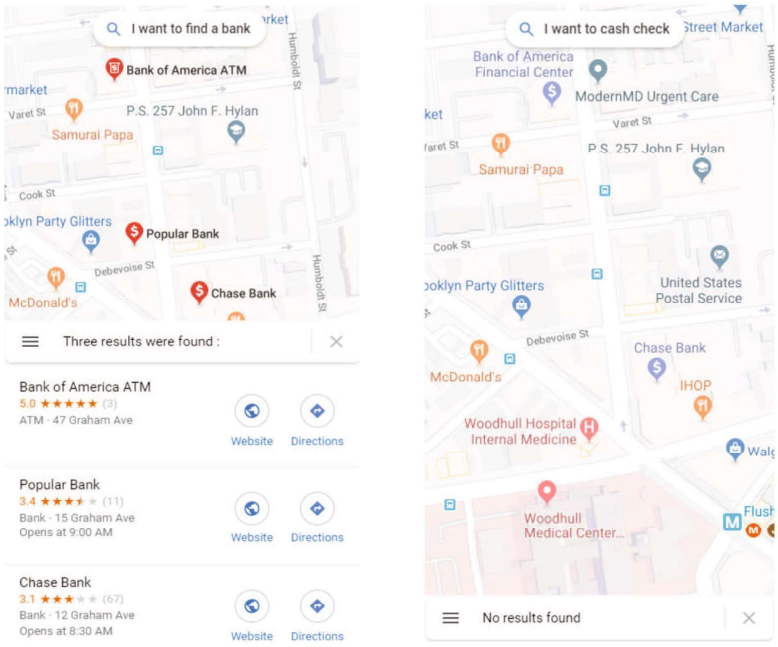
1 Introduction

Navigation applications that find optimal routes in road networks are one of the fundamental and most used applications in a wide variety of domains. Over the past several decades, the problem of computing the optimal path has been extensively studied and many efficient techniques have been developed, while the way of how users interact with navigation applications does not change a lot in the past years. Most navigation applications require users to input a specific source and a specific destination.

However, there is a very obvious trend in the IT industry. Many widely used applications take natural languages as inputs. That is to say, in the past, users need to tap words and click buttons as input, but now users only need to say their needs, and the application can respond accordingly. For example, we can use Siri to make phone calls

Y. Nah et al. (Eds.): DASFAA 2020, LNCS 12114, pp. 698–710, 2020.
https://doi.org/10.1007/978-3-030-59419-0_43

by simply saying 'call Michael'. Navigation applications are widely used during vehicle driving. At this time, it is very dangerous for users to use their hands to type in words on cellphones. So nowadays, a few navigation applications have followed the trend and taken natural language as inputs. As shown in Fig. 1(a), when a user says 'I want to find a bank', these navigation applications will recommend several nearby banks to the user. In this case, the user has specified the destination, i.e., bank. Meanwhile, users may only describe what they will do next, e.g., 'I want to cash check', without specifying the destination. Obviously, if the user wants to cash check, she needs to go to the bank. But current navigation systems fail to realize that the user intends to go to the bank and recommend nearby banks to the user, shown in Fig. 1(b). Meanwhile, with the ever-growing usage of check-in applications, e.g., Twitter and Foursquare, in mobile devices, a plethora of user-generated check-in data became available. This data is particularly useful for producing descriptions of locations. For example, users usually mention that they cash check in banks. Thus by mining the check-in data, we can know that 'cash check' has a strong correlation to 'bank' and the user may intend to go to the bank. So in this paper, we propose an intention-based destination recommendation algorithm, which detects users' moving intentions and recommends appropriate destinations to users, used in navigation systems.



(a) Recommedation Result with a Specific Destination

(b) Recommedation Result without a Specific Destination

Fig. 1. Existing navigation systems

In order to achieve intention-based destination recommendation for the users, there are some challenges we have to overcome. The first one is how to measure the correlations between stuff, actions and locations. The second one is how to detect a user's moving intention and correctly recommend locations to users. To address these challenges, we propose a two-phase framework, i.e., off-line category dictionary constructing and online recommendation. The off-line category dictionary constructing is to train

a classifier model, and use a semantic word extractor to output a category dictionary. With the user's natural sentence t_{input} and her current location loc as input, the online recommendation module detects the moving intention by utilizing the trained classifier. And then the online module recommends a list of destinations to return the user.

