Lifestyle-based Personalized Query in Drug Networks *

Fubao Wu
University of Massachusetts, Amherst

ABSTRACT

Drugs contains multiple properties including indication, target, side-effect etc. Common side effects of a drug should happen to everyone. However, side effects vary among individuals in real life. For different people with different lifestyle, such as vegetarian, exercise, they are likely to have different reactions to a drug. To assist media practitioners and patients in prescribing drugs, we provide personalized query system, not only providing drug query's general property, but also considering the effect of personal lifestyle on the drug's side effect. These data are from user's posts in web discussion forum we extract users' lifestyle information and drug-side effects To provide convenient personalized query support, we extend user's lifestyle and drug-side effect information into common drug-side effect database and provide a personalized distributed query system to find the rich information about personal behavior's effect on the drug's side effect.

1. INTRODUCTION

To prescribe a drug, medical practitioner needs to consider multiple information about drugs properties, including indication, target, side effect. There are some existing resources like drugBank,SIDER which provides these rich information. But these information didn't consider personalized lifestyle profile with drug. The lifestyle profile include physcial activities, weight, vegetarian etc. Because different users could have different reaction to a drug with different lifestyle, it is necessary to dig out this information and further help practitioner better prescribe drugs. With the popularities of networks and social media, more and more users prefer to expose their personal life situation and drug usage which provide this opportunity to extract personal information. Currently, extensive researches on adverse drug reaction(ADR) extraction in pharmacovigilance [**?**, **?**, **?**] have been done. For those existing researches, they still only consider the general drug-side effect extraction from these sources. However, there are also other abundant information in these posts about user life and health behavior. Based on this consideration, our research is aimed to provide a tool to do personalized query based on the extracted user lifestyle and previous database information. commonly, a drug has side effects, indication, target and pathway etc. Given some side effects and indications, we query some candidates drugs to get drugs. In our study,we extend users lifestyle to the graph schema to personalize the drug graph query. As shown in the Figure 1, we add an weight W(Drug,SE, lifestyle) to the drug and side effect nodes based on the extracted lifestyle information of users. Weight between other nodes are 1 which is not considered.

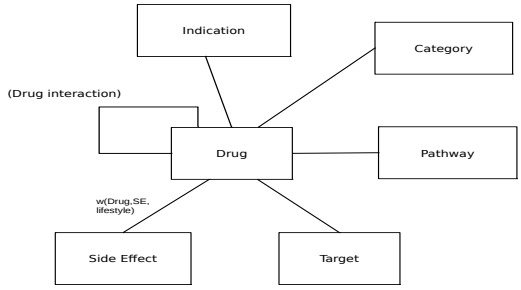


Figure 1: Schema of personalized drug graph

In order to cope with the incomplete and noisy data from database and social media, we not only consider exact answers which are directly connected nodes, but also the indirect query based on metapath methods. We do the following queries:

(1)Without personalized lifestyle factors, given some known specific nodes, we want to query a set of drug nodes and get the top K best candidate drugs. For example, a doctor want to prescribe some drugs that cure diabetes but it does not interact with another drug(Aspirin) which is being taken by the patient.The drugs should also not cause dizziness and weight gain. This query is shown in Figure 2.

(2)Considering personalized lifestyle, given some known specific nodes, we want to query top K best candidate drugs. For example, considering a patient who doesn't exercise, we want to prescribe some drugs that cure diabetes but it does not interact with another drug(Aspirin) which is being taken by the patient.The drugs should also not cause dizziness and weight gain which is shown in Figure 3.

