RELATIONAL SOCIAL MEDIA SEARCH ENGINE

by

Indervir Singh Banipal

APPROVED BY SUPERVISORY COMMITTEE:

Dr. Latifur Khan, Chair

Dr. Alvaro A. Cardenas

Dr. Weili Wu
RELATIONAL SOCIAL MEDIA SEARCH ENGINE

by

INDERVIR SINGH BANIPAL, BS

THESIS
Presented to the Faculty of
The University of Texas at Dallas
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN
COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS
December 2016
ACKNOWLEDGMENTS

I sincerely thank Prof. Latifur Khan for his continuous mentorship, guidance and support throughout the research experience. I also thank the Head of the Department Prof. Gopal Gupta, who has always been a source of encouragement.

I thank all my lab mates at Data Management and Data Mining Lab, and Semantic Web Lab. It is my pleasure to thank my colleagues with whom I worked during my Research Assistantship: Vishal Karande, Bikramjit Mandal and Brian Ricks. I thank my colleagues at IBM Watson who encouraged me to pursue research in this area. I thank my friends who have been an important part of my journey during my graduate studies: Vineet Dhanawat, Dilpreet Singh and Vibha Belavadi.

Last but not the least, I thank my father Dr. Tarlok Singh Banipal, my mother Dr. Parampaul Kaur Banipal and my brother Sub Lt. Karanvir Singh Banipal without whom this journey would not have been possible.

November 2016
Given a collection of names of some people who are related to each other by a common relation, our goal is to find their profiles from social networks/media. For instance, they might belong to a particular class from a particular university. The problem poses severe challenges as there might be numerous people present in social media with the same name. The goal is to propose novel algorithms which will help us extracting the correct set of people from enormous possibilities.

Social networks have become a primary source of communication and the number one choice for people to stay in touch with each other. Apart from the use of social networks in private life, people rely heavily on the social networks for their professional lives as well. We exploit this fact and use social networks to find the relevant people we are looking for. Facebook [11], LinkedIn [12] and Twitter [13] are the most widely used social networks, for staying in touch with others privately, as well as professionally. Our approaches are applicable to any of these major social networks although we focus on Facebook.

We go through the chosen social network and search for the names in the collection. Every name in the input collection should be searched in the social network and its results are stored. After the searching through the social network, we will have a list of search results for each of the names in the collection. For each user, the social network has suggested multiple possible social network profiles which may or may not contain the target user which we are
looking for. We record all such search results for each user and this serves as our baseline. The goal is to improve this baseline/non-verified list by augmenting with semi-automated method to prune some irrelevant profiles and find the relevant one. We will discuss about social media crawler which we developed for extracting the information from the social media. We will also talk about the algorithms which we designed and implemented for showing the relevant profiles to the user. We will also talk about the human intervention in the form of feedback and finally, we present a complete generic framework for solving this problem.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>ACKNOWLEDGMENTS</th>
<th>iv</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>viii</td>
</tr>
<tr>
<td>CHAPTER 1    INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Problem Context</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Thesis Statement</td>
<td>6</td>
</tr>
<tr>
<td>1.3 Problem Description</td>
<td>8</td>
</tr>
<tr>
<td>1.4 Motivation</td>
<td>10</td>
</tr>
<tr>
<td>1.5 Document Organization</td>
<td>14</td>
</tr>
<tr>
<td>CHAPTER 2   DATA EXTRACTION FROM SOCIAL NETWORK</td>
<td>16</td>
</tr>
<tr>
<td>2.1 Formulating the Problem</td>
<td>16</td>
</tr>
<tr>
<td>2.2 Knowledge of the Relation(s)</td>
<td>19</td>
</tr>
<tr>
<td>2.3 Searching the Social Network</td>
<td>22</td>
</tr>
<tr>
<td>2.4 Data Storage and Management</td>
<td>23</td>
</tr>
<tr>
<td>2.5 Challenges while Extracting the Data</td>
<td>25</td>
</tr>
<tr>
<td>CHAPTER 3   ALGORITHMS</td>
<td>26</td>
</tr>
<tr>
<td>3.1 Social Media Search</td>
<td>26</td>
</tr>
<tr>
<td>3.2 Social Media Crawler</td>
<td>30</td>
</tr>
<tr>
<td>3.3 Social Graph Algorithm</td>
<td>36</td>
</tr>
<tr>
<td>3.4 Specifying Ground Truth</td>
<td>41</td>
</tr>
<tr>
<td>3.5 Combining Social Graph and Ground Truth</td>
<td>44</td>
</tr>
<tr>
<td>3.6 Seed’s Friends Algorithm</td>
<td>47</td>
</tr>
<tr>
<td>3.7 Filtering using Machine Learning</td>
<td>49</td>
</tr>
<tr>
<td>CHAPTER 4   RESULTS &amp; DISCUSSION</td>
<td>53</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>55</td>
</tr>
<tr>
<td>VITA</td>
<td>57</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