In summary, the contributions of this paper are as follows:

- We propose a new navigation system to take natural languages as inputs and recommend destinations to users by detecting users' moving intentions. We formalize the intention-based destination recommendation problem, and then propose efficient algorithms to solve them optimally.
- We propose an intention-based destination recommendation algorithm, which detects users' moving intentions and recommends appropriate destinations to users. To the best of our knowledge, this paper pioneers to recommend destinations without specifying.
- We conduct extensive experiments based on real datasets, and as demonstrated by the results, our navigation system can accurately recommend destinations to users.

The remainder of the paper is structured as follows. Section 2 introduces the preliminary concepts and the work-flow of the proposed system. The intention-based destination recommendation algorithm is described in Sect. 3. The experimental studies are presented in Sect. 4. We review the related work in Sect. 5. Section 6 concludes the paper.

2 Problem Statement

In this section, we introduce preliminary concepts, and formally define the problem. Table 1 shows the major notations used in the rest of the paper.

Table 1. Summarize of notations

Notation	Definition	Notation	Definition
l	A POI in the space	$l.s$	The significance of a POI l
c	A venue category	$\mathbb{L}$	A list of l
t	A check-in data	V	A word vector
λ	A weight of linear combination	$D_{\mathbb{C}}$	The category dictionary
$F_{threshold}$	The filtering threshold	loc	A user's current location
$t_{input}.c$	The predicted category of a t_{input}	$t.c$	The category of a check-in data t
$\mathbb{T}_c$	A set of t with the same category c	t_{input}	A user-entered natural sentence

2.1 Preliminary

Definition 1 (POI). *A point of interest l, or POI, is a specific point location that someone may find useful or interesting.*

Definition 2 (Check-in data). *A Check-in data t is a short text that people announce their arrival at a hotel, airport, hospital, bank, or event.*

Definition 3 (Category). *A category c is a text label e.g.,Bank, Museum, Hotel, which refers to the Venue Category provided by Foursquare.*

In this work, we have nine root categories with one hundred and fifty leaf categories related to Foursquare Venue Categories. $\mathbb{C}$ represents the set of one hundred fifty nine category values obtained by adding nine root categories and one hundred fifty leaf categories. The root category is denoted by $c(R)$ and the leaf category is denoted by $c(L)$. e.g., $c(R)$ is Shop or Service then $c(L)$ is Bank. A POI l belongs to one or more categories, so we use $l.C$ to indicate all categories to which the POI l belongs. Since a t is a text that occurs at a specific POI, it has category $t.c$ corresponding to POI l.

Definition 4 (Category Dictionary). *A category dictionary D_c is a dictionary containing all words and corresponding to a category c.*

A D_c refers to a category and the corresponding dictionary content. Each word has a weight indicating its relevance to the category. e.g.,$D_{c=Bank}$ is money(0.74), deposit(0.63), withdraw(0.61) and so on. We use $D_{\mathbb{C}}$ to represent all dictionary contents.

Definition 5 (User Input). *A user input t_{input} is a natural sentence, which indicating a user's moving intention.*

In navigation applications, the input may or may not specify the destination. Previously, the task of natural sentence recognition including destination words has been well studied, so this article focuses on natural sentences t_{input} that do not contain destination words. Since we need to predict natural sentences that do not contain the destination vocabulary, we use $t_{input}.c$ to represent the category of text predicted by the system. $t_{input}.c$ here is formally consistent with c above.

From an overall perspective, we use check-in data t to achieve category dictionary $D_{\mathbb{C}}$, and then use the results of $D_{\mathbb{C}}$ to recommend a list of destinations $\mathbb{L}$ related to a user when the user enters t_{input} and location loc.