*for use with vldb.cls

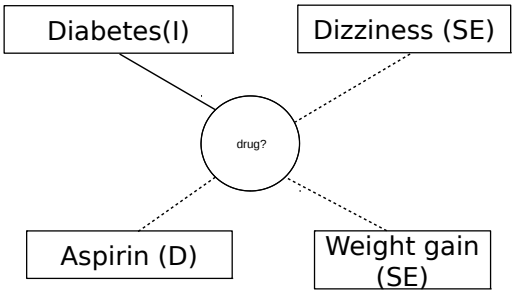


Figure 2: Example of query graph without lifestyle

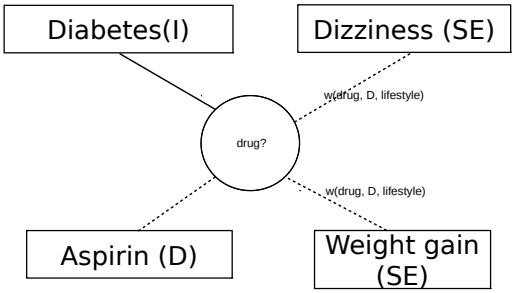


Figure 3: Example of query graph with lifestyle

These drug database could be modeled as a kind of heterogeneous information network which contains different types of entities and different relationship between them. With the increase of network information and drug database, effective performing on this query is challenging. Answering query on drug networks is a subgraph matching problem.

Our study is to provide a distributed query system and extend our lifestyle factors as personalized query system. The key contributions of this paper are as follows:

- We construct a classification to mine the user lifestyle from user posts on a web forum. Use this classification, we obtained the user lifestyle data which is extended into the original drug graph.
- We also mine user drug side effect information on the web forum. Then we match these two data together with the user id to get users' lifestyle and drug-side effect database. According to this data resources, we construct personalized weight for drug-side effect nodes.
- We extend an personalized lifestyle into drug network and better help to get personalized result for drug prescription
- We develop a distributed query system, and do general query and personalized lifestyle query in this system.

2. PROBLEM DESCRIPTION

Our drug database is heterogeneous information networks which is modeled as drug graph with multiple different nodes, relationship and weights.

2.1 Drug graph

Drug graph is modeled as a typed graph $G(V, E, T, W)$ where V is a set of nodes,E is a set of undirected edges, and T is a set of node types. Every node has a type. We denote the type of node v by type(v). Each edge between nodes has an weight. Generally, all the edges' weight values are 1. While we consider lifestyle between Drug and Side Effect nodes, it is separately quantified by the extracted database. For simplicity, we consider an undirected graph. However, for drug nodes, we have positive edge and negative edges. A positive edge means a node type has positive effect on drug type. A negative edge indicates a drug has negative effect on another node type such as side effect.

As shown in Figure 1, there are 6 different node types. They are drug (D), pathway (P), drug target (T),side effect (SE),drug indication (I), drug category(C).

2.2 Query graph

As show in Figure 2. A query is represented as a graph $Q(V, E, T, W)$. We refer to nodes in the query graph as query nodes. The query graph follows the same schema as the Drug graph as a subgraph. There are two groups of nodes in the query nodes. One is the group of known nodes(also called reference nodes) which is the query conditions provided by users. The other is set of candidates nodes(also called variable nodes) which users want to get.

3. METHODOLOGY

3.1 Data Sources

For general query graph, we extract the data from DrugBank, SIDER,KEGG Drug database about drug, drug side effect, drug indication, drug category, drug target to model as heterogeneous drug networks.

When considering personalized query, we first consider diabetes disease data only to initialize this exploration and simplify the process, We crawled the data from diabetes.co.uk forum. It is the leading community website and forum for people who share their own experiences in relation to diabetes and its associated complications. We crawled the forum thread and parse the following information: title, user identifier, post date and time and text of each post. After crawling these posts, we combined the same users post together and thence get 6526 users' posts instance in total between 2014-2015. The posts are furthered extracted for lifestyle and drug-side effects.

3.2 Data Extracton

To provide the personal query, we need to collect the personalized data for the query system. To effective mine the lifestyle and drug-side effects, we use natural language processing and supervised machine learning method.

3.2.1 Lifestyle Data Ming

Lifestyle could have broad implication. Here we consider the basic aspects, including physical activity, weight loss, vegetarian etc. To initialize the process of research, we focus on the exercise-related activity first. To extract the exercise from users post, we first filter the posts and get 1389 posts that contains exercise-related words. Using this filtered posts as instances for machine learning classification. All the instances are manually annotated for cross validation training and testing. The proposed method is as follows:

1. we performed the preprocessing such as tokenization, lowercasing and stemming of all the terms using porter stemmer. We also use stopword from Stopwords Corpus distributed with NLTK to remove irrelevant information.
2. We use the following features to do classification.
(1)semantic types: tf-idf we compute the tf-idf values for the semantic terms. We extract the first frequent 10000 words/phrase as dictionary and calculate the tf-idf as our feature set from our corpus. There are 10000 features here.
(2) N-grams Our second feature set consists of word n-gram of the comments. N-gram is a sequence of continguous n words in a text segment. It enables us to represent a document using the union of its terms. We use 1-, 2-,3-grams as features here.
(3)topic-model features we use topic modeling to extract topic from post as features. Our intuition and observation from some posts that users in some posts mentions exercise a lot in his lifestyle situation as a topic which could be helpful to differentiate from other posts. We use non-negative matrix factorization technique to extract the topics and ge the score of the topic terms as features in each instance.
(4) Exercises and frequency pairs. We observed that in many posts authors mentions their exercise with the frequency. For example the following instances from user posts: I walk for half an hour every day Ive started walking 3 or 4 times a week I walk or exercise everyday So we consider the features: frequency-exercise pairs. We use exercise-related dictionary mentioned before and also generate frequency terms dictionary in English manually. The frequency dictionary consists of a group of words(56 in total): constantly, continuously,daily, every hour, every month... We detect whether a sentence express the idea of this pairs, when a exercise token in exercise-related dictionary is found, we detect all the token before the current token and after the token in the sliding window. If the frequency token is detected in the window, then the feature is activated which the value is added. Then we use the accumulated value as the features. The initial feature value is 0. The window length is currently set to 5 according to observation of user post content.
(5) PRP-exercise pairs features Considering some users mentions exercise lifestyle which is not himself/herself. For example: "you should take exercise every day!" So we consider the 'I', My detection with exercise-related features to calculate the score for this pairs in a sentence.

3. we use supervised machine learning methods-nave bayes, logistic regression, SVM classifiers in our classification setting and compare their performance. 10-fold cross validation is done in the classification which is 10% are test and 90% are training data and the classification iteration is 10 times.

3.2.2 Drug-Side Effect Data Ming

To simplify the analysis, we first only consider diabetes here. Diabetes disease including type 1, type 2 and Gestational diabetes etc. Ping etc[] analyzed small database about drug-disease and drug-side effects database extracted from Drugbank and SIDER database. We use small database to get the common drugs for diabetes and the corresponding side effects for these drugs. There are 37 drugs extracted and 2562 pairs of diabetic drug-side effects obtained There are common drug-side effect. To do personalized query, we need to dig out the user's side effect with the drug taken. Similarly, we use natural language and machine learning method drawn from diabetic.co.uk web forum. We filtered 2474 posts mentioning drugs treating diabetes. Also, We use common drug-side effect as keywords to filter 615 posts mentioning the drugs and the corresponding side effects. Use this as training and test data instances, we construct classifier to extract drug-side effects. We use a method proposed by[ ] which uses machine learning classification based on the effective features. We apply ADR lexicon matches, n-gram, topic-based feature, negation, Sentiword scores as features to do SVM and NB classification to identify the side effects in the posts. We matched the data with lifestyle data, we obtained 145 user posts which mentioned exercise or no-exercise and drug-side effect in the posts. Other user either mention exercise or drug-side effect in different group.

3.3 Query Method

We model query as subgraph graph query problem. To define the nodes similarity,here we use metapath-based similarity measures. Based on metapaths of drug query, we use multiple edges of metapath types which are D-SE edge, D-I edge and D-D edge, D-P edge, D-T edge etc. for every edge, we select 5 types of metapaths. The reason we select these path type is based on the potential relationship between nodes. For example, the path type D-D-SE is based on the hypothesis that drugs that interact tend to have similar side effects. The D-I-D-SE indicate two drugs could share similar properties-side effect and target. Similarly, other metapath types have the similar rules to be derived from.

Table 1: three metapath types example

Path ID	D-SE edge	D-I edge	D-D edge
p1	D-I-D-SE	D-I-D-I	D-I-D-D
p2	D-P-D-SE	D-P-D-I	D-P-D-D
p3	D-T-D-SE	D-T-D-I	D-T-D-D
p4	D-D-SE	D-D-I	D-D-D
p5	D-SE-D-SE	D-SE-D-I	D-SE-D-D

