Query Embedding Learning for Context-based Social Search

Cheng-Te Li and Yi-Chun Chen
Institute of Data Science, National Cheng Kung University, Taiwan
chengte@mail.ncku.edu.tw,guava830420@gmail.com

ABSTRACT

Recommending individuals through keywords is an essential and common search task in online social platforms such as Facebook and LinkedIn. However, it is often that one has only the impression about the desired targets, depicted by labels of social contexts (e.g. gender, interests, skills, visited locations, employment, etc). Assume each user is associated a set of labels, we propose a novel task, *Search by Social Contexts* (SSC), in online social networks. SSC is a kind of query-based people recommendation, recommending the desired target based on a set of user-specified query labels. We develop the method *Social Query Embedding Learning* (SQEL) to deal with SSC. SQEL aims to learn the feature representation (i.e., embedding vector) of the query, along with user feature vectors derived from graph embedding, and use the learned query vectors to find the targets via similarity. Experiments conducted on Facebook and Twitter datasets exhibit satisfying accuracy and encourage more advanced efforts on search by social contexts.

KEYWORDS

social search, context query, query embedding, social network

1 INTRODUCTION

Users in online social services such as Facebook, Instagram, and LinkedIn, often perform search tasks to find the particular persons. Given the name of the target person specified, the goal of social search is usually to accurately recommend targets who are user desires. However, it is every now and then that users have no idea about the names of the targets, and have only very limited information about the targets (e.g. gender, interests, hometown, company, and graduated school). On the other hand, sometimes users want to find some persons (e.g. experts or celebrities) who meet some requirements (e.g. skills, awards, and visited locations). The common settings of both scenarios is that the target persons can be depicted by a set of labels. We think it is potential to develop a system allowing such kind of social search without specifying the names of targets. In this paper, we propose a novel search task, *Search by Social Contexts* (SSC), which is to find the desired targeted persons using the social network behind users and the labels associated on users. SSC can be also considered as a *query-based people recommendation*. By specifying a small set of labels depicting the target, rather than her name, SSC is designed to recommend a list of persons such that the true desired target is covered and ranked at the top position in the list.

The proposed *Search by Social Contexts* lies in that there should be latent correlation between the input query labels and the target through the interactions between users and labels and the connections between users in social networks. To perform search and recommendation by contexts, we leverage the technique of node feature representation learning in graphs to model the hidden correlation between query labels and potential target users in the latent space. A novel method, *Social Query Embedding Learning* (SQEL), is developed to learn the feature representation (i.e., embedding vector) of the query, and use the learned feature vector to find the users with highest vector similarity scores as the recommended targets. Here is our basic idea: the query embedding feature vector is a certain ensemble of an accurate *query language model* and the *proximity distribution* between the query and other labels. The former depicts the semantic similarity considering the interactions between users and labels in the latent feature space, while the latter quantifies the connectivity of query labels in the graph structure. We conduct the experiments using Facebook and Twitter datasets to validate whether SQEL can truly find the targets. The results exhibit that SQEL outperforms a number of competing methods in terms of accuracy.

Related Work. J. Kleinberg [5] first defines *social search* in a network – searching over a social graph to find the set of individuals closest to a given user. Vieira et al. [12] re-formulate the task as a structural ranking problem via friend recommendation for users. Though Schendel et al. [9] recommend items possessing a given set of tags in a social tagging graph, social interactions are neglected. With search log data in LinkedIn and Facebook, Huang et al. [4] unfold the relationships between graph distances and user search behaviors. On the other hand, Roth et al. [8] and McAuley and Leskovec [6] group and recommend friends based on social activities of people. Though Mouratids et al. [7] use both graph and geographical distance for people search, labels associated on users are not considered. Last, a recent relevant study is Person-of-Interest Search (POI-Search) [3], which allows user-specified labels and the input user as the query. It is apparent that our SSC is more challenging since the input user is not considered. In addition, requiring the input user in POI-Search is neither realistic or generalized because the search task can be performed by anyone, rather than only users in the specific social service. Nevertheless, we will compare with POI-Search in the experiments.

2 PROBLEM STATEMENT

We first define a number of terms and notations, followed by the problem definition of *Search by Social Contexts*.

