2018 IEEE International Conference on Data Mining Workshops (ICDMW)

TCENR: A Hybrid Neural Recommender For Location Based Social Networks

Omer Tal
Department of Physics and Computer Science
Wilfrid Laurier University
Waterloo, Ontario, Canada
talx6630@mylaurier.ca

Yang Liu
Department of Physics and Computer Science
Wilfrid Laurier University
Waterloo, Ontario, Canada
yangliu@wlu.ca

***Abstract*—Point-Of-Interests (POI) recommendation, an important application of location-based social networks (LSBN), has been extensively researched in recent years. This sub-field of recommender systems (RS) poses unique challenges due to high data sparsity and its relative complexity. An emerging technique is the use of deep neural networks to improve the performance of collaborative filtering (CF) based models. Recent works have successfully integrated such networks with external data, such as social networks, locations, categories and written reviews. In this paper, we propose a new method, Textual and Contextual Embedding-based Neural Recommender (TCENR). The suggested algorithm combines two types of neural networks to model the user-POI interactions based on implicit ratings, social networks, geographical locations and natural language reviews. Experiments on the Yelp dataset show that the proposed model is able to learn the complex interaction and enables improved recommendation performance.**

***Index Terms*—Recommender systems, Neural networks, Multilayer perceptrons, Social network services, Natural language processing**

I. Introduction

With increasing volumes in web-based transactions and activities, recommender systems have become an industry standard for providing users with personalized suggestions. They consist of proposing relevant items, such as movies, songs, products and more, to potential customers.

POI recommendation, a specific case of recommender systems, is usually incorporated into location-based social networks, applications and websites such as Yelp and Foursquare, where users share experiences and contents regarding visited locations. Supplying LBSN users with relevant suggestions allows them to make informed decisions towards the places they are about to visit. With 72 million mobile users per month, as reported by Yelp in June 2018[1], the amount and variety of available data presents a promising environment for the development of such recommender systems.

Probably the most common approach for recommendation is collaborative filtering, a method that consists of modeling users' preferences over items based on past interactions. There are, however, some challenges presented when developing POI recommender systems, making classic CF methods less desirable. First, LSBNs usually include only implicit feedback such as check-ins indication by a user [5]. This results in no negative instances available for the learning process. The second challenge is data sparsity, an issue common to all recommender systems [2], [6], that exists to a higher degree in POI recommenders. Third, there are many factors effecting the user's decision process when choosing a POI, such as her geographical location, preferred category, where the user's friends are and more [4]. These challenges proved the classic CF methods to be insufficient and prone to over-fit the data as reported by previous works [1], [2].

A common approach to address sparsity while acknowledging the user's deriving factors is by utilizing contextual data, such as social networks [1], [4], [10], geographical locations [1], [5], POI category [4] and textual reviews [3], [6]–[9]. This allows the system to learn the interactions between users and locations by taking the users' profiles into account.

A recent development in the field of recommender systems is the use of deep neural networks in place of the standard CF approach. Previous studies [1]–[3], [5], [6] have shown the potential of using such methods in the learning of complex user-item interactions, especially in sparse environments. Different neural network architectures have been introduced to improve the recommendation performance, such as multi-layer perceptrons (MLP) [1], [2], [4], [11], convolutional neural networks (CNN) [3], [8] and recurrent neural networks (RNN) [5], [6], [9]. These networks were shown to highly benefit from the addition of contextual attributes [1], [3]–[5]. As more data becomes available, incorporating these features presents a promising opportunity to improve the personalized POI recommendation process, as will be demonstrated in this paper.

While many existing solutions adopt a single type of neural network that suits best the available data source, in this work we intend to explore multiple network frameworks, i.e., MLP and CNN, to provide POI recommendation with various types of input. To make use of available data and to tackle the sparsity issue, the users' social and geographical context will be employed along with textual reviews and implicit interactions. By training multiple networks together, the suggested model can learn different aspects of the same interaction in conjunction, and therefore to capture the underlying factors in the user's selection. To the best of our knowledge, no work has been done in jointly training an MLP and CNN for the task

[1]https://www.yelp.ca/factsheet

2375-9259/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDMW.2018.00170

of POI recommendation using social networks, geographical locations and natural language reviews as inputs.