2.2 System Overview

In this section, we present the workflow of the recommendation system. Figure 2 presents the overview of the proposed system, which basically consists of four parts:

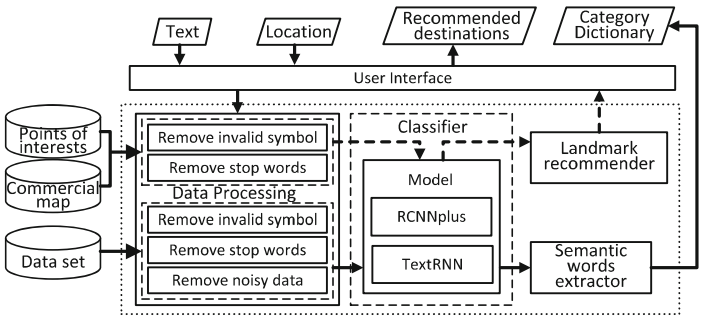


Fig. 2. System overview

data processing, classifier, landmark recommender and semantic words extractor. The system has two tasks.

The first task is an off-line task for constructing the category dictionary. The input of the whole task is the foursquare data set. First, we remove the illegal symbol, stop words and the noise data from the all check-in data t. Then we input the cleaned data into the classifier, tune and train the model, and save the best model parameters, the model outputs $t.c$ for each t. Next the semantic words extractor recognizes the correct predictions (t,$t.c$), then aggregate them as $\mathbb{T}_c$ to extract keywords, finally generate the dictionary $D_{\mathbb{C}}$ as the output of the task.

The second task is an online task to recommend destinations. When the user enters t_{input} and loc, firstly the system has removed invalid symbols and stop words from the sentence t_{input}. Next, entering the cleaned t_{input} into the model trained by the off-line task. Then the model outputs the category of the natural sentence $t_{input}.c$. After that, the landmark recommender recommends the destinations $\mathbb{L}$ based on loc and $t_{input}.c$.

3 Destination Recommendation

In this section, we present our algorithm to explain how to extract user's moving intention based on natural sentences, including data clean algorithm, RCNN-plus classifier, and destination recommender. The user inputs a natural sentence t_{input} and the current location loc, then the system uses the above algorithms to output a recommended destination list $\mathbb{L}$. We will introduce these algorithms in the following subsections.

3.1 Data Clean Algorithm

With the ever-growing usage of check-in applications, e.g., Foursquare, a plethora of user-generated check-in data became available. The navigation system firstly utilizes a wealth of check-in data to extract corresponding words, including stuff and actions, for different types of locations. Textual content is the most frequently-used feature of check-in data, since users may mention semantic words while posting messages. However comment data may contain a lot of irrelevant characters including symbols, emotions, and URLs. Therefore, we need to do data preprocessing. We first remove irrelevant characters and the stop words. But we found that the check-in dataset contains a lot of noisy, which leads to a low accuracy classifying model. In other words, the t is ambiguous and irrelevant to it's $t.c$. Therefore, we propose a new data cleaning algorithm, we simply use TextRNN [6] model and word vectors to recognize noisy text. Firstly, we use original check-in data to train some TextRNN model classifiers, we use these rough classifiers to classify check-in data t into root categories. For each classifier, we filter out all the check-in data t that the TextRNN model predicts correctly, and prepare to use the Stanford pre-trained word vectors 'Glove' for data cleaning. The vectors contain 2.2M words and each word can transfer to a 300-dimensional word vector V.

After we trained TextRNN classifiers, for each classifier we select the check-in data t that predicted correctly. Then we calculate the text vector distance $Dis(V_t, V_c)$ based on Equation (1). In detail, the V_t represents the average of all word vectors contained in the t, similarly, the V_c represents the average of all word vectors contained in the $t.c$.

After that, we calculate the filtering threshold for each root category c using Equation (2). Each $F_{threshold}(c)$ represents how much noise the category c can tolerate. Next, we use $F_{threshold}(c)$ to clean all the original check-in data. According to Equation (1), we calculate the text vector distance between each t and $t.c$. When the distance value is greater than corresponding $F_{threshold}(c)$, the check-in data t should be deleted.