3.4 Edge Likelihood Function

Using the proposed metapath types before, for every specific unknown nodes, we need to calculate 5 metapaths(path ID is 5) in each correspondent edge.Take a SE node as an example, there are 5 metapaths in D-SE edges. For example, the path $d_1(D)-t_1(I)-d_2(D)-se_1(SE)$ represents the relationship where a drug shares a target with another drug that has a particular side effect. The likelihood of an edge is quantified as a function of the number of paths in different path type. Let $M = m_1, m_2, ..., m_k$ be the set of path types of an edge. Let $c_{m_l}(v_i, v_j)$ be the number of paths between v_i and v_j that has type $m_l(1 \leq l \leq k)$. The likelihood of an edge between v_i and v_j, denoted by $p(v_i, v_j)$, is based on a logistic regression model as follows:

$$p(v_i, v_j) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \cdot c_{m1}(v_i,v_j) + ... + \beta_k \cdot c_{mk}(v_i,v_j))}} \quad (1)$$

where $\beta_0, \beta_1, ..., \beta_k$ are the model parameters indicating the importance of each metapath type.

3.5 Score Function

In a given query, there are multiple known nodes and correspondent likelihood values to a unknown node. Intuitively, if the edge of the query nodes should correspond to the edges

in the query graph. More specifically, if there is a positive edge between q_i and q_j, then there should be an edge between $f(q_i)$ and $f(q_j)$, the matches of q_i and q_j. The score of an answer is defined as a function of these likelihoods based on this concept. As mentioned before, between nodes,there are positive edges and negative edges. LetE_Q^+ and E_Q^- indicates the set of positive edges and the set of negative edges in the query graph, respectively. The score of an answer is defined as follows:

$$S(f) = \prod_{e(q_i,q_j)\in E_Q^+} p^{'}(f(q_i), f(q_j)) \prod_{e(q_i,q_j)\in E_Q^-} (1 - p^{'}(f(q_i), f(q_j))) \quad (2)$$

If there is direct edge between node v_i and v_j, $p^{'}(v_i, v_j) = 1$, otherwise $p^{'}(v_i, v_j) = p(v_i, v_j)$. the first product in $S(f)$ considers the edge likelihood among the node pairs that are connected by a positive edge. The second product considers the complement of the edge likelihood, that is the node pairs that are connected by negative edges.

4. ALGORITHM DESCRIPTION

We use master/worker models to design our algorithms. There are one master and multiple workers. Input data graph are hash-partitioned stored in different workers here. There is a distributed in-memory state table that is distributed across the workers. It has structure $T(K, V1, V2, D)$ in which K stores the graph vertices, D stores the partitioned graph information, V1 and V2 are used to store the old and new metapath values for different unknown nodes of different sources. In every BFS iteration, the metapath number of every node pair would be updated until the maximum iteration number is satisfied which achieves the termination condition. The maximum iteration number is decided by the maximal length of metapath types of an edge for a given node pair.

4.1 Subgraph Query Algorithm

To get the topK answer score of unknown nodes for a subgraph query, there are two main steps. Firstly, we need to get the metapath values of traversed unknown nodes for different metapath types. Secondly, we calculate the likelihood function for between unknown nodes and known nodes pairs.Then, we can get the final answer score of the query based on the likelihood and positive edges or negative edges.

To get the topK answer score of unknown nodes for a query, there are two main steps. Firstly, we need to get the metapath values of traversed unknown nodes for different metapath types. Secondly, we calculate the likelihood function for between unknown nodes and known nodes pairs.Then, we can get the final answer score of the query based on the likelihood and positive edges or negative edges.

The whole process for a subgraph query is shown as Algorithm 1.

4.1.1 Calculate Metapath Numbers

We calculate every possible unknown nodes's metapath number values to get the likelihood. Intuitively, there are massive calculation of unknown nodes. However, These possible unknown nodes are determined by the BFS traversed level which is decide by the maximal metapath length. Specifically, for different metapaths type, here the maximum length of metapath is 4. If we do BFS search from one unknown node, only 3 levels of iteration(that is, the maximum iteration number is 3) are needed to calculate the number of that metapath and only the traversed unknown nodes, denoted by V^T, are needed to be computed. Therefore, the computation expense of BFS is acceptable. Specifically, there are two parts in calculating metapath numbers. One is to do BFS to get the traversed nodes's metapath. The second part is to update the old metapath value from new metapath values in table. In the first part, for each BFS iteration of one source, we operate BFS in distributed workers. The master would notify workers to do BFS with assigned iteration level. When one iteration finishes, every worker maintains a queue storing the nodes for next BFS traversal.