Definition 1: Social Network. A social network is a weighted graph $G = (V, E)$. Each node v has a set of labels. Each edge represents the interaction between two nodes. Edge weights refer to the interaction cost, where lower values indicate better interactions.

Definition 2: Context Query. A *context* query consists of a set of labels q, in which $\forall c_i \in q : c_i \in L$, where L is the universe set of labels.

Definition 3: User-Label Interaction. A user-label interaction $r_{ij} = \langle u_i, c_j \rangle$ is a connection that links user $u_i \in V$ with label $c_j \in L$ if u_i possesses c_j. We can have a set of user-label interaction $R = \{r_{ij}\}$ from all of the interactions between users and labels.

Definition 4: Hybrid Graph. A hybrid graph is a weighted graph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where the node set is the union of user set V and label set L, i.e., $\mathcal{V} = V \cup L$, and the edge set is the union of social tie set E and user-label interaction set R, i.e., $\mathcal{E} = E \cup R$. Note that we create a node for each label in the hybrid graph. The hybrid graph can be considered to jointly model the connections between users and the interactions between users and labels.

Problem Definition: Search by Social Contexts (SSC). Given a social network G, the label set L, the user-label interaction R, the context query q, and the budget parameter k, the task of *search by social contexts* (SSC) is to recommend a ranking list of k users S ($|S| = k$) such that the ground-truth target $u^\star$ can be not only covered by list S (i.e., $u^\star \in S$), but also accurately ranked at the highest position in list S.

3 THE PROPOSED METHOD

To deal with the task of search by social contexts, we propose a novel method, *Social Query Embedding Learning* (SQEL). The main idea of SQEL is to learn the feature representation (i.e., embedding vector) of the query, and use the learned feature vector to find the users with highest vector similarity scores as the results. SQEL consists of four main steps. First, based on the constructed hybrid graph, we can learn the feature representation of each node using the technique of *node2vec* [2], which maps nodes to a low-dimensional space of features that maximizes the likelihood of preserving graph neighborhoods of users and labels). For each label $c \in L$ and each user $u \in V$, we can derive d-dimensional vectors $\vec{c}$ and $\vec{u}$, where d is set as 128 through this work. When the query is given, the second step is to build the *query language model* (Section 3.3) that estimates the probability of any label c for the set of query labels q. The third step is to learn the feature representation of the query (Section 3.1). The query embedding vector $\vec{q}$ will be derived using the query language model and the graph proximity between nodes in the hybrid graph. The last step is to generate the recommended list of users based on the similarity between feature vectors of the query and users (Section 3.2).

3.1 Learning Query Representation

We aim to leverage the technique of maximum likelihood estimation (MLE) to learn the feature representation (i.e., embedding vector) of the query. The central idea of our method is to maximize the likelihood of an *query embedding probabilistic distribution* towards a query language model and the proximity distribution from query labels to other labels in the hybrid graph $\mathcal{H}$. The underlying assumption is that the query feature vector is a certain ensemble of query language model and the graph proximity distribution. The query feature vector is required to be reflected by not only the co-occurrence between labels, but also how users adopt labels in the graph. In other words, we seek to learn the query embedding feature vector that is close to the fusion of the query language model and the graph proximity distribution by MLE. Here we first define the probability of each label c given a query vector $\vec{q}$, termed *query embedding distribution*, as below:

$$p(c|\vec{q}) = \frac{s(\vec{c}, \vec{q})}{\sum_{c' \in L} s(\vec{c'}, \vec{q})}, \tag{1}$$

where $s(\cdot, \cdot)$ is a similarity function that calculates the similarity between two feature vectors, $\vec{c}$ is the embedding feature vector of label c, and L is the set of all labels. Then for the query q, we assume that there is a query language model θ_q, which quantifies the importance of each query label $c_i \in q$ in the query. Let $\vec{q^\star}$ be the optimal query feature vector, which is the query embedding vector (Eq. 1 that is closest to the ensemble of the query language model θ_q and the proximity distribution $p_{\mathcal{H}}(q \rightarrow c)$. Therefore, we propose to find the query feature vector that maximizes the following log-likelihood function:

$$\vec{q^\star} = \arg\max_{\vec{q}} \sum_{c \in L} \left(\alpha \cdot p(c|\theta_q) + (1 - \alpha) \cdot p_{\mathcal{H}}(q \rightarrow c)\right) \log p(c|\vec{q}), \tag{2}$$

where the proximity distribution from query labels to other labels $p_{\mathcal{H}}(q \rightarrow c)$ is derived using *Random Walk with Restart* [11] in the hybrid graph $\mathcal{H}$, $\alpha \in [0, 1]$ is the parameter to determine the relative importance between the query language model and the proximity distribution, and we will explain how to obtain the query language model θ_q later in Section 3.3. We empirically set $\alpha = 0.7$ by default to derive the best results. Directly maximizing the objective in Eq. 2 is time-consuming due to the normalization in the denominator of Eq. 1. And such normalization relies on the query vectors, so we can have offline processing. We resort to take the relaxation [14] that assumes the normalization is equal for all query vectors (Note that it is a kind of heuristic). Consequently, we can use the similarity $s(\vec{c}, \vec{q})$ to replace $p(c|\vec{q})$ in Eq. 2. Since the similarity depends on the query vector $\vec{q}$, maximizing objective function can be still influenced by the query.

We take advantage of *softmax* function to estimate the similarity between learned feature vectors of labels. We can define the similarity function $s(\cdot, \cdot)$ as below:

$$s(\vec{c}, \vec{c'}) = \exp\left(\frac{\sum_{i=0}^{d} \vec{c}_i \vec{c'}_i}{\|\vec{c}\| \|\vec{c'}\|}\right), \tag{3}$$

where d is the dimension of learned feature vectors. We assume that each learned feature vector of label is a unit vector, i.e., the norm equals to 1, under the setting without loss of generality. Then we can rewrite the objective function from Eq. 2 to the following:

$$\vec{q^\star} = \arg\max_{\vec{q}} \sum_{c \in L} \left(\alpha \cdot p(c|\theta_q) + (1 - \alpha) \cdot p_{\mathcal{H}}(q \rightarrow c)\right) \cdot \left(\frac{\sum_{i=0}^{d} \vec{c}_i \vec{q}_i}{\|\vec{q}\|}\right) \tag{4}$$

By letting the query feature vector be a unit vector, i.e., $\|q\| = 1$, we can have the objective function in the form of Lagrange function:

$$\mathcal{L}(\vec{q}, \lambda) = \sum_{c \in L} \left(\alpha \cdot p(c|\theta_q) + (1-\alpha) \cdot p_{\mathcal{H}}(q \to c)\right) \sum_{i=0}^{d} \vec{c}_i \vec{q}_i + \lambda \left(1 - \sum_{i=0}^{d} (\vec{q}_i)^2\right), \tag{5}$$

where λ is the Lagrange multiplier. Through the mathematical optimization method for Lagrange multipliers, we can derive the first derivatives of the Lagrange objective function, as below:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \vec{q}_i} = \sum_{c \in L} \vec{c}_i \left(\alpha \cdot p(c|\theta_q) + (1-\alpha) \cdot p_{\mathcal{H}}(q \to c)\right) - 2\lambda \vec{q}_i \\ \frac{\partial \mathcal{L}}{\partial \lambda} = 1 - \sum_{i=1}^{d} (\vec{q}_i)^2 \end{cases}, \tag{6}$$

where $\vec{q}_i$ is the i-th value of query feature vector $\vec{q}$. By setting the partial derivatives in Eq. 6 as zero, we can obtain the closed-form solution of our objective as follows:

$$\vec{q}_i = \frac{\sum_{c \in L} \vec{c}_i \left(\alpha \cdot p(c|\theta_q) + (1-\alpha) \cdot p(q \to c)\right)}{\sqrt{\sum_{j=1}^{d} \left(\sum_{c \in L} \vec{c}_j \left(\alpha \cdot p(c|\theta_q) + (1-\alpha) \cdot p_{\mathcal{H}}(q \to c)\right)\right)^2}}. \tag{7}$$

Eventually we can have the query feature vector $\vec{q}_i$ using the closed-form formula in Eq. 7.