1.1 LinkedIn Search  ......................................................... 2
1.2 Relation based Search: LinkedIn University Search .............. 3
1.3 Relation based Search: Facebook Search  .......................... 5
1.4 Relation based Search: Facebook Search  .......................... 5
1.5 High School/College Re-unions  ...................................... 11
1.6 Connecting to people during disasters  .............................. 11
1.7 Detecting criminal gangs  ............................................. 12
1.8 Finding lost friends  .................................................... 12
1.9 Deducing relationships  ............................................... 13
2.1 Working flow for Searching Related People ....................... 22
2.2 Advantages of using MongoDB (NoSQL) database [20] .......... 24
3.1 Search Results (selected few) for 'mahmudul hasan buet' ....... 28
3.2 Big Data of the Social Media  ........................................ 28
3.3 Python’s Mechanize library [23] ..................................... 31
3.4 Social Graph  .......................................................... 38
3.5 Relevant relationships get highlighted while others are discarded [24] .......................... 40
3.6 Candidate profiles presented to the user for 'Manoj Kumar Roy' . . . . . . . . . . 43
3.7 Online website portal showing the verified list ................. 43
3.8 Search results for 'abdul halim buet'  .............................. 46
3.9 Search results for 'asif mahmud buet'  .............................. 46
3.10 Connections with verified list ....................................... 47
3.11 Find no-result-users using Seed’s Friends Algorithm ......... 49
3.12 IBM Watson Visual Recognition  .................................... 51
3.13 Example (reject) ..................................................... 52
3.14 Example (accept) ..................................................... 52
4.1 Verified list improving with time  .................................. 54
CHAPTER 1
INTRODUCTION

1.1 Problem Context

With the advent of social media and social networks, people have started using them as their primary source of communication. Facebook [11], LinkedIn [12] and Twitter [13] are some of the prominent social networks which people use to stay in touch with each other. Almost all of the prominent social networks offer the feature to search other people. This is one of the reasons why people have resorted to social media/networks as one of the primary sources of communication. One may need to connect with someone from his/her childhood, high school, graduate school, previous work place etc. Or someone may be looking for a person with specific skill set or background for professional purposes. People have the freedom to chose whom they want to connect and stay in touch with. Social networks play a very important role in representing the people on Internet in the form of social network profiles unless he/she is a very famous person. This representation in the form of social network profiles enable others to look for the target person provided all of them are on the same social network. This is not a very strict condition as in some cases, we may be able to look for people without joining the social network because we have access to powerful search engines like Google [14], Bing [15] etc. We may be able to find someone’s social network profile using Google but we may not be able to access detailed information about him/her unless we join the social network. Whether we can find someone’s social network profile completely, partially or not at all, depends on the security features offered by the social networking site and the person who is represented by the social network profile.