In summary, the algorithm filtered two parts of check-in data. The first part is misclassified check-in data, and the second part is check-in data that has a low correlation with its category. Therefore, the retained check-in data is of high quality.

$$Dis(V_t, V_c) = \|V_t - V_c\|_2 \tag{1}$$

$$F_{threshold}(c) = \max_{\forall t.c \in c} Dis(V_t, V_c) \tag{2}$$

3.2 TextRNN-RCNN Model

The RCNN [5] model is a text classifier with recurrent and convolutional neural network structure. It has a good effect on extracting key components in the text. The TextRNN [6] is a text classifier with recurrent structure. It has a good effect on obtaining context information. In order to leverage the power of both RCNN and TextRNN, we improve them and propose a joint model named *TextRNN-RCNN*.

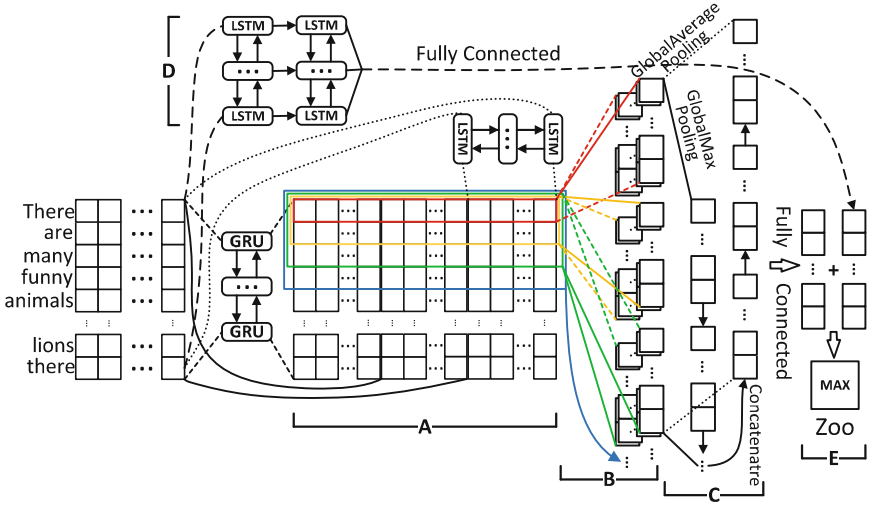


Fig. 3. TextRNN-RCNN model

For RCNN, it uses a bi-directional recurrent structure to capture the contextual information, and use the convolutional neural network to capture the semantic information of texts. [5] adds a max-pooling layer to capture the key component in the texts which can automatically judge the key features in text classification. Since the RCNN only uses one single max-pooling layer, under the 'without destination words' scenario the RCNN model is not comprehensive enough in feature selection. Besides, the capability of the model to capture semantic information features and the key components in the texts is insufficient. In order to improve the RCNN model, we modify the network structure and proposed a new model named RCNN-plus.

As Fig. 3 shown, firstly we delete the left context and right context structure to simplify the model. Then we modify the RCNN recurrent structure, we use the bidirectional GRU and bidirectional LSTM simultaneously as the bi-directional recurrent structure to capture the contextual information as much as possible. We concatenate the output of GRU, embedding layer and LSTM, shown in part A of Fig. 3. Besides, for convolutional Neural Network, we replace a single convolution kernel structure with a multi-convolution structure in RCNN-plus model, and use different convolution kernels to represent different word selection window widths, as shown in part B of Fig. 3. Convolution kernels of different colors represent windows of different lengths. The amount of convolution kernels depends on different text classification tasks. In order to capture key components in the texts with different sizes, shown in part C of Fig. 3, we perform maximum pooling and average pooling operations on all convolution kernels' results, while the RCNN model only uses a max-pooling layer to capture the key component. After that, we concatenate and input pooling results to the fully connected layer.

Since the above structure pays more attention to the key components in the texts essentially, we use the TextRNN to better obtain context information. As shown in part D of Fig. 2, we add a TextRNN to improve the generalization ability of the model, since it has a two-layer bidirectional LSTM structure. Then we use a threshold λ to linear combine two outputs and weigh them shown in part E of Fig. 2. Finally, we choose the maximum probability as the prediction.

