Relative Difficulty Estimation in Community Answering Services

Student Name: Adesh Pandey, Deepak Thukral, Rishabh Gupta
Roll Number: 2014004, 2014036, 2014086

BTP report submitted in partial fulfillment of the requirements for the Degree of B.Tech. in Computer Science & Engineering on November 16, 2017

BTP Track: Research

BTP Advisor
Dr Vikram Goyal
Dr Tanmoy Chakraborty

Indraprastha Institute of Information Technology
New Delhi
Student’s Declaration

I hereby declare that the work presented in the report entitled "Relative Difficulty Estimation in Community Answering Services" submitted by me for the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science & Engineering at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under guidance of Dr Vikram Goyal and Dr Tanmoy Chakraborty. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

Place & Date: New Delhi, 16 November 2017

Adesh Pandey, Deepak Thukral
Rishabh Gupta

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Place & Date: New Delhi, 16 November 2017

Dr Vikram Goyal
Abstract

Given a question on a crowd sourced Q&A platform, we tried to gauge the difficulty of the question. Earlier, we tried various NLP based techniques for this purpose. Now, we tried various graph models in order to model our intuition of how difficulty is associated with questions, the answerers, the asker and how over time the difficulty of one’s questions change. We used various ranking techniques in order to compute the difficulty values of the questions in our graph model.

Keywords: Data Analysis, Stackoverflow, Graph Mining, Time-evolving networks
Acknowledgments

We would like to acknowledge the following

1. Dr Vikram Goyal, our advisor
2. Dr Tanmoy Chakraborty, our advisor

Work Distribution

1. Implementation and analyze [4] by Adesh, Deepak
2. Implement and analyze [3] by Rishabh
3. Basic model implementation by Adesh, Deepak, Rishabh
4. Temporal model implementation by Deepak
5. Advanced temporal model implementation by Adesh, Deepak, Rishabh
7. Work on Local directed search [9] by Rishabh
Contents

1 Introduction ................................................. 1

2 Literature Survey ........................................... 2

3 Dataset Collection ........................................ 3
   3.1 Test Set Generation ..................................... 3
   3.2 Accuracy ............................................. 3

4 Initial Approach ............................................ 4
   4.1 Categorization of questions ............................ 4
      4.1.1 Topic Modelling using LDA ....................... 4
      4.1.2 Word2Vec and Doc2Vec ........................... 4
      4.1.3 Naive Bayes ....................................... 5
   4.2 Binary classification .................................. 5
      4.2.1 Unigram Matching ................................ 5
      4.2.2 Bigram Matching ................................ 5
      4.2.3 Weighted Unigram Matching ..................... 5
      4.2.4 Multipass Approach ............................... 6

5 Evaluating Existing Models ............................... 7
   5.1 Question Difficulty Estimation in Community Question Answering Services ............................. 7
      5.1.1 Graph Modelling ................................... 7
      5.1.2 Algorithm ......................................... 8
      5.1.3 Results ........................................... 8
   5.2 Competing to Share Expertise: the Taskcn Knowledge Sharing Community ............................ 9
      5.2.1 Graph Modelling ................................... 9
      5.2.2 Algorithm and Results ............................. 9

6 Proposed Models ........................................... 11
   6.1 Basic Model ........................................... 11
Chapter 1

Introduction

In many community-based information web sites, such as Stack Overflow, users contribute content in the form of questions and answers, which allows others to learn through the contributions of the community.

**In our project we intend to use Stack Overflow as a tool to determine the difficulty of a given set of questions.**

We focused on questions related to java for research purposes. Last semester, we started our approach with NLP based techniques. We tried various classifiers like Word2Vec, Doc2Vec, Naive Bayes to classify the questions in 4 classes. Then, we switched to a hypothesis that if the answer is found in the corpus, it is an easy question. We tried many techniques from naive approach to more advanced approach for matching.