Although the proposed solution has been developed to provide recommendations for specific types of inputs, we claim it can be easily generalized to a framework able to support additional features. The proposed model had been empirically evaluated over the Yelp dataset and found to outperform the baselines in terms of accuracy, MSE and precision.

II. BACKGROUND

Incorporating additional data about users and items is a common method to provide meaningful recommendations and to mitigate the cold-start problem. In the area of POI recommender systems, where the data is highly sparse, such practices are essential.

A. Contextual-based Recommender Systems

Location-Based Social Networks are usually rich with contextual input which presents various opportunities for data enrichment in RS. Such features can include time [5], spatial location [13], user's social network [10], item's meta-data [4], demographics [11] and more.

Contextual data is usually incorporated into RS either as part of the input or as a regularizing factor. [11] exploits the strengths of MLP based networks in modeling complex relationships by concatenating multiple feature embeddings to the input before feeding it to a series of nonlinear layers. The final layer's output is then a representation of a user-item interaction, adjusted to the given context. The use of spatial data is demonstrated in [13], where the input space is divided into geographical regions before being incorporated into the model. Instead of learning a single embedding for a given user, a representation is sampled based on her current region and whether she's close to home or traveling.

In case of tasks in highly sparse environments, such as POI recommendation, adding user or item specific inputs may diminish the model's ability to generalize. However, applying the same data as a regulating factor can enhance the model's performance and reduce over-fitting. Such has been done in [10], where the similarity between connected users in the social network was used to constrain a Matrix Factorization model. [1] utilized social networks and geographical distances to enforce similar embeddings for users and locations, thus improving the model's ability to generalize for users and locations with few historical records.

B. Textual-based Recommendation

Since many websites encourage users to provide a written explanation to their numeric ratings, textual reviews are one of the most popular types of data to be integrated into RS. Previous works had adopted probabilistic-based approaches to alleviate the data sparsity problem using textual input [6], [7]. By expressing each review as a bag of words, LDA-based models are able to extract topics which can be used to represent users' interests and locations' characteristics [14]–[16]. These probabilistic methods are usually successful in handling issues that standard CF approaches struggle with, such as out-of-town recommendations where similar users lack sufficient historical data. However, as demonstrated in recent works [3], [6], failing to preserve the original order of words and ignoring their semantic meaning prevents the successful modeling of a given review. On the other hand, an emerging trend of adopting neural networks over reviews allows such learning without the loss of data. These implementations can generally be categorized into RNN-based [6], [9] and CNN-based [3], [8] models.

RNN based Recommender Systems usually rely on the sequential structure of sentences to learn their meaning. In [9], the words describing a target item are fed to a bi-directional RNN layer. Following the Gated Recurrent Unit architecture, the model utilizes an accumulated context from successive words to provide better representation of each word. To preserve and update the context for every word, the recurrent model has to manage a large number of parameters, an issue that is worsened when adopting the popular Long-Short Term Memory paradigm as done in [6].

Following its success in the field of computer vision, CNN-based models are gaining popularity in other areas, such as textual modeling. By employing a sliding window over a given document, such networks are able to represent different features found within the text by identifying relevant subsets of words. [8] follow the standard CNN structure, comprised of an embedding to represent the semantic meaning of each word, convolution layer for generating local features and a max pooling operation to identify the most relevant factors. In [3], two CNNs are developed to represent the target user and item based on their reviews. The resulting vectors are regarded as the user and item representations, which are then fed to a nonlinear layer to learn their corresponding rating.

We claim that by jointly learning contextual and textual based deep models our proposed method will better exploit the strengths of collaborative filtering, while being more resilient to its shortcomings in sparse scenarios. This will be achieved by learning users' and locations' representations as similarities in direct interactions along with the correlation in underlying features extracted from their written reviews.