3.3 Select Significant Destinations

In our application scenario, when the user provides a complete text, the model can calculate the corresponding POI category. Then the category needs to be replaced with certain POIs (e.g., Bank is replaced with Bank of America Financial Center). Therefore, we need to recommend proper destinations based on the selected category and the user's current location. It is a common sense that destinations have different significances. For instance, the White House is famous globally, but Pennsylvania Ave, where the White House is located, is only known to the locals in Washington DC. The selected destinations should have a high significance, so that users can find it easily. By regarding the users as authorities, destinations as hubs, and check-ins/visits as hyperlinks, we can leverage a HITS-like algorithm [16] to infer the significance of a destination $l.s$. The recommended destinations list $\mathbb{L}$ contains all POIs which belong to the certain category within a distance to the user's current location and are sorted by significance $l.s$.

4 Experiment

In this section, our experimental results are presented to evaluate the performance of model. This model is performed using Java and Python on Ubuntu 16.04. All the experiments are conducted on a computer with Intel Core i7-4770K(3.9GHz) CPU and 16GB memory.

4.1 Experimental Setup

Check-in Dataset: We use two check-in datasets (New York and Los Angeles) from Foursquare to carry out the experiments. The New York dataset contains 1,711,646 check-in texts, while the Los Angeles dataset contains 1,921,214 check-in texts.
POI Dataset: We have two POI databases. There are approximately 300 thousand POIs in New York and 500 thousand in Los Angeles.
Category Dataset: We have a category dataset. It contains nine root categories and 900 leaf categories. Among the 900 leaf categories, 150 leaf categories are tagged in a high frequency while the rest 750 leaf categories are tagged in a low frequency.
Baseline Methods: We compare our proposed TextRNN-RCNN method and text selecting method with the following algorithms:

- CNN-sentence [3] is a simple CNN with one layer of convolution on top of word vectors obtained from an unsupervised neural language model.
- RCNN [5] judges which words play key roles in text classification. It captures contextual information in texts with the recurrent structure and constructs the representation of text using a convolution neural network.
- TextRNN [6] assigns a separate LSTM layer for each task. Meanwhile, it introduces a bidirectional LSTM layer to capture the shared information for all tasks.
- TextAttBiRNN [10] is a simplified model of attention that is applicable to feed-forward neural networks. The model demonstrates that it can solve the synthetic 'addition' and 'multiplication' long-term memory problems for sequence lengths which are both longer and more widely varying than the best-published results for these tasks.

Parameters Setting: The penalty for TextRNN-RCNN λ is set to 0.5 in our method and the number of RCNN-plus kernels is set to 5. In the following experiments, parameters are set to default values if unspecified.

4.2 Case Study

Before assessing the performance of the system, we perform a case study to show the effectiveness of intention-base destination recommendation. Figure 4.2 presents a case study where we input a sentence and get a POI recommendation. In this case, a user said 'I will deposit money' and the navigation system took the sentence as the input. Then the system removed invalid symbols and stop words of the sentence and got words 'deposit money'. After that, the system utilized the classifier to identify which category the words belonged to, i.e., category 'bank'. At last, the system recommended nearby POIs, which belongs to the category 'bank', to the users (Fig. 4).

Besides, we perform another case study to show the effectiveness of semantic words extraction. After training the check-in data, we can get a category dictionary that contains all words, including stuff and actions, corresponding to each category. Table 2 shows some categories, e.g.,'bank', 'zoo', 'bookstore', 'bar' and 'Asian restaurant', and their corresponding semantic words. For instance, for category 'bank', words such as 'deposit', 'money', 'credit', 'debit', 'withdraw'and 'ATM' have strong correlation to 'bank'.

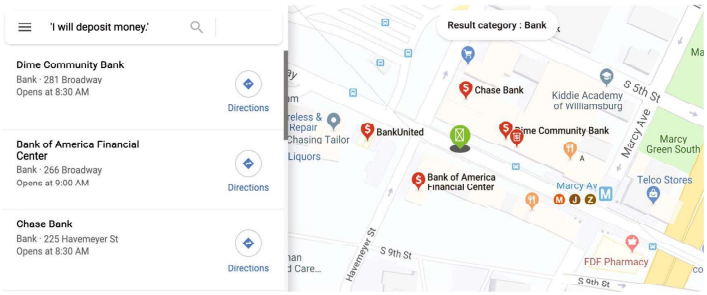


Fig. 4. An example of intention-based destination recommendation navigation system