3.2 Deriving the Ranking List

Since our goal is to recommend a ranking list of k targeted users S, we aim at estimating the *social affinity* between the query q and each user $u_i \in V$ based on the learned embedding vectors of users and query $\vec{u}_i$ and $\vec{q}$. Users with higher social affinity scores to the query are considered as more relevant to the query, and thus are ranked at higher positions in the final list S. We select the top k users with highest social affinity scores. We define the *social affinity* $\delta(u_i, q)$ between user u_i and query q using the similarity between two embedding vectors defined in Eq. 3: $\delta(u_i, q) = \frac{s(\vec{u}_i, \vec{q})}{\sum_j s(\vec{u}_j, \vec{q})}$. We can simplify the computation of $\delta(u_i, q)$ by discarding the denominator because it is the same for all users, i.e., $\delta(u_i, q) \approx s(\vec{u}_i, \vec{q})$.

3.3 Query Language Model

The word embedding techniques have been validated to improve the accuracy of document retrieval [1][13]. Based on such an idea, to develop an accurate language model θ_q for the query, we further extend the embedding from vocabulary terms to label nodes in the graph. To estimate the probability of any label c given the language model θ_q, i.e., $p(c|\theta_q)$, our main idea is to impose an assumption: the conditional independence between query labels – when users performing context queries, they tend to describe the underlying target using clues from multiple aspects that might not be directly correlated (e.g. $q = \{\texttt{NYC}, \texttt{Python}, \texttt{Spaghetti}, \texttt{BMW}\}$). Note that this conditional independence assumption may not hold for all cases of user queries. To simplify the computation, we make such an assumption, and leave the modeling of correlation between query labels in the future work.

In order to estimate the query language model $p(c|\theta_q)$, the Bayes rule is applied: $p(c|\theta_q) = \frac{p(\theta_q|c)p(c)}{p(q)} \approx p(\theta_q|c)p(c)$ (the term $p(q)$ can be neglected as it is independent of label c). Based on the abovementioned conditional independence between query labels, we can have the query language model as below:

$$p(c|\theta_q) \approx p(c_1, c_2, \cdots, c_k|c)p(c) = p(c) \prod_{i=1}^{k} p(c_i|c), \tag{8}$$

where $\{c_i | i = 1, 2, \cdots, k\}$ are the set of query labels, and k is the size of query label set. We compute $p(c_i|c)$ in Eq. 8 using the similarity formula Eq. 3 with the learned embedding feature vectors that depict the high-order relationships between labels and users in the hybrid graph, as follows.

$$p(c_i|c) = \frac{s(\vec{c}_i, \vec{c})}{\sum_{c_j \in L} s(\vec{c}_j, \vec{c})}. \tag{9}$$

In addition, we can to compute $p(c)$ based on the law of total probability: $p(c) \approx \sum_{c_j \in L} s(\vec{c}_j, \vec{c})$. A label with higher similarity with all the other labels will derive higher probability.

4 EVALUATION

The experiments are conducted using Facebook and Twitter social network data provided by SNAP[1]. The data statistics are shown in Table 1, in which CC means clustering coefficient while APL refers to average path length in the social network G. Labels in both data used include the terms of school, hometown, employer, skill, gender, location, position, etc. In the social networks, we compute the edge weights using the Jaccard coefficient: $w_{u,v} = 1 - (|L_u \cap L_v| / |L_u \cup L_v|)$, where L_u is the set of labels of node u. Our general aims at examining whether or not the proposed SQEL can help accurately find the desired targets for the task of search by social contexts.

Table 1: Statistics of Facebook and Twitter data.

	#nodes	#edges	CC	APL
Facebook	4,039	88,234	0.606	4.7
Twitter	81,306	1,768,149	0.565	4.5

We compare SQEL with four competing methods: (a) *label matching* (LM): using the query label set Q to find a list of possible target users u such that the labels of u have the highest overlap with Q, $LM(u, Q) = \frac{|L_u \cap Q|}{|L_u|}$; (b) *Center-Piece Subgraph* (CePS) [10] with OR operation: consider those users containing at least one label in Q as the source nodes in CePS, run the CePS algorithm, and return a list of possible targets u with the highest CePS scores $CePS(t, Q)$; (c) *CePS+LM*: the selection procedure is similar to CePS but returns the list of possible target users with the highest average rank scores of CePS and LM, $CePSLM(u, Q) = \frac{1}{2}(rank(CePS(u, Q) + rank(LM(u, Q)))$; and (d) *POI-Search* [3] is a greedy heuristic algorithm to find the target covering the query labels and having strong social interactions and higher proximity scores towards the query labels.