The most important reason why people have resorted to social networks is that they can no longer need a key/token to search about other people. In other modes of communication, that key can be an email id or a phone number. Social media and social networks have given
the ability to the people to communicate with anyone they want, yet providing them the freedom to choose the levels of security they want for their profiles. This is one of the major differentiating factor of this mode of communication. The ability to search someone using a particular social network depends a lot on the features provided by the search functionality of that particular social network. Additionally, the quality of the search of a social network is a very important factor when a user is deciding which social network to use for staying in touch with friends or relatives.

Generally, when someone wants to search about some specific person, he/she uses the search functionality provided by the social networking site. For example, LinkedIn has a very well built search functionality which helps finding the target users with different types of relational information which a user can provide to track down the correct user. A search query on the person named John Smith will result in numerous profiles. But LinkedIn has a very mature search functionality which makes this easier for the user. When we first visit the LinkedIn search bar, we are offered the following options:

![LinkedIn Search](image)

**Figure 1.1. LinkedIn Search**

As we can see, LinkedIn provides us multiple options to narrow down our search. As this is a social network, the topmost search option is 'People'. As this is a professional social
network, user would want to look for other people who are relevant to his/her search criteria for professional purposes. LinkedIn is a professional social network, so the second search option should be ‘Jobs’. We can also see the option of searching educational institutions under the option of Universities. Basically these are various options which we can select and then form a relation which will narrow down our search query. Consider the case of Universities section which helps us forming the relation using the information or knowledge about someone’s graduate school.

Suppose we select the option with Universities written on it and we search for Birla Institute of Technology and Science [16], and then select ‘Students and Alumni’ section:

![Figure 1.2. Relation based Search: LinkedIn University Search](image)

As we see in the above image, we get various options to build a relation and then search for the relevant users. The relation can be the location where the target person is present, or
the company/firm where the target person works, or the industry where the target person is working in. Based on these relations, the user can narrow down the search results to considerably low number. When we use the term relation in this thesis, we mean the piece of information about our search query which can narrow down our search results. For instance, in the above image, we specify the following relations to our LinkedIn University Search:

\[ R_a: \] Went to Birla Institute of Technology and Science.

\[ R_b: \] Lives in San Francisco Bay Area.

\[ R_c: \] Works in Engineering or is an Engineer.

\[ R_d: \] Is a researcher or works in a research profile.

With the application of all the above relations, the search results become very narrow. If we are looking for a person named Indervir who went to Birla Institute of Technology & Science, lives in Bay Area, and is a research engineer, we will get very few results or maybe only one search result which will be the correct Indervir.

Let us see how Facebook search works. Facebook is one of the largest social networks used by people, and has around 1.65 billion monthly active users. Facebook provides a unique way of searching where the user can specify very user-friendly queries similar to the way we communicate. This is called Facebook Graph Search [25]. Graph Search operated by use of a search algorithm similar to traditional search engines such as Google. However, the search feature is distinguished as a semantic search engine, searching based on intended meaning. Rather than returning results based on matching keywords, the search engine is designed to match phrases, as well as objects on the site [26].

If we want to find the people named John Smith who went to Harvard University, we execute the following query:
We get the following search results:

By default, Facebook Search shows us the top results from all the possible categories: Top, Latest, People, Photos, Videos, Shop, Pages, Places, Groups, Apps, Events. We are interested in people named John Smith who attended Harvard, so we can select People and see the candidate social network profiles for the real John Smith which we are looking for.

As we see, Facebook (like LinkedIn) takes the names of the people as input and lets us define the relations, and then based on the knowledge of the type of relation they have, suggests us numerous candidate social network profiles. But none of them takes a group
of people as input (who are related to each other through some relation), and then fetches results which most likely contain that particular solution set which contains all the correct social network profiles for all the names in the input collection.

We can say that even Facebook Graph Search, which is one of the most advanced social media search tools, does not provide a mechanism to search a group of related people.