III. THE NEURAL RECOMMENDER

A. Neural Network Architecture

The following recommender system aims to improve the POI recommendation task by learning user-location interactions using two parallel neural networks, as shown in Fig. 1. The context-based network, presented in the left part of the figure, is designed to model the user-POI preferences using social and geographical attributes and based on a multi-layer perceptron structure [1], [2]. Shown in the right side of the drawing, the convolutional neural network is responsible for the textual modeling unit [3]. It attempts to learn the same preference by analyzing the underlying meaning in users' and locations' reviews. Each of the two networks is based on modeling the user / POI input individually with regard to their shared interaction, defined in the merge layers.

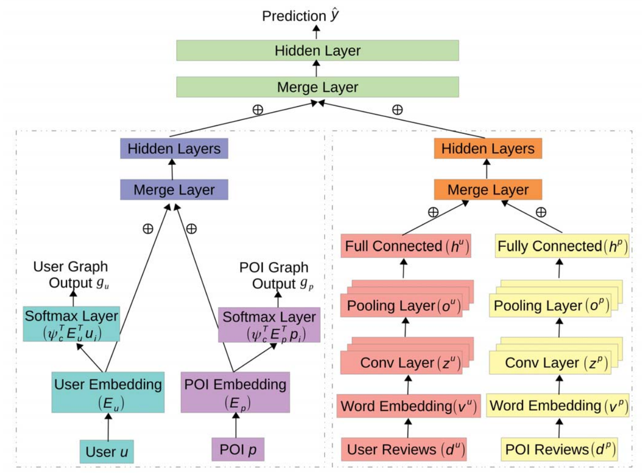

Fig. 1. TCENR Framework

1) Contextual based network layers: To best capture the complex relations between users and locations in LSBN, we chose to adopt the multi-layer perceptron architecture. By stacking multiple layers of nonlinearities, MLP is capable of learning relevant latent factors of its inputs. It is first fed with user and location vectors of sizes *N* and *M*, where each input tuple $< u_i, p_i >$ is transformed into sparse one-hot encoding representations. The two fully connected embedding layers, found on top of the input layer, project the sparse representations of users and locations into smaller and denser vectors. For u_i and p_i, the respective embedding matrices are $E_u \in \mathbb{R}^{k_u \times N}$ and $E_p \in \mathbb{R}^{k_p \times M}$, where k_u and k_p are the corresponding dimensions.

We exploit the users' social networks along with the locations' geographical graphs to constrain the learned embeddings. Two softmax layers take the representations $E_u^T u_i$ and $E_p^T p_i$ as input and transform them back to *N* and *M* sized vectors, respectively. The user output layer, $\psi_c^T E_u^T u_i \in \mathbb{R}^N$ can be formally described as:

$$\psi_c^T E_u^T u_i = a(W_c^u \times E_u^T u_i + b_c^u), \tag{1}$$

where W_c^u and b_c^u are the layer's weight matrix and bias vector, and *a* is a non-linear activation function. Due to the similarity between the user and POI specific layers, the location output layer, $\psi_c^T E_p^T p_i$, will not be developed in this section. Enforcing $\psi_c^T E_u^T u_i$ and $\psi_c^T E_p^T p_i$ to resemble user *u*'s social graph, g_u, and location *p*'s spatial graph, g_p, results in a smoothing factor that limits the amount by which connected entities' embeddings differ.

The user and location embedding are projected to a merge layer and combined by concatenation. Using concatenation instead of dot-product allows varied embedding representations, which in turn improves the generated representations [2]. As the input layer for the following neural network, the merge layer can be represented as $h^{(0)}(x)$ where:

$$x = E_u^T u_i \oplus E_p^T p_i. \tag{2}$$

Since simple concatenation of the user-location embedded vectors does not allow for interactions to be modeled, hidden layers are added to learn these connections. The popular Rectified Linear Units (ReLU) is employed as the activation function for these layers. More formally, the *q*-th hidden layer can be defined as:

$$h_{context}^q(x) = ReLU(W^q h_{context}^{q-1}(x) + b^q), \tag{3}$$

where W^q and b^q are the *q*-th layer parameters.