In the evaluation setting, we first choose users in a random manner as the ground-truth targets. Then for each ground-truth target user $\bar{u}$, we randomly select k labels from the label set of users $\{\bar{u}\} \cup \mathcal{N}(\bar{u})$, denoted by $L_{\{\bar{u}\} \cup \mathcal{N}(\bar{u})}$, to be the query label set $Q_{\bar{u}}$, where $\mathcal{N}(\bar{u})$ is the immediate neighbors of user $\bar{u}$ in the social network G, $L_X = \bigcup_{v in X} L_v$ is the union of label sets of users X, and

[1] http://snap.stanford.edu/data/egonets-Facebook.html
http://snap.stanford.edu/data/egonets-Twitter.html

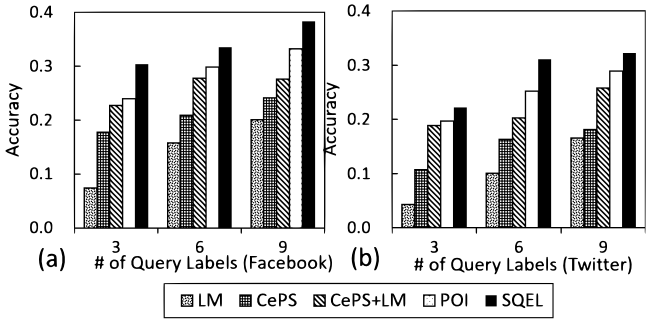

Figure 1: Accuracy scores by varying the number of query labels in Facebook ad Twitter data.

$|Q_{\bar{u}}| = k$. Here we set $k = 10$ by default, and will vary k to simulate different knowledge extent about the target. To control the hardness of the search task, we introduce a *hardness* parameter $\tau \in [0, 1]$. In selecting the query labels, we randomly choose τk labels from $L_{\mathcal{N}(\bar{u})}$ and $(1 - \tau)k$ labels from $L_{\bar{u}}$. Higher τ values make the task more challenging. The design τ can also reflect that real users sometimes provide inaccurate evidences about the desired targets. We set $\tau = 0.6$ by default. We generate 500 pairs of query Q_i and corresponding ground-truth user $\bar{u_i}$ based on the above setting, i.e., $Q_i \in T, |T| = 500$. Given the top-n targets $S_n(Q_i)$ returned by a method and the ground-truth targets $S^\star(Q_i) = \bar{u_i}$, we define the evaluation metric using accuracy: $accuracy = \sum_{Q_i \in T} hit(S_n(Q_i), S^\star(Q_i))/|T|$, where $hit(S_n(Q_i), S^\star(Q_i)) = 1$ if $|S_n(Q_i) \cap S^\star(Q_i)| \neq 0$; otherwise: $hit(S_n(Q_i), S^\star(Q_i)) = 0$. It can be observed that as n increases, the search task tends to be easier and will obtain higher accuracy. We set $n = 5$ by default.

The evaluation plan consists of two parts. The first is General Evaluation: varying the number of query labels $k = |Q|$ to show the performance of SQEL. A larger Q set means that the query contains more information depicting the target. The second is sensitivity analysis: varying the hardness τ – higher τ will introduce more indirect labels that might be less relevant to the target, and thus make the task more difficult; and vary the parameter α in Eq. 2 – higher α will make the query embedding more significantly learned from the query language model.

Experimental results are shown in Figure 1. By varying the number of query labels k, we can find that the proposed SQEL outperforms the other competing methods, especially when the number of query labels increases. It is because more labels provide more evidences about the desired targets. Though the integration (CePS+LM) of structural connectivity in CePS and label matching in LM can boost the effectiveness, the accuracy values are still lower than SQEL. In addition, SQEL also clearly and consistently outperforms the recent work POI-Search. These results imply that learning the feature representation of query and users can effectively capture the search intent and accurately obtain the targets.