What if we do not know about these relations? What if we just have a bunch of names (let us say William Grant, Julia Smith, Aston Miller etc.) who went to University of Texas and we want to search about them? In this type of scenario, we can definitely fetch a search result list of people named John Smith who went to University of Texas using social media. But from this list, we want to look for those people named John Smith who went to college with William, Julia and Aston in 1994 and not those people named John Smith who went to college during some other time frame say 1970s or say 1980s where they did not have William, Julia and Aston as their batchmates. We need to identify these results which are related to William, Julia, Aston etc. and just not any person named John Smith who went to University of Texas.

Our aim is to build algorithms which can help us search this input collection of names using social networks. In the coming sections, we will see how we formulate the problem, build tools to extract the required information from the social network(s), and then implement the algorithms designed by us. We will test them on a dataset which contain names of people who attended a specific college and then publish the results. We will also provide a conclusion talking about the performance of our framework.

1.2 Thesis Statement

The thesis aims at solving the problem where we want to find out certain group of individuals from social networks where we are given the names of those individuals, and they are related
to each other through some common relation. We develop some algorithms which we will be
talking about in the upcoming chapters.

Generally, in all social networks, as explained in the first section, social networks provide
the raw ability to search for users by using their names and then pruning the results using
various information like education, location, workplace etc. But none of them have worked
on to search for a group of names of some people who are related to each other in some way
and then show the correct results from numerous different possibilities.

Even if we do not have any information about the relationships between the names in the
input collection, our work attempts to establish the best possible relations between them and
suggest the most accurate set of people which the user might be interested in.

Our research work aims to address the following problems:

a) How to search group of people from social media who are related to each other?

b) How to search group of people from social media who are related to each other but
we do not know about the type of relationship which exists between them? *This can
also be called as investigative relationship search.*

c) Given a group of names, what is/are the type of relation(s) which exist(s) between
them?

We address these problems by inventing a series of algorithms which take into account
many different features of the social network profiles before suggesting which is the correct
social network profile for a particular person in the input collection.

We exploit following data to establish relationships between the users which helps us
finding the correct social network profiles for the input collection:

a) Social Network Graph [18]: helps us understanding how a person is connected to
others.
b) User input in the form of Ground Truth to identify the relationship we want to use (like college name).

c) Connectivity with the verified list (starting point generated from human feedback).

d) Media posted or projected by a person in the form of profile pictures etc.

e) Connectivity of a person with well-connected people.

We will also spend some time discussing how we can use the pictures of the people which they have put on the social network profiles to extract important features which can be used to predict the correct social network profiles.

1.3 Problem Description

With the recent advances in search technologies, it has become easy to search for a particular person provided we know certain key attributes about him/her which distinguish him/her from the other people with the same name. Suppose we want to search for a particular person named John Smith, we cannot simply search his name in the social network and get the correct John Smith in the first go. We would get a lot of search results which claim to be the target user but we need to provide further information to narrow down to the correct one. Let us say that we also know that John Smith attended University of Texas. So if we again search for John Smith but this time with the knowledge if the information that he attended University of Texas, we will narrow down to a comparatively less number of people named John Smith. We still we do not have the correct John Smith but we have fewer candidate profiles now. Similar to the relation through common education, other relations can be of many forms like:

\[ R_a: \text{Graduate School name.} \]
$R_b$: Time frame of attending the graduate school.

$R_c$: Combination of city, state, country where an individual lives.

No doubt we have reduced the number of social network profiles which claim to be the real John Smith, we are still not sure about which one of them is the correct John Smith. We have applied the knowledge of the relation but we have not yet exploited the information available in the input collection which says that certain other people like William Grant, Julia Smith, Aston Miller and so on, also attended University of Texas with John Smith.