In this semester, we started relating asker of a question, answerer, best answerer and the asker’s own questions. We noticed several intuitions and modelled several graph models to capture the associated ideas. For example, a user’s questions’ difficulty should increase over time. This report presents these ideas and the associated results. For testing our models, we generated our own test set as described in one of the chapters. The models were tested by several techniques like PageRank, TrueSkill and link prediction.
Chapter 2

Literature Survey

PageRank
PageRank is a link analysis algorithm which works on graphs to give an importance of a node in the graph. It assigns a weight to each node of a graph which is directly proportional to its importance. This algorithm is often used to rank web pages of the internet where edges represent a hyperlink between two web pages. The following is the update equation ran iteratively. d is the damping factor (usually $d = 0.85$). $N(v)$ denotes neighbours of v.

$$PR(p_i) = \frac{1 - d}{N} + d \sum_{p_j \in N(p_i)} \frac{PR(p_j)}{\text{outdegree}(p_j)}$$  \hspace{1cm} (2.1)

HITS Algorithm
HITS is also a link analysis analysis on graph which is mainly used in context of web pages. It has hubs and authority scores for nodes which gets updated according to the updation rule. A good hub score represents a page that pointed to many other pages, and a good authority score represents a page that was linked by many different hubs.

The Authority score for node $p$ is updated as follows- (Node i is connected to node p)

$$\text{auth}(p) = \sum_{i=1}^{n} \text{hub}(i)$$  \hspace{1cm} (2.2)

The Hub score is updated as follows-

$$\text{hub}(p) = \sum_{i=1}^{n} \text{auth}(i)$$  \hspace{1cm} (2.3)

TrueSkill
TrueSkill [5] is a Bayesian skill rating model that is developed for estimating the relative skill levels of players in games. It assumes that players skill is a normal distribution with mean $m$ and variance $v$ and are assigned some value at the beginning. The mean is updated as the game proceeds and it increases after a win and decreases after a loss. The amount of update will depend on its rating(mean) at the time of updating.
Chapter 3

Dataset Collection

We chose java as the topic for research purpose. We picked StackOverflow as the Collection of Dataset.
We extracted 1,00,000 questions related to java between the time frame 2008-08-01 to 2010-12-08.
The answers for all the above question were captured upto 2011-05-12.

3.1 Test Set Generation

• We randomly selected pairs of questions from the dataset
• All three of us independently and manually annotated the pair to mark the more difficult question
• If all three annotated the same question as more difficult in the pair, we keep it as a ground truth AKA test set otherwise we reject the pair.

We finally created our test set consisting of 250 pairs.

3.2 Accuracy

For testing, we predict the more difficult question for each pair in the test set by our algorithm.
We define accuracy as follows:

\[
\text{Accuracy} = \frac{\#\text{correct predictions}}{\#\text{pairs}}
\]  

(3.1)
Chapter 4

Initial Approach

The earlier approach we followed to predict difficulty of questions was NLP based. We analyzed text of the questions of SO and predicted the difficulty. This didn’t use the attached metadata. The approach and techniques have been briefly listed below.

4.1 Categorization of questions

We applied topic modelling using LDA, Word2Vec, Doc2Vec and Naive Bayes on the Dataset of questions to find out the categories as per [6].

4.1.1 Topic Modelling using LDA

LDA was applied on the questions of StackOverflow to divide the text into topics. Top few words of each topic were then examined as shown in the results. The main parameter to the model was number of topics. We kept no of topics equal to 4 initially as discussed in the paper [6].

4.1.2 Word2Vec and Doc2Vec

We consulted the paper Ranking Crowd Knowledge to Assist Software Development [1] by Lucas B. L. de Souza et al which deals with analyzing the information on SO to help developers improve their documentation. In this paper they did categorization of questions into four categories from where we discovered the the top word corresponding to the each category. To use word2vec in this, We need to set the context window which determines how many words before and after a given word would be included as context words of the given word as one of the important parameters. We ran Word2Vec for context window size of 10. Then we found the most similar words from the model trained corresponding to the top words found above. The vectors obtained can be used to find a word similar to a given word.
4.1.3 Naive Bayes

Since Naive Bayes is a supervised algorithm we needed labelled data for training and testing. So we used the crawled 20,000 dataset to separate out questions of 4 categories based on some of the common keywords found in the above results (Word2Vec and Doc2Vec). We had around 250 questions for each category. Dataset was divided as 80 percent as training data and 20 percent as testing data.

4.2 Binary classification

Hypothesis: If the answer of a question is easily or directly found in a corpus related to the same topic, the question is easy.

This was the initial hypothesis. We applied various techniques to test this hypothesis. Although, as discovered later on, if the answer is spread out in the corpus it is relatively difficult than the question whose answer is present in close proximity in the corpus. In other words more local the answer in the corpus more easily it is.

4.2.1 Unigram Matching

1. Apply Lemmatization and Stop words removal on the answers to obtain only relevant keywords.

2. Taking a window of 100 lines in the corpus, search for the keywords found above and give a score based on the number of words matched.