In the second part, when one iteration finishes in each worker, because different workers may finish iteration at different paces, Here we use synchronized method. When one iteration finishes, it will send message to master. While the master receives all the workers' notifications, it will notify all workers to begin updating old metapath number from new metapath number, which is to copy metapath number in V2 to that in V1 and meanwhile initialize V2 to 0 for each visited node from last iteration.

Algorithm 2 shows one iteration executed for calculating metapath number of multiple unknown nodes for a subgraph query in distributed workers. In one iteration, the most important work is to store the qualified data node(which node's type is in the metapath) and queue the visited data along the metapath for next iteration visit in each worker.

4.1.2 Calculate Top-k Scores

When all the workers finish the iteration numbers notified by the master, they will begin to calculate the edge's likelihood function and score function locally. For every BFS visited nodes in queue, they calculate their likelihood according to equation 1 function and score function according to the equation 2. After they have obtained the local top-k score answers for visited nodes V^T, they send the local scores to the master and the master would receive all the aggregated worker's local scores. Then master sorts the re-

Algorithm 1: subGraphQueryTopK(G,V_Q^S,K)

Input: data graph G, one set of known nodes V_Q^S, metapaths MP, K
Output: topK query answer
InitializeGraph(G)
$iterationNum \longleftarrow max(metapath\_length)$
while *iterationNum* **do**
  if *not first iteration* **then**
    update $V1 \longleftarrow V2$
    initialize $V2 \longleftarrow 0$
  end
  metapath values MPVals(V^T) $\longleftarrow$ **OneIterationMetapath(V_Q^S,MP)**
  $iterationNum = iterationNum - 1$
end
top-k Answer(W,K) $\longleftarrow$ **TopKAnswers(V^T, MPVals(V^T))**
return Answer(W,K)

Algorithm 2: OneIterationMetapath(V_Q^S,metapath MP)

Input: data graph G, a set of known nodes V_Q^S, metapaths MP
Output: queue q[w] along the metapath in each worker
initialize queue for in every worker
for *every metapath p in* MP **do**
  for *every known node* s_i *in* V_Q^S **do**
    initialize $mp[s_i][p] \longleftarrow 1$ in table V1, V2
  end
end
for *every known node* s_i *in* V_Q^S **do**
  for *every meathpath p in* MP **do**
    while *queue u not empty* **do**
      $u \longleftarrow queue(u)$
      if *u's* $mp[s_i][p]$ *empty* **then**
        initialize u's $mp[s_i][p] \longleftarrow 0$ for table V1, V2
      end
      for *all* $w \in$ *all u neighbors* **do**
        if *w type is in the metapath p* **then**
          if *w is in local worker* **then**
            add $mp[s_i][p]$ in $V1$ to $mp[s_i][p]$ in $V2$
            enqueue local worker $q[w]$
          else
            send to worker j where w is stored
            add $mp[s_i][p]$ in $V1$ to $mp[s_i][p]$ in $V2$
            queue $q[w]$ in worker j
          end
        end
      end
    end
  end
end
return;

ceived node scores and get the global top-k score answer result for query.

4.2 Lifestyle-based Personalized Query

It is common in some situation, drug would have different side effect on persons who have different lifestyle. Based on the proposed general subgraph query above, we extend the the lifestyle factors to personalize this query.It is important to quantify the weight for different lifestyle. Here we consider exercise database as an initiative. Based on the extracted information about user exercise and drug-side effect information, we generally know whether user has mentioned an side-effect of drugs and nodes with exercise or not. Because of the information extracted are noisy and incomplete, we use different weight assignment to different situation.The drug graph is extended so that the existence of each edge is conditioned based on patient's lifestyle. We define a personalized weight $w_G(v_i, v_j, L)$ to represent the association strength between node v_i and v_j. the range of $w_G(v_i, v_j, L)$ is in [0,1]. In the general graph schema, the weights between any nodes are assumed to be 1. But here for personal query, It is quantified according to different lifestyle and we only consider the weight between Drug and Side Effect nodes. The lifestyle L is a collection of exercise, vegetarian, weight gain/loss, sleeping et.c Currently, we only have the exercise lifestyle data available.