As we have more and more relations, more is the likelihood to get the correct set of people. But in the real life, we do not have that privilege. In the most practical situations like presented above we just have the knowledge of one or two relations, and the list of people who are related to each other through that relation. This is a generic problem where we need to find the correct information about a group of people where all of them are related to each other by some relation(s). We have an input collection of names of some people who went to a given graduate school but we do not know when did they attend, where they live, what they do etc. During our research, we have the following relation available to us:

$R_a$: Graduate School name.

Based on the input collection and this relation, we have to find the true social network profiles of the people who are present in the collection. In the above case we have mentioned that we know that the people in the input collection are related to each other using a known relation which is the same graduate school, but the algorithms which we are going to talk about, they can be applied without the knowledge of any relation(s).

We can apply the algorithms we designed, in situations where we do not have any knowledge about any relation existing among the names in the input collection. During our research, we assume that we have the knowledge of a relation so that we can exploit this knowledge of a relation to reduce computations.
So our goal is whether we have the knowledge of the relation or not, we should be able to find the correct social network profiles for all the names present in the input collection.

In the coming chapters we will see the algorithms which we have designed and implement to find the correct social network profiles. The first step is data extraction. This is done by searching all the names in the input collection through the social network and record all the profiles suggested by the social network against that particular name in the input collection. If we have the knowledge about the relation through which these names in the input collection are related, we modify our search queries to include that knowledge which will result in fewer number of search results. This will basically filter our all the irrelevant profiles which do not adhere to that relation. Then we run our algorithms which assign relevance scores to candidate profiles which determine whether this is the profile which we are looking for or not.

1.4 Motivation

In this era of growing importance of social media, there is a huge scope of research work in understanding the relationships and patterns exhibited by people on the social networks. If you want to understand relationships among a group of people, there is no better way than to exploit social media. Our proposed framework is generic and is applicable to all major social networks like LinkedIn, Twitter etc. with minor changes. There are a lot of scenarios where our research work can help solve numerous practical problems:

a) High School & College Re-unions.
Though people have professional networks like LinkedIn to stay in touch with, but they do not offer opportunities for informal gatherings like high-school or college re-unions. Basically if people on social media share any common aspect like education, work, location, etc. they can use this feature of the framework to get together and plan re-unions.

b) Connecting people during disasters.

During any kind of disaster like earthquake, tsunami etc. people with a common geographical location or background can connect to each other using this approach.
\( d\) Detecting criminal gangs.

Government and law enforcement agencies can detect criminal gangs of some mass criminal activity where they have a list of suspects. Usually suspicious people would not like to have their presence on social media but even if out of lets say 30 suspicious names, we find 5-7 names related to each other. There can be a reasonable doubt to track those individuals for investigation purposes.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{detecting_criminal_gangs.png}
\caption{Detecting criminal gangs}
\end{figure}

\( e\) Finding lost friends.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{finding_lost_friends.png}
\caption{Finding lost friends}
\end{figure}
World is a global village today where people move from one place to another because of new opportunities. It's difficult to track old friends and stay in touch with them. Especially people who belong to the era when social media was not popular, have less social media presence than today’s generation. We just need some small bunch of names related to each other through some relationship to track all of the people related to each other using the same relationship. During our research we were able to find correct social media profiles with the same relationship, which were not even in the input collection list provided by the user. This is very interesting because we start getting some new verified profiles even though we are not looking for them. This framework can help re-connect to even those lost friends and relatives whose names also we have forgotten.

\(\text{d)}\) Relationship deduction.

![Figure 1.9. Deducing relationships](image)

If you have a bunch of names, you can use this framework to find any kind of relationship which exists between individuals with the same set of input names.
1.5 Document Organization

In the first chapter, we first talk about the context of the problem which is related to searching a group of names from the Internet using social networks. We explain about the existing web infrastructure provided by the social networks like Facebook, LinkedIn and Twitter and then we talk about their limitations. Finding social network profiles of people just from their names who are related to each other through some common relationship are difficult to find. Some of our suggested algorithms work independently from the fact that we know the type of relationship or not. So basically in this chapter, we talk about the problems which have not been solved by the social networks yet and how we intend to address them. We also talk about the formal thesis statement which contains a clear description of the problems which we intend to solve.