2) Textual modeling network layers: To improve the model's coverage, a textual-based network is introduced. It simultaneously learns the same interaction as the contextual-based network, but with natural language input. Two additional vectors $d_{1:n}^u$ and $d_{1:n}^p$, representing user *u* and location *p* textual reviews, respectively, are applied as input for this network. Each vector is comprised of all *n* words wrote in user or location reviews merged together, kept in the same order as in the original reviews. These words are then mapped to *c*-dimensional vectors defining their semantic meaning in the following embedding layers. The output of the user embedding layer is the representation of all words used by a user *u* in the form of a matrix, and can be denoted as:

$$V_{1:n}^u = \phi(d_1^u) \oplus \phi(d_2^u) \oplus ... \oplus \phi(d_n^u), \tag{4}$$

where $\oplus$ is the concatenation operator and $\phi : D \rightarrow \mathbb{R}^{k_w}$ is a lookup function to a pre-trained textual embedding layer [12] that represents each word in vocabulary D as a vector in size k_w.

Due to the amount of parameters required to train the aforementioned contextual model, the textual network is implemented using a CNN-based architecture which is usually more computationally efficient than RNN. The semantic representations of users' and locations' reviews are fed to convolution layers, to detect parts of the text that best capture the review's meaning. These layers produce feature maps over the embedded word vector, using a window size of *t* and filter $K_{1:f} \in \mathbb{R}^{k_w \times t}$. As suggested by [3], ReLU is used as an activation function for this layer, resulting the output for a single convolutional neuron *j*:

$$z_j^u = ReLU(V_{1:n}^u * K_j + b_j) \tag{5}$$

Based on the standard CNN structure, feature maps produced by the convolution layers are reduced by a pooling layer:

$$o_j^u = max(z_1^u, z_2^u, ..., z_{n-t+1}^u), \tag{6}$$

where max-pooling is selected to identify the most relevant words. These are followed by fully connected layers that jointly model the different feature maps to result in the latent representations of h_u and h_p :

$$h_u = ReLU(W_1^u \times o^u + b_1^u) \tag{7}$$

To combine the outputs of the user's and location's fully connected layers to the same feature space, a shared layer is utilized. It concatenates its two inputs and learns their interaction by employing an additional hidden layer:

$$h_{reviews} = ReLU(W_2 \times (h_u \oplus h_p) + b_2) \tag{8}$$

The two neural networks are then finally merged to produce a prediction $\hat{y}_{up} \in [0,1]$. The last layers of the two networks, each representing a different view of the user-location interaction, are concatenated and fed to yet another hidden layer, responsible to blend the learning:

$$\hat{y}_{up} = \sigma(W_3 \times (h_{context} \oplus h_{reviews}) + b_3), \tag{9}$$

where the sigmoid function was selected to transform the hidden layer output to the desired range of [0,1].

B. Training the Network

To train the recommendation model, we adopt a pointwise loss objective function, as done in [1]–[3], [6], where the difference between the prediction $\hat{y}_{up}$ and the actual value y_{up} is minimized. To address the implicit feedback nature of LSBNs ,we sample a set of negative samples from the dataset, denoted as Y-.

Due to the implicit feedback nature of the recommendation task, the algorithm's output can be considered as a binary classification problem. As the sigmoid activation function is being used over the last hidden layer, the output probability can be defined as:

$$p(Y, Y\text{-} \mid E_u, E_p, V^u, V^p, \Theta_f) = \prod_{(u,p)\in Y} \hat{y}_{up} \prod_{u,p'\in Y\text{-}} (1 - \hat{y}_{up'}), \tag{10}$$

where E_u and E_p are the embedding layers for users and locations, respectively. Similarly, V^u and V^p are the textual reviews embedding layers and Θ_f represents the model parameters. Taking the negative log-likelihood of p results in the binary cross-entropy loss function for the prediction portion of the model:

$$L_{pred} = -\sum_{(u,p)\in Y\cup Y\text{-}} log\hat{y}_{up} + (1 - y_{up})log(1 - \hat{y}_{up}) \tag{11}$$