Table 2. Examples of the category dictionary

Category	Semantic words
Bank	Deposit,money,credit,debit,withdraw,ATM
Zoo	Animal,lion,monkey,feed,baboon,aquarium
Bookstore	Book,read,magazine,comic,section,literature
Bar	Cocktail,wine,whiskey,karaoke,beer,bartender
Asian restaurant	Noodle,dumpling,tofu,sushi,rice,tea

4.3 Performance Evaluation

In this subsection, we test the methods proposed previously by collecting data from foursquare. To make the evaluation purpose, we train different models to find categories of check-in data and user inputs t_{input}, and evaluate their performance by their accuracies. Besides, we do ablation experiments about λ to evaluate RCNN-plus and TextRNN. At last, we do another ablation experiments to evaluate the data-filtering method.

Model Accuracy. As discussed above, we train 80% of check-in data to generate a category classifier. After training, this classifier takes a check-in data t as its input, and then outputs the corresponding check-in category $t.c$ of t. We use the rest 20% of check-in data to test the effectiveness of the model. A good model should have a high accuracy of classifying test check-ins into correct check-in categories. So in this set of experiments, we calculate the accuracies of our algorithm (TextRNN-RCNN) with comparison methods (CNN-sentence, RCNN, TextRNN, TextAttBiRNN, RCNN-plus). Figure 5(a)–5(b) shows the accuracies of the root classifier and the accuracies of the leaf classifier. As shown in figures, TextRNN-RCNN outperforms all the comparison methods in both the root classifier and the leaf classifier, which demonstrates the superiority of our approach. Two TextRNN related methods, i.e., TextRNN and TextRNN-RCNN, have higher accuracies than others. It indicates that this model is more suitable in this application scenario since it is sensitive to obtain context information.

Effect of λ: For the joint model TextRNN+RCNN, RCNN-plus model pays more attention to key components in the texts essentially, and TextRNN has a good effect on obtaining context information. We use λ to linearly combine model RCNN-plus

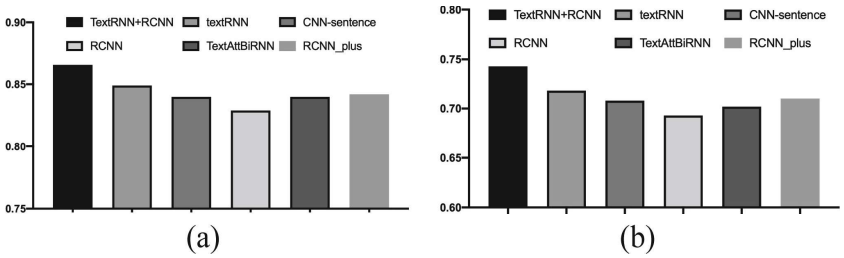


Fig. 5. (a)Accuracies of root classifier; (b) Accuracies of leaf classifier

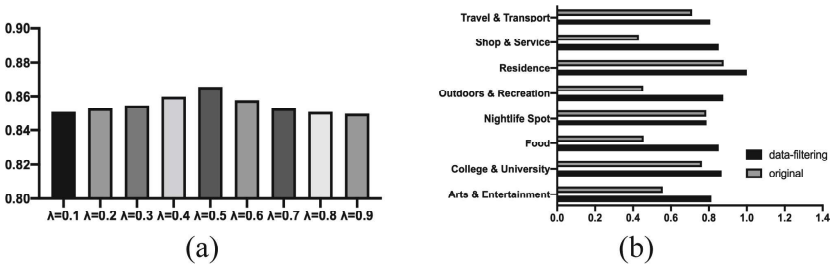


Fig. 6. (a)Accuracies of different λ; (b) Effectiveness of data filtering

and model TextRNN, and a bigger λ indicates a higher weight of TextRNN in the joint model. In this set of experiments, we tune λ from 0.1 to 0.9 with a step of 0.1. Figure 6(a) shows accuracies of different λ of the joint model. As shown in the figure, with the increase of λ, the accuracy of the joint model initially rises and subsequently falls. And the highest accuracy is obtained when λ is 0.5.