Here there are two factors that affects the weight of node d_i and Side effect Se_i. One factor is the number of e information about lifestyle L and drug-side effects $Count(d_i, Se_i, L)$ extracted in the post, the other is whether the general drug-side effect exists. We assign the weight coefficient based on these two factors and get four different weight values in the Table 2

Table 2: weight assignment for drug-side edges

General v_i and v_j exists	$Count(v_i, v_j, L) > 0$	$w_G(v_i, v_j, L)$
True	True	1
True	False	0.75
False	True	0.25
False	False	0

We assign different confidence level to different situation, giving more level to the general drug graph.The association between drug and side effect is the strongest if both general drug-side effect exists and extracted drug-side effect for lifestyle exist. Considering the bias for extraction of drug-side effect with lifestyle, there could be better weight assignment to be explored, but our study is only to provide an initiative tool for this exploration.

Base on this measurement of lifestyle on drug-side effect, we need to modify the likelihood measurement $p'(v_i, v_j)$ in equation 2 when considering personalized query. If no personalized profile L is indicated in the input query: $p^{'}(v_i, v_j)$ is the same as the original one $p(v_i, v_j)$ If personalized profile L is not provided, $p^{'}(v_i, v_j)$ is the personalized weight-$w_G(v_i, v_j, L)$. The equation is as follows:

$$S(f) = \begin{cases} \prod_{e(q_i,q_j)\epsilon E_Q^+} p^{'}(f(q_i), f(q_j)) \cdot \prod_{e(q_i,q_j)\epsilon E_Q^-} (1 - p^{'}(f(q_i), f(q_j))) \\ \quad .... \ profile\ L\ is\ empty \\ \prod_{e(q_i,q_j)\epsilon E_Q^+} w_G(v_i, v_j, L) \cdot \prod_{e(q_i,q_j)\epsilon E_Q^-} (1 - w_G(v_i, v_j, L)) \\ \quad .... \ profile\ L\ is\ not\ empty \end{cases} \tag{3}$$

5. DISTRIBUTED IMPLEMENTATION

Since a single machine has limited memory and computation resources, we design a distributed system to store the web-scale information networks graph and execute our algorithm for drug query.

5.1 System Overview

Our system is implemented based on a distributed graph computation framework, Piccolo [**?**], which consists of one master process and multiple worker processes. Figure 4 shows the architecture of our system. In the system, the master coordinates the workers to perform metapath iteration calculation and the termination check of all algorithms. Distributed in-memory state table is also shown in Figure 4. For efficiency, the data graph is stored in an in-memory state table that is distributed across workers. Each data node corresponds to one row in the table. The row associated with a node v, row(v), stores the metadata of node v including the node name, the node type, and the neighbors keys. Additionally, row (v) contains a hash map that

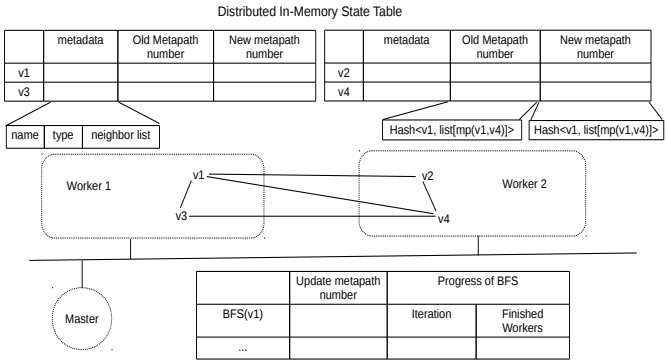


Figure 4: System Architecture.

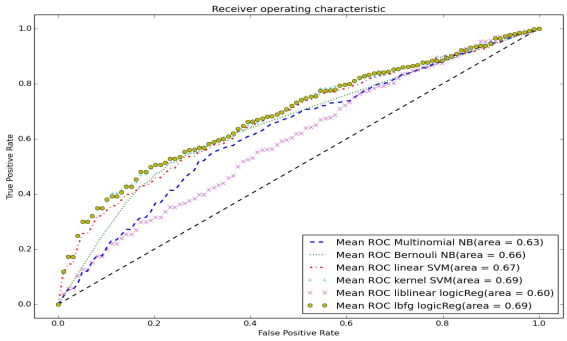


Figure 5: ROC with tf-idf,n-gram and topic model features.

stores the old(current)metapath values of different metapath types between the known nodes and unknown node v, and the new(next) one between them. Each worker stores a partition of the metapath values of different metapath types between the known nodes and unknown node v.Each worker stores a partition of the state table. In our implementation, we assign data nodes to every worker by applying a hash function to the node keys, but other partition strategies can also be applied. With the distributed in-memory state table, we next show how to perform the distributed metapath values calculation and the distributed termination check.