As there are two more outputs in the model, the users' social network $\psi_c^T E_u^T u_i$ and and the locations' distance graph $\psi_c^T E_p^T p_i$, two additional loss functions are required to train the network. We follow the process done in [1], assuming two users who share the same context should have similar embeddings. This is achieved by minimizing the log-loss of the context given the instance embedding:

$$L_{u\_context} = -\sum_{(u_i,u_c)} log(\phi_c^T E_u^T u_i - log \sum_{u_c'\in C_u} exp(\phi_{c'}^T E_u^T u_i)), \tag{12}$$

where $\phi_c^T E_u^T u_i$ is as defined in Eq. (1). Taking the binary class label into account prompts the following loss function, corresponding with minimizing the cross-entropy loss of user i and context c with respect to the y class label:

$$L_{u\_context} = -I(y \in Y)log\sigma(\phi_c^T E_u^T u_i) - I(y \in Y\text{-})log\sigma(-\phi_c^T E_u^T u_i), \tag{13}$$

where I is a function that returns 1 if y is in the given set, and 0 otherwise. The same logic is used to formulate the loss function for the POI context and will not be provided due to space limitations.

We simultaneously minimize the three loss functions L_{pred}, $L_{u\_context}$ and $L_{p\_context}$. The joint optimization improves the recommendation accuracy while enforcing similar representations for locations in close proximity and socially connected users. The loss functions are combined using two hyper-parameters, to weight the contextual contribution:

$$L = L_{pred} + \lambda_1 L_{u\_context} + \lambda_2 L_{p\_context} \tag{14}$$

To optimize the combined loss function, a method of gradient descent can be adopted, and more specifically we utilize the Adaptive Moment Estimation (Adam). This optimizer automatically adjusts the learning rate and yields faster convergence than the standard gradient descent in addition to making the learning rate optimization process more efficient. In order to avoid additional over-fitting when training the model, an early stopping criteria is integrated. The model parameters are initialized with Gaussian distribution, while the output layer's parameters are set to follow uniform distribution.

IV. EXPERIMENTS AND EVALUATION

A. Experimental Setup

To evaluate our proposed algorithm, we use Yelp's real-world dataset[2]. It includes a subset of textual reviews along with the users' friends, and the businesses locations. Due to the limited resources used in the model evaluation, we chose to filter the dataset and keep only a concise subset, where all users and locations with less than 100 written reviews or less than 10 friends are removed. The filtered dataset includes 141,028 reviews, and 98.08% sparsity for the rating matrix. The social and geographical graphs were constructed by random walks. 10% of the original vertices were sampled as base nodes while 20 and 30 vertices were connected to each base node for users and locations, respectively, with a window size of 3. To build the POI graph, two locations were directly connected if they are up to 1 km apart.

To test the performance of the model the original data was split to training-validation-test sets by random sampling, with the respective ratios of 56%-24%-20%, resulting 78,899 training instances. In addition, the input data was negatively sampled with 4 negative locations for every positive one.

To effectively compare our proposal with other alternatives, we adopt the same settings as applied in [1], [2]. The MLP input vectors are represented with an embedding size of 10, while two hidden layers are added on top of the merged result. Following the tower architecture, where the size of each layer is half the size of its predecessor, the number of hidden units are 32 and 16 for the first and second layers, respectively.

In the CNN each word is represented by a pre-trained embedding layer with 50 units, while the convolutional layer above is constructed with a window size of 10 and a stride of

[2]https://www.yelp.com/dataset/challenge

3. It results 3 feature maps that are flattened after performing the max-pooling operation with a pool size of 2. The results are further modeled by a hidden layer with 32 units. Following the merge of the two hidden units, their interaction is learned using another hidden layer with 8 units. To combine the three loss functions as described in Eq.14, we follow the results of [1] and set the hyper-parameters $\lambda_1 = \lambda_2 = 0.1$. For the training phase of the model, a learning rate of 0.005 was used over 50 maximum epochs and a batch size of 512 samples.

B. Baselines

To evaluate our algorithm, we chose to compare it to these four, empirically proven, frameworks:

- HPF [17]. A Bayesian framework for modeling implicit data using Poisson Factorization.
- NeuMF [2]. A state-of-the-art model combining Matrix Factorization with MLP on implicit ratings.
- PACE [1]. A MLP based framework with the addition of contextual graphs' smoothing.
- DeepCoNN [3]. A CNN based method that jointly learns an explicit prediction using the users' and locations' natural language reviews.