Effectiveness of Different Amounts of Kernels: RCNN-plus plays an important role in the joint model. The amount of the convolution kernels may affect the effectiveness of RCNN-plus, since the amount of the convolution kernel determines the amount of the sliding window. Specifically, the number of convolution kernels represents the incremental convolution kernel size. For example, kernel=3 means that the convolution kernel size from 1 to 3 are used simultaneously. In this set of experiments, we compare the accuracies of different amounts of kernels on root categories. As shown in Table 3, when kernel equals to 5, the RCNN-plus outperforms others in most cases. This is because with the rising of the kernel amounts, more continuous features can be extracted; while too many kernels are utilized, it may cause overfitting.

Effectiveness of Data Cleaning. We proposed a data clean algorithm in Sect. 3.1, that all experiments are conducted on cleaned data. So in this set of experiments, we evaluate the effectiveness of our data clean algorithm. We conduct a training process on both raw check-in data and cleaned check-in data and compare the accuracies of these two classifiers. Figure 6(b) shows the accuracies of leaf classifier. In this experiment, we can find that all the accuracies of cleaned data are higher than those of raw data. It indicates that our data clean algorithm is effective for this system.

Table 3. Accuracy of different amount of kernel

Root category	kernel = 1	kernel = 3	kernel = 5	kernel = 7	kernel = 9
Arts & Entertainment	0.80463	0.80435	**0.80578**	0.7988	0.80162
College & University	0.86475	0.86438	**0.86584**	0.86548	0.85563
Food	0.82289	0.8201	**0.8239**	0.82118	0.81852
Nightlife Spot	0.78391	**0.78785**	0.78023	0.77881	0.77361
Outdoors & Recreation	0.84325	**0.85297**	0.85158	0.84964	0.8527
Residence	0.99371	0.98023	**0.9986**	0.89937	0.89937
Shop & Service	0.83611	0.83884	**0.83666**	0.83884	0.83666
Travel & Transport	0.78736	0.77797	**0.78968**	0.7811	0.77594
Accuracy	0.83984	0.84074	**0.84121**	0.84	0.84048
Fine-grained	0.8464775	0.845385	**0.8484625**	0.8343025	0.83161375

5 Related Work

The problem referred to in this work is relevant to text processing and POI querying issues, including text classification, semantic location label mining, location recommendation and so on.

Text Classification. Given a set of texts, [5] uses recurrent structure to captures contextual information and uses convolutional neural network to constructs the representation of text. [6] proposes three multi-task architectures for RNN which learns to map the arbitrary text into semantic vector representations with both task-specific and shared layers. [6] uses a simple convolutional neural networks (CNN) structure trained on top of pre-trained word vectors for sentence-level classification tasks. On the contrary, in our work, an improved TextRNN-RCNN model is used for text classification tasks to improve accuracy.

Semantic Location Label Mining. Semantic place labeling is the process of defining a location with a category. Place labels have been proposed for automatically updating a person's status on social networking sites, such as [7], and automatically annotating check-ins, such as [15]. [4] develop a classifier that identifies place labels are based on the timing of visits to that place, nearby businesses, and simple demographics of the user. Several researchers use [9,13,14] nearby points of interests to characterize places. In addition, our focus is on mining semantic words corresponding to location label and we mine geographic categories from text features instead of using users' spatiotemporal features.

Location Recommendation. In different application scenarios, the system needs to recommend a series of locations according to the user's location and query category. [16] proposed a HITS-based model to infer users' travel experiences and the relative interest of a location. [2,12] build a recommender system by using multiple users' real-world location histories to recommend geographic locations. For users who need real-time recommendations, [1,8,11] typically recommend locations based on a user's real-time location.

6 Conclusions

In this paper, we have proposed a new navigation system based on check-in data, which can take natural language as inputs and recommend appropriate destinations to users. We propose a two-phase framework, i.e., off-line category dictionary constructing and online recommendation. The off-line category dictionary constructing is to train the classifier model, and use a semantic word extractor to output a category dictionary. The online recommendation module detects the moving intention by utilizing the trained classifier. And then, the online module recommends a list of destinations to return the user. We have conducted extensive experiments on real datasets. As demonstrated by the experimental results, in most cases, our navigation system can accurately recommend destinations to users.