4.2.2 Bigram Matching

The approach is similar to unigram matching.

1. Apply Lemmatization and Stop words removal on the answers to obtain only relevant bigrams.

2. Taking a window of 100 lines in the corpus, search for the bigrams found above and give a score based on the number of bigrams matched.

4.2.3 Weighted Unigram Matching

To solve above problem we needed to weight different keywords based on the importance of the word. So we came up with very famous approach of TF-IDF. TF-IDF reflects how important a word is to a document in a collection or corpus.

1. Apply Lemmatization and Stop words removal on the answers to obtain only relevant words.
2. The TF-IDF score for words obtained from step-1 was calculated using the corpus (Stack-Overflow) and stored as a weighted dictionary.

3. Then we applied sliding window matching for window size of 100 lines to search all the Java books and found various windows that contained the keywords.

4. A score was calculated for each matching which was sum of the TF-IDF score of the matched words. The best score was given the best matching.

4.2.4 Multipass Approach

This approach is useful in giving us an absolute difficulty value which enabled us to rank the question according to their difficulty.

1. In the first part, we found out the window in the corpus that matches the most with the answer (i.e, has the maximum sum of scores of the unigrams) and get the maximum possible score (say M). The score is the sum of the tf-idf of the matched words.

2. Now we set a new threshold $T = M/2$

3. In the second part, count the number of windows whose score $\geq T$.

4. We tried different thresholds but $T = M/2$ worked best among all.
Chapter 5

Evaluating Existing Models

We started with the reading some papers and research work done in the area of finding difficulty in community question answer based platforms. We found two relevant papers which are described below.

5.1 Question Difficulty Estimation in Community Question Answering Services

The paper [4] aims to estimate the question difficulty in community question answering services. They used a competition-based model for estimating question difficulty by doing pairwise comparisons (competitions) between questions and users.

5.1.1 Graph Modelling

A user graph is created. Let question q be considered as a pseudo user $u_q$

1. One competition between pseudo user $u_q$ and asker $u_a$,

2. One competition between pseudo user $u_q$ and the best answerer $u_b$

3. One competition between the best answerer $u_b$ and asker $u_a$

4. $S$ Competitions between the best answerer $u_b$ and all non-best answers where $S$ is the no of non-best answers of the question q.

Results of above competition

Pseudo user $u_q$ wins the first competition
The best answerer $u_b$ wins all remaining $(S + 2)$ competitions
5.1.2 Algorithm

The problem of estimating question difficulty score (and user expertise score) is cast as a problem of learning the relative skills of players from the win-loss results of the generated two player competitions. For user, its score indicates expertise. For pseudo user, its score indicates difficulty of the question. They used a tool TrueSkill [5] which is essentially a skill based ranking tool which uses Bayesian model to learn skill of the player. Once all the competition pairs are constructed, each pair is sent to compete. The detailed algorithm is as follows:

All players are initialized with an initial and equal mean and variance. Now, all the competition pairs are sent for competition and the mean and variance of each player is updated accordingly as follows:

1. If in a new competition, the player with higher mean (skill level) wins the game, it will cause small update in skill level mean and uncertainty of the players.

2. If in a new competition, the player with lower mean (skill level) wins the game, it will cause large update in skill level mean and uncertainty of the players;

Prediction

For two questions $q_1$ and $q_2$ if mean of $q_1$ (skill level of pseudo user) is greater than mean of $q_2$ (skill level of pseudo user) then we predict that $q_1$ is more difficult than $q_2$.

5.1.3 Results

We formulated the above model on our dataset and tested it on our test set. We got an accuracy of 56.02%. We also tried two other algorithms on the edges we created using the above algorithm.
Algorithm Accuracy on our Test Set

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrueSkill</td>
<td>56.02 %</td>
</tr>
<tr>
<td>PageRank</td>
<td>54.47 %</td>
</tr>
<tr>
<td>HITS</td>
<td>52.87 %</td>
</tr>
</tbody>
</table>

5.2 Competing to Share Expertise: the Taskcn Knowledge Sharing Community

The paper [3] analyses one of the biggest Witkey websites in China, Taskcn.com. They applied social network prestige measures to a user and task networks based on competitive outcomes between them and discover the underlying properties of both users and task.

5.2.1 Graph Modelling

A question graph is created.