5.2 Distributed Metapath Calculation

For doing metapath calculation, a total $max(MP\_length)$*V_Q^S BFSes are needed in which $max(MP\_length)$ is 4 here. The complexity of metapath calculation is $O(V_Q^S * V^T)$. The multiple known nodes' BFS are performed simultaneously(from Algorithm 2). To do multiple BFSes, we use the strategy shown next.

Multiple BFSes. To implement multiple BFSes concurrently, we need to distinguish nodes visited for different known nodes and workers. We set a node and worker Id as a key. For example, there are common nodes that need to be visited in different worker for different known u and v. In the worker with Id 0,we designate key $u+0$, $v+0$. In the worker with Id 1, we designate key $u+1$, $v+1$ as key to differentiate the queue. Then we assign each worker a queue with the key. The queue stores the visited neighbor nodes in hash structure $< key, V^T >$ for next iteration. When one worker finishes its current iteration, it would update the new metapath number for the known source in distributed in-memory state table and send finishing notification to the master.

Expanding BFS. Each iteration of a BFS finishes when all workers have finished processing for that iteration. The master keeps track of these workers that have finished the iteration and notifies these workers to expand the next BFS when the iteration is completed.

5.3 Distributed Termination Check

We now discuss how to perform termination checks when the answer scores are distributed across the workers. We distribute the termination check workload among the workers. Each worker first finds its local top-k candidates (according to the BFS traversal level designated by master) and sends them to the master. The master finds the global top-k candidates among the local top-k candidates. Additionally, each worker is responsible for checking the top-k condition and calculate the local candidate scores on its local nodes. Using the results from the workers, the master determines whether the real top-k candidates are found.

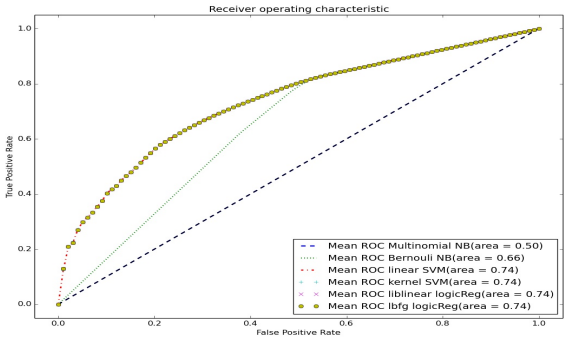


Figure 6: ROC with frequency-exercise and PRP-exercise pairs features.

6. EVALUATION

In our study, we have three parts of evaluations, we show the results of lifestyle extraction, distributed query system scalability, and the query answers based on lifestyle.

6.1 lifestyle extraction

In the lifestyle extraction, we use multiple classification methods and combine multiple features mentioned to do classification tasks. Nave bayes, logistic regression, SVM classfiers are used in our classification. We use nltk-tools [**?**] and scikit-learn packages [**?**] to help our implementation of the extraction.

we compare the different features and show the different result for the experiments (1)tf-idf, n-gram and topic modeling features result

(2)We consider only frequency-exercise and PRP-exercise pairs features to get the result in Figure 6

(3) Combiing all the features result shown in Figure 7

Comparing to these classification, We find use frequency-exercise and PRP-exercise pairs features to get the best

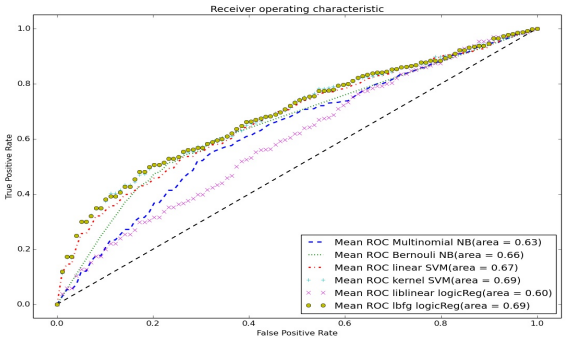


Figure 7: ROC with All the features combined.

performances,the ROC score and achieve 0.74 using SVM classification and logic regression. This is because the features designed for social media can be effectively reflect the complexity of text expression. However, the performance is needed to be improved in the future work.