Figure 2 shows the results of sensitivity analysis for the parameters of (a) the hardness τ and (b) α that determines the importance between query language model and proximity distribution in Eq. 2. We can apparently find, in Figure 2(a) that higher values of τ lead to lower accuracy scores while lower τ produces better performance, since more labels describing the target make the task easier.

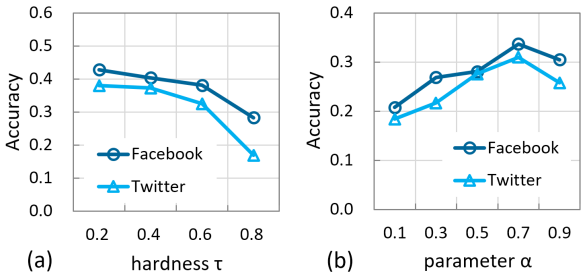

Figure 2: Accuracy by varying (a) the hardness parameter τ, and (b) α in Eq. 2.

The results are consistent in both Facebook and Twitter datasets. On the other hand, in Figure 2(b), too high or too low values of α values lead to worse accuracy scores. The performance tends to be better when *alpha* is around 0.7, which is therefore used as the default value. Such results also inform us the influence of query language model is more significant than the proximity distribution in learning query representation.

5 CONCLUSIONS

This paper proposes the task of *Search by Social Contexts* (SSC) to recommend the target individuals desired by the user based on a set of query labels in social networks. The main technical contribution is the novel development of *Social Query Embedding Learning* (SQEL), which learns the feature representation of the query considering the query language model and the proximity distribution in a user-label interaction hybrid graph. Experimental results show promising accuracy. We will also evaluate SQEL using a user study by a developing Facebook application. The future work is to extend SSC by imposing spatio-temporal contexts – the targets are required to reflect the current time and location of the user.

REFERENCES

[1] Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth J.F. Jones. 2015. Word Embedding Based Generalized Language Model for Information Retrieval. In *SIGIR*.

[2] Aditya Grover and Jure Leskovec. 2016. Node2Vec: Scalable Feature Learning for Networks. In *KDD*.

[3] Hsun-Ping Hsieh, Cheng-Te Li, and Rui Yan. 2015. I See You: Person-of-Interest Search in Social Networks. In *SIGIR*.

[4] S.-W. Huang, D. Tunkelang, and K. Karahalios. 2014. The Role of Network Distance in LinkedIn People Search. In *SIGIR*.

[5] J. Kleinberg. 2006. Social Networks, Incentives, and Search. In *SIGIR*.

[6] J. McAuley and J. Leskovec. 2012. Learning to Discover Social Circles in Ego Networks. In *NIPS*.

[7] K. Mouratidis, J. Li, Y. Tang, and N. Mamoulis. 2015. Joint Search by Social and Spatial Proximity. In *IEEE TKDE*.

[8] M. Roth, A. Ben-David, D. Deutscher, G. Flysher, I. Horn, A. Leichtberg, N. Leiser, Y. Matias, and R. Merom. 2010. Suggesting Friends Using the Implicit Social Graph. In *KDD*.

[9] R. Schenkel, T. Crecelius, M. Kacimi, S. Michel, T. Neumann, and G. Weikum. 2008. Efficient Top-k Querying over Social-Tagging Networks. In *SIGIR*.

[10] H. Tong and C. Faloutsos. 2006. Center-Piece Subgraph: Problem Definition and Fast Solutions. In *KDD*.

[11] H. Tong, C. Faloutsos, and J.-Y. Pan. 2006. Fast Random Walk with Restart and Its Applications. In *ICDM*.

[12] M. V. Vieira, B. M. Fonseca, R. Damazio, P. B. Golgher, D. D. C. Reis, and B. Ribeiro-Neto. 2007. Efficient Search Ranking in Social Networks. In *CIKM*.

[13] Hamed Zamani and W. Bruce Croft. 2016. Embedding-based Query Language Models. In *ICTIR*.

[14] Hamed Zamani and W. Bruce Croft. 2016. Estimating Embedding Vectors for Queries. In *ICTIR*.