1. Suppose users A and B participate in the same task. If A wins in that task, then an edge is added from B to A.

2. If users participate in tasks X and Y. If a user A wins in task X but fails in task Y, then a edge is built from X to Y. This implies that task Y is more prestigious than task X.

Graph Modelling on StackOverflow Data

Let $q_1$ and $q_2$ be two questions and u be an user who answered in both questions. Then an edge $(q_1, q_2)$ is created when when u answers to $q_2$ is the best and accepted answer whereas its answers to $q_2$ is an non-best answer. This implies that question $q_2$ is likely to be more difficult than question $q_1$.

5.2.2 Algorithm and Results

Algorithms

Page Rank Algorithm was applied on the graph created using above edges and Page Rank score was obtained for each question.

Prediction:

For two questions $q_1$ and $q_2$ if Page Rank score of $q_1$ is greater than Page rank score of $q_2$ then we predict that $q_1$ is more difficult than $q_2$. 
Results

The results for PageRank and TrueSkill on the above graph modelling are tabulated below.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy on our Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>PageRank</td>
<td>47.31 %</td>
</tr>
<tr>
<td>TrueSkill</td>
<td>44.50 %</td>
</tr>
</tbody>
</table>
Chapter 6

Proposed Models

6.1 Basic Model

6.1.1 Model

We propose a graph modelling whose basic idea is, if a user u gives an accepted answer to a question Q, then the questions asked by the user u would be more difficult than Q.

Understanding using Example
Let user A asks a question Q on StackOverflow. Let B is the user whose answer is accepted. If accepted answer does not exist then B could be user who has maximum upvotes to downvotes ratio.

Now we can make some relationship between Questions of A and B.

All questions asked by B = q₁, q₂, q₃, ... will be more difficult than Q (Question of user A). This is because B responded to question asked by A hence indicating a higher expertise of user B, hence we can say with confidence that questions asked by B will be more difficult than questions asked by A.

This is illustrated using Figure 6.1.

6.1.2 Results

We applied PageRank, HITS and TrueSkill on the graph created above. In each case the question with higher score was predicted to be more difficult.
6.2 Temporal Model

6.2.1 Model

The temporal view is of huge importance in this type of setting, i.e., a trivial question posted by an expert user long time ago does not mean it is difficult.

For incorporating it, first we are dividing questions bi-weekly, i.e. all questions belonging to one interval would be assumed to be posted at same time. [1] states that bi-weekly buckets are best and preserves space. [8] find three parameters capturing temporal significance. We leverage these findings to creating the model.

First, we created a map structure, which corresponds each node with a bucket-number. From the graph created using the rule (stated at section 2), we create a second graph but pick only those edges which are from a node of lower bucket number to a node having higher bucket number.

Consider a directed graph, $G = \{q_1, q_2, ... q_n\}$, having buckets $\{t_1, t_2, ... t_n\}$, where $q_1, ... q_n$ represent the questions and $t_1, ... t_n$ denote the corresponding bucket number when the question was posted.

We use the following three parameters as [8] to compute the score for temporal graph:

**Time-Length Factor**

Consider an edge $(q_1, q_2)$ in graph, i.e. $q_1$ is at lower bucket number while $q_2$ is at higher bucket number, then greater the time difference between nodes, there are more chances of latter being more difficult.
The edge \((q_1, q_2)\) where \(q_1\) is posted at bucket number \(t_1\), while \(q_2\) is posted at \(t_2\) (bigger than \(t_1\)) is attributed an edge weight as described:

\[
score_1 = 1 - e^{-(t_2-t_1)} \tag{6.1}
\]

As the difference increases, edge weight also increases signifying more hardness of question \(q_2\).

**Frequency Factor**

This factor attributes activeness and regularity of the asker of question to the edge weight. More active and regular he is, more effect it has on edge weight. If a seasoned person asks a question and other person answers it, then there is high probability that other guy would have asked even more difficult questions. We say a guy is seasoned if he asks or answers regularly.

Let \(N\) be number of questions or answers posted by asker, \(N_t\) be number of questions or answers posted by asker at bucket \(t\), and number of buckets be \(N_b\).

\[
Avg = \frac{N}{N_b} \tag{6.2}
\]

\[
Var = \sum_{i=1}^{N_b} (N_i - Avg)^2 \tag{6.3}
\]

\[
score_2 = e^{-Var} \tag{6.4}
\]

Smaller the variance, better effect it has on edge weight.