Acknodgements. This work is partially supported by Natural Science Foundation of China (No. 61802054, 61972069, 61836007, 61832017, 61532018, 61902134), Alibaba Innovation Research (AIR) and the Fundamental Research Funds for the Central Universities (HUST: Grants No. 2019kfyXKJC021, 2019kfyXJJS091)

References

1. Abowd, G.D., Atkeson, C.G., Hong, J., Long, S., Kooper, R., Pinkerton, M.: Cyberguide: a mobile context-aware tour guide. Wirel. Netw. **3**(5), 421–433 (1997)
2. Horozov, T., Narasimhan, N., Vasudevan, V.: Using location for personalized poi recommendations in mobile environments. In: International Symposium on Applications and the Internet (SAINT 2006), p. 6-pp. IEEE (2006)
3. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
4. Krumm, J., Rouhana, D.: Placer: semantic place labels from diary data. In: Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, pp. 163–172. ACM (2013)
5. Lai, S., Xu, L., Liu, K., Zhao, J.: Recurrent convolutional neural networks for text classification. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
6. Liu, P., Qiu, X., Huang, X.: Recurrent neural network for text classification with multi-task learning. arXiv preprint arXiv:1605.05101 (2016)
7. Miluzzo, E., Lane, N.D., Eisenman, S.B., Campbell, A.T.: CenceMe – injecting sensing presence into social networking applications. In: Kortuem, G., Finney, J., Lea, R., Sundramoorthy, V. (eds.) EuroSSC 2007. LNCS, vol. 4793, pp. 1–28. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75696-5_1
8. Park, M.-H., Hong, J.-H., Cho, S.-B.: Location-based recommendation system using Bayesian user's preference model in mobile devices. In: Indulska, J., Ma, J., Yang, L.T., Ungerer, T., Cao, J. (eds.) UIC 2007. LNCS, vol. 4611, pp. 1130–1139. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73549-6_110
9. Phithakkitnukoon, S., Horanont, T., Di Lorenzo, G., Shibasaki, R., Ratti, C.: Activity-aware map: identifying human daily activity pattern using mobile phone data. In: Salah, A.A., Gevers, T., Sebe, N., Vinciarelli, A. (eds.) HBU 2010. LNCS, vol. 6219, pp. 14–25. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14715-9_3
10. Raffel, C., Ellis, D.P.: Feed-forward networks with attention can solve some long-term memory problems. arXiv preprint arXiv:1512.08756 (2015)

11. Simon, R., Fröhlich, P.: A mobile application framework for the geospatial web. In: Proceedings of the 16th International Conference on World Wide Web, pp. 381–390. ACM (2007)
12. Takeuchi, Y., Sugimoto, M.: CityVoyager: an outdoor recommendation system based on user location history. In: Ma, J., Jin, H., Yang, L.T., Tsai, J.J.-P. (eds.) UIC 2006. LNCS, vol. 4159, pp. 625–636. Springer, Heidelberg (2006). https://doi.org/10.1007/11833529_64
13. Wolf, J., Guensler, R., Bachman, W.: Elimination of the travel diary: an experiment to derive trip purpose from GPS travel data. In: Transportation Research Board 80th Annual Meeting, pp. 7–11 (2001)
14. Xie, R., Luo, J., Yue, Y., Li, Q., Zou, X.: Pattern mining, semantic label identification and movement prediction using mobile phone data. In: Zhou, S., Zhang, S., Karypis, G. (eds.) ADMA 2012. LNCS (LNAI), vol. 7713, pp. 419–430. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35527-1_35
15. Ye, M., Shou, D., Lee, W.-C., Yin, P., Janowicz, K.: On the semantic annotation of places in location-based social networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 520–528. ACM (2011)
16. Zheng, Y., Zhang, L., Xie, X., Ma, W.-Y.: Mining interesting locations and travel sequences from GPS trajectories. In: World Wide Web, pp. 791–800. ACM (2009)