For the task of evaluating our model and the baselines, we chose to apply Accuracy and Mean Square Error (MSE) over all *n* test samples, as well as Precision (Pre@10) and Recall (Rec@10) for the average top 10 predictions per user.

The proposed model was implemented using Keras[3] on top of TensorFlow[4] backend.

C. Performance Evaluation

The performance of the proposed algorithm, TCENR, and the four baselines is reported in Table I. The presented results are based on the average of three individual executions.

TABLE I
PERFORMANCE OVER TEST DATA

Measure	HPF	NeuMF	Pace	DeepCoNN	TCENR
Accuracy	0.8141	0.8273	0.8239	0.8037	**0.8279**
MSE	0.1800	0.1421	0.1186	0.1454	**0.1171**
Pre@10	0.5526	0.6488	0.6406	0.5385	**0.6549**
Rec@10	0.3699	**0.5586**	0.5049	0.323	0.5215

As can be witnessed from the results, the proposed model, TCENR, outperforms the baselines in all measures but recall. Furthermore, TCENR was found to significantly improve HPF, Pace and DeepCoNN in terms of accuracy for $p < 0.05$ based on a one-sided unpaired t-test. The contrasting results in terms of precision and recall compared to NeuMF suggests that TCENR offers less, but more relevant recommendations to the user. Taking a closer look shows that, surprisingly, NeuMF outperforms PACE in accuracy, precision and recall. This may be due to the less sparse dataset tested, which does not allow the contextual regularization to be fully harvested. In addition, the use of only the first 500 words to represent the textual input for each user and location may explain the relatively low scores of the DeepCONN model on the dataset.

[3]https://keras.io
[4]https://www.tensorflow.org

D. Model Design Analysis

In this section we discuss the effect of several design selections over the suggested model's performance.

1) Merge Layer: The importance of the model's final layers, responsible for combining the dense output of both the MLP and convolutional networks, requires a close attention, as it effects the networks' ability to jointly learn and the prediction itself. To properly select the fusion operator the following methods had been considered:

- Combining the last hidden layers of the two models using concatenation. A model using this method will be denoted as $TCENR_{con}$ and described in Eq. 9.
- Merging the last hidden layers using dot product, resulting a model named $TCENR_{dot}$ that can be defined as:

$$\hat{y}_{up} = h_{context} \cdot h_{reviews} \tag{15}$$

- Combining the two previously described methods, where the two representations will be jointly learned by concatenation and dot product. The resulted model will be denoted as $TCENR_{dot\_con}$ and can be developed by combining Eq.9 and Eq.15 using addition and translating the result to a range of $[0, 1]$ with the sigmoid function:

$$\hat{y}_{up} = \sigma(\sigma(W_3 \times (h_{context} \oplus h_{reviews}) + b_3) + h_{context} \cdot h_{reviews}) \tag{16}$$

- Adopting a weighted average for the prediction result of the two networks. Denoted as $TCENR_{weight}$, this model can be defined as:

$$\hat{y}_{up} = \lambda_1 \sigma(W_4 \times h_{context} + b_4) \tag{17}$$
$$+ \lambda_2 \sigma(W_5 \times h_{reviews} + b_5) \tag{18}$$

As shown in Figure 2, $TCENR_{con}$ was found to produce the best results, and therefore integrated into the final model.

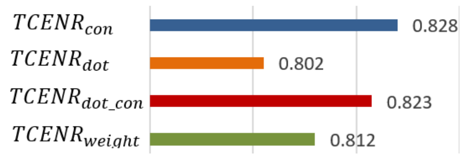

Fig. 2. Comparison of merging methods in terms of accuracy

2) MLP Layer Design: Although found by [2] that adding more layers and units to the MLP based recommender has a positive effect, the use of CNN and the additional hidden layer suggests it is a subject worth investigating. To this end, we test the proposed algorithm with 2,3 and 4 hidden layers used to learn the user-item interaction with contextual regularization in varying sizes from 8 to 64 hidden units. The results in terms of test set's accuracy are presented in Table II, where the number of hidden layers is defined as columns and the size of the first unit is presented as rows. Unlike the results in [2], we find that two hidden layers with 32 and 16 hidden units result the best performance for our dataset.