**Note:** Here for edge \((u,v)\), we only see Frequency Factor of user who posted \(u\) as \(u\) is pointing to \(v\).

**Acceptance Factor**

The time difference between posting of question and answer (which got accepted by asker) determines the hardness of question. If a question got answered within few hours, while other question got answered in few days (indicating it required decent amount of work or research), then the questions to which latter would be pointing may be more difficult.

Let \(t_1\) be time when question was posted and \(t_2\) be time when it was answered (by best answerer).

\[
score_3 = 1 - e^{-(t_2-t_1)} \tag{6.5}
\]

As the difference increases, the edge weight too.

**Note:** Here for edge \((u,v)\), we only see acceptance factor of \(u\) as it is pointing to \(v\).
Score of Temporal Model

All the three scores computed above are merged to obtain one final edge weight.

\[ Weight_{12} = (\alpha \ast score_1) + (\beta \ast score_2) + ((1 - \alpha - \beta) \ast score_3) \]  

(6.6)

\(\alpha\) and \(\beta\) are chosen based on experiment (choosing different values, and selecting the one which gives the best result).

To take the impact of temporal view on the given question, run PageRank on weighted graph and compute score of each node, in \(\{score_{11}, score_{12}, .. score_{1n}\}\), where \(score_{1i}\) is the score of node \(i\).

Combining Two Models

Let \(score_{1i}\) be score of a node (question) computed using normal PageRank (as stated in 2), and \(score_{2i}\) be score of node taking temporal view into account.

Final Score of each node, signifying the hardness of question:

\[(\alpha \ast score_{1i}) + ((1 - \alpha) \ast score_{2i})\]  

(6.7)

\(\alpha\) is chosen based on experiment (choosing different values, and selecting the one which gives the best result).

6.2.2 Results

The accuracy obtained using this model was 55.4%.

6.3 Advanced Temporal Model

The basic model we used above had one issue that it was incorporating all the edges, i.e. if a user, \(u_1\) answered \(u_2\) then all questions asked by user \(u_2\) would be difficult even if \(u_2\) developed proficiency in skill (say java, required to answer \(u_1\)) from the timestamp he answered \(u_1\). To overcome this problem, we came up with a structure which takes care of back edges and temporal edges efficiently.

6.3.1 Model

As previously, we have bucket length of 14 days (2 weeks). Consider a directed graph, \(G = \{q_1, q_2, .. q_n\}\), having buckets \(\{t_1, t_2, .. t_n\}\), where \(q_1,..q_n\) represent the questions and \(t_1,..t_n\)
denote the corresponding bucket number when the question was posted. In the current model, we have three types of edges:

1. Window Defined Edges: These are back edges, and here we enforce a condition that for back edges to exist, the absolute difference between timestamps should be $<=$ 1. The basic premise of doing this is to ensure that questions asked much before the current question shouldn’t be regarded as more difficult.

2. User Defined Edges: The questions asked by same user are ordered by difficulty with respect to timestamp, that is question, q would be having edge to nearest q2 (in case of tie, i.e. $(q, q3)$ edge would be included too if q3 is also asked at same timestamp as q, all edges would be included) asked by same user, and $timestamp_{q2} > timestamp_{q}$.

3. Temporal edges: These are edges from a current question, q to q2 when q2’s asker answered q and q2 was posted in $timestamp_{q2} >= timestamp_{q}$.

6.3.2 Algorithm

We used PageRank to compute score of each node. However, we modified initial weight in PageRank to incorporate weight of node, which is:

Let user who asked question q be $u_1$ and his reputation be $r$ and max reputation of a user in our Stack Overflow dataset be $r_m$. Then, we normalize the reputation by, $i_{mod}$: $r/r_m$.

So, initial weight:

$$PR(p_i) = (1 - d) * i_{mod} + d \sum_{p_j \in N(p_i)} \frac{PR(p_j)}{\text{outdegree}(p_j)}$$

(6.8)

6.3.3 Results

We applied PageRank, HITS and TrueSkill on the graph created above. In each case the question with higher score was predicted to be more difficult.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy on our Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrueSkill</td>
<td>56.28 %</td>
</tr>
<tr>
<td>HITS</td>
<td>43.97 %</td>
</tr>
<tr>
<td>PageRank</td>
<td>61.25 %</td>
</tr>
</tbody>
</table>
Chapter 7

Link Prediction Approach

Up until now, we have been mainly relying on PageRank for finding ranking to compute relative difficulty. We now turned to another domain of link prediction. The objective in link prediction is, given an incomplete graph we need to predict if an edge exists between two nodes.