6.2 distributed query system scalability

to measure the distributed query system scalability, we use these test environments. We have basic test input parameters: graph size is 1 million nodes, topK is 10 and query size is 20 sources nodes, worker number is 4. If we test one element, we will change one of its parameters, keep other parameters fixed.

Varying worker numbers

Figure 8a shows the scalability with the numbers of varying worker numbers in local cluster.The speedup ratio is 0.78 on average.

Varying top K number Figure 8b shows the scalability with the numbers of varying worker numbers in local cluster.

Varying query Size

Here we test the different numbers of given sources nodes. we are given different sources of numbers and one unknown drug node to query. Figure 9a shows with different query Size result

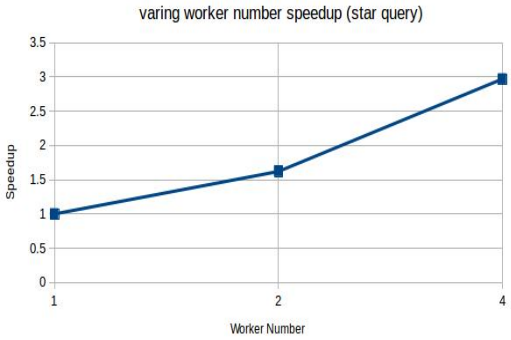
Varying graph Size

Figure 9b shows with different graph size increase.

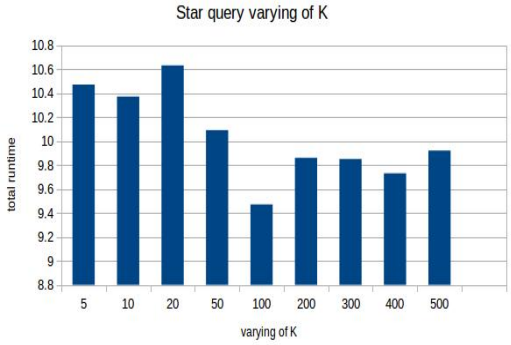
6.3 lifestyle query answering

In this section, we illustrate the usefulness of our query system by showing examples of the query results returned from our system and the personalized lifestyle results when considering lifestyle.

First, we show the top results for the query

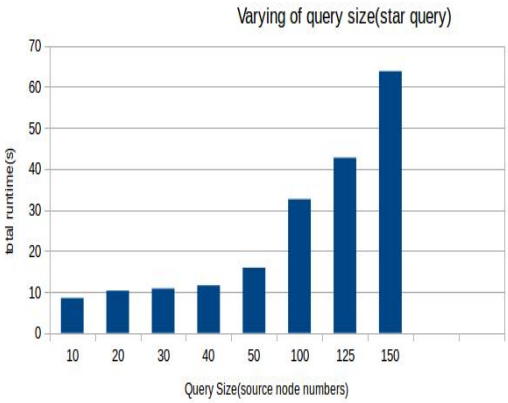


(a) Increasing worker number

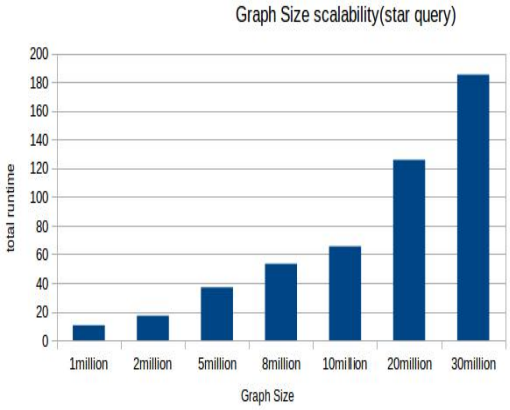


(b) Increasing K size

Figure 8: Query worker number and K size scalability



(a) Increasing query size number



(b) Increasing graph size

Figure 9: Query size and graph size scalability