TABLE II
MODEL'S ACCURACY WITH DIFFERENT LAYERS

1^{st} layer	H=2	H=3	H=4
16	0.827	-	-
32	**0.837**	0.825	-
64	0.829	0.83	0.827

3) Number of Words: The use of written reviews in their original order allows the strengths of CNN to be exploited by finding the best representation for every few words, and eventually for the whole text. Our final dataset, however, is composed of very long reviews, where to fully represent a single user or location, more than 20,000 words are required, making it computationally expensive to extract relevant representations. To benefit from the sequential nature of the written reviews while keeping the solution feasible, the number of words were limited to a smaller number in the range of 500-3000. As can be witnessed from Figure 3, there is a slight improvement in accuracy as the number of words increase.

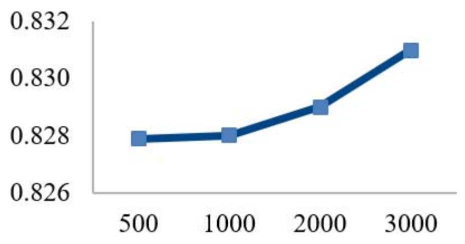

Fig. 3. Number of words comparison in terms of accuracy

V. CONCLUSION AND FUTURE WORK

In this paper, we developed a neural POI recommender system called TCENR. The system exploits data about users, locations, spatial data, social networks and textual reviews to predict the implicit preference of users regarding POIs. TCENR models two types of user-location interactions: native check-ins regularized by contextual information and the words used to describe the users' experiences. Evaluated over the Yelp dataset, the proposed algorithm outperformed the baselines in terms of accuracy, MSE and precision.

For future work, we will further evaluate the model when more words are used to represent reviews. In addition, we plan to investigate the proposed model's contribution to the cold-start problem by analyzing its performance on additional data, while taking new users and locations with few reviews into account.

REFERENCES

[1] C. Yang, L. Bai, C. Zhang, Q. Yuan, and J. Han, "Bridging collaborative filtering and semi-supervised learning: a neural approach for poi recommendation," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 1245–1254.

[2] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[3] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 425–434.

[4] H. Li, Y. Ge, R. Hong, and H. Zhu, "Point-of-interest recommendations: Learning potential check-ins from friends," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016, pp. 975–984.

[5] J. Manotumruksa, C. Macdonald, and I. Ounis, "A deep recurrent collaborative filtering framework for venue recommendation," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 1429–1438.

[6] A. Almahairi, K. Kastner, K. Cho, and A. Courville, "Learning distributed representations from reviews for collaborative filtering," in *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 2015, pp. 147–154.

[7] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 448–456.

[8] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 233–240.

[9] T. Bansal, D. Belanger, and A. McCallum, "Ask the gru: Multi-task learning for deep text recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 107–114.

[10] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 2011, pp. 287–296.

[11] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 2016, pp. 7–10.

[12] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[13] H. Yin, W. Wang, H. Wang, L. Chen, and X. Zhou, "Spatial-aware hierarchical collaborative deep learning for poi recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 11, pp. 2537–2551, 2017.

[14] H. Yin, Y. Sun, B. Cui, Z. Hu, and L. Chen, "LCARS: a location-content-aware recommender system," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 221–229.

[15] H. Yin, X. Zhou, Y. Shao, H. Wang, and S. Sadiq, "Joint modeling of user check-in behaviors for point-of-interest recommendation," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 1631–1640.

[16] W. Wang, H. Yin, L. Chen, Y. Sun, S. Sadiq, and X. Zhou, "Geo-sage: A geographical sparse additive generative model for spatial item recommendation," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1255–1264.

[17] P. Gopalan, J. M. Hofman, and D. M. Blei, "Scalable recommendation with hierarchical poisson factorization." in *UAI*, 2015, pp. 326–335.