7.1 Recursive subgraph-based ranking

The paper [2] uses a ranking based approach to predict links in a graph. They combine local and global indicators to construct a ranking. Then, any node ranked higher (less important) is predicted to have an edge to lower ranked nodes (more important).

Algorithm

recursive_ranking(V, E):

1. Sort the graph in descending order of \( \Delta D = \text{indegree}(v) - \text{outdegree}(v) \). The index then denotes the rank.
2. if \( \|V\| \leq \frac{1}{\alpha} \): Return ranks
3. \( V_L = \{ j : j \in V, rank(j) \leq \alpha \|V\| \} \)
4. \( V_F = V - V_L \)
5. Compute induced graph edges \( E_F \) by \( V_F \) and \( E_L \) by \( V_L \)
6. \( rank_L = \text{recursive\_ranking}(V_L, E_L) \)
7. \( rank_F = \text{recursive\_ranking}(V_F, E_F) \)
8. Increase ranks in \( rank_F \) by \( \|V_L\| \)
9. rank = \( rank_L \cup rank_F \)
10. return rank

Results

The rankings obtained by the algorithm is used to imply edge from any higher ranked (less important) to lower ranked (more important). This is then employed for testing on our dataset. Upon trying out our proposed temporal model with edge prediction by the above model, the accuracy achieved is 60.65%.

7.2 Local directed path

The paper [9] proposes an efficient solution named Local Directed Path to predict link direction. Given a graph with sufficient number of directed edges, Local Directed path Algorithm finds a direction of edge between nodes where there is no edge in original given graph. The key point is that it adds a ground node in the graph. This ground node is connected to all nodes and all node is connected to ground node by directed links. The effect of ground node is to add an additional relation between nodes that have no common nodes, which can naturally release the cold-start problem.

Algorithm

Local Directed Path Algorithm

For predicting the direction of edge between i and j do the following.

Let $LDP_{i\rightarrow j} = \text{paths(2)}_{i\rightarrow j} + \text{paths(3)}_{i\rightarrow j} + \text{paths}^{GN}(3)_{i\rightarrow j}$

where

$\text{paths(2)}_{i\rightarrow j} - \text{no of paths between i and j of length 2}$

$\text{paths(3)}_{i\rightarrow j} - \text{no of paths between i and j of length 3}$

$\text{paths}^{GN}(3)_{i\rightarrow j} - \text{no of paths between i and j of length 3 via the ground node}$

$\text{paths}^{GN}(3)_{i\rightarrow j} = \text{OutDegree}_i + \text{InDegree}_j + \text{paths(2)}_{i\rightarrow j}$

1. Compute $LDP_{i\rightarrow j}$.

2. Compute $LDP_{j\rightarrow i}$.

3. if $LDP_{i\rightarrow j} > LDP_{j\rightarrow i}$ then we have an edge (i→j)

4. if $LDP_{i\rightarrow j} < LDP_{j\rightarrow i}$ we have an edge (j→i)

5. In case of equal value random edge direction is choosen.
Results

The paper has used 2 length and 3 length path to accommodate the notion of Local path. Upon trying out the proposed temporal model with edge prediction by the above model, we could not find even a single path of length 2 or 3 for any of our 250 test pairs. We tried increasing the path length to 4, 5 and 6. In all cases we still could find only 4 paths in one test pair. Increasing it to something more reasonable like 8 or anything above was computationally infeasible for a huge graph (our model contains 400k edges) and also going for path length of 8 or above reduces the notion of local path discussed in the paper.
Chapter 8

Conclusion and Future Work

8.1 Future Work

We plan to analyze the technique in the paper [7] which uses supervised learning problem in which the directions of some edges are known. An MST of data is built from training set (90% of edges), and two graphs, directed and undirected (of training and test set) are created and few properties are calculated using it to predict the direction of link. Using these features, SVM will be built to predict the output.

We plan to further analyze link predictions and incorporate it for difficulty computation. We also plan to extend the model to use deep learning techniques.

8.2 Conclusion

We proposed three algorithms and two of them beat the state of the art algorithms. The best algorithm [4] was 56.02% accurate. Our basic model was 56.56% accurate. Our proposed advanced temporal model raised the accuracy upto 61.25%.
Bibliography


[3] Jiang Yang1, Lada A. Adamic, M. S. Competing to share expertise: the taskcn knowledge sharing community.