In the second chapter, we talk about how we extracted the public data for our experimentation purposes. But before that we must define our problem in a very logical and mathematical form. We do that in the Formulating the Problem section. For our experimentation purposes, we assume that we do know about the relationship which occurs between the social network profiles beforehand as it saves us a lot of computations. But we do state that our algorithms work independently of the fact that if we know about the type of relationship or not. Then we discuss about the techniques we used to search people in the social network. We will talk about the architecture of a secure, polite and robust social network crawler which we built for the data extraction purposes. We will also talk about the type of storage we used to store the extracted data and about data management in general. Then we will move on to the challenges which we faced while data extraction and how can we build efficient data extraction tools for extracting data from social networks.

In chapter three, we will see an overview of all the algorithms which we came up with. We will also discuss about the ideas, motivation and the reasons which we went through while designing them. We will talk about a total of four approaches which we have applied.
The first one is called the Social Graph approach where we analyze the distances of social network profiles from each other in terms of number of connection hops between them. The second approach is about analyzing the Social Graph with some ground truth information collected from the user. The third approach considers some Seed Friends which are one of the most connected, and deriving correct results by exploiting their social network information. Then we have another approach which is little bit different from all of them, here we use the images of the people to derive the correct relationships between people. This is done by using Watson Visual Recognition on the profile pictures of the candidate profiles to check if they follow a certain criteria with respect to their age, gender etc.

In chapter four, we talk about the results which we get from the experiments. We discuss about the results got from different approaches mentioned in chapter three and based on that we will derive conclusions and give recommendations. Our approaches are based on various different methods like determining the social connectivity, collecting ground truth from users, checking connectivity with the most well connected people, visual recognition which helped us extracting important features from the profile pictures of the profiles which we got from the search results.

In chapter five, we describe the future work which is possible in this direction. There is an enormous potential in exploring the sea of data available in social networks which can be used to establish relationships and find the relevant people connected through them.
CHAPTER 2
DATA EXTRACTION FROM SOCIAL NETWORK

2.1 Formulating the Problem

Suppose we have $n$ names in the input collection named as:

$$X_1, X_2 \ldots X_n$$

The first thought which comes to our mind is to directly start searching these names in the social network, but we will first study this information thoroughly and then come up with efficient queries which will try to cover all the possible profiles which can be extracted against these names. People do not represent themselves in the social media exactly with their formal names. Due to which each $X_i$ will expand to many different possible names.

$$X_i \rightarrow X_{i1}, X_{i2} \ldots X_{in_i}$$

where $n_i$ denotes the number of possible names which can be formed from the name $X_i$ from the input collection. This expansion of $X_i$ to $n_i$ different possible names is done by carefully analyzing the data. More details will be explained in the upcoming chapters.

Obviously, we cannot cover all the cases but we have to come up with different possibilities with which we can expand the horizon of our queries to include all the people with the names very similar to $X_i$. So on the whole, we can say that we would have multiple queries which we have to prepare for each person in the input collection. Assume that for person $X_i$, we come up with a list of queries $LQ_i$. So we have the following list of queries against each person in the input collection which we need to run through the social network:
\(X_1 \rightarrow LQ_1\)
\(X_2 \rightarrow LQ_2\)
...
\(X_n \rightarrow LQ_n\)

Please note that here, \(LQ_i\) is not one query but it is a list of queries,

\(LQ_1 : Q_{11}, Q_{12}, \ldots, Q_{1t_1}\)
\(LQ_2 : Q_{21}, Q_{22}, \ldots, Q_{2t_2}\)
...
\(LQ_n : Q_{n1}, Q_{n2}, \ldots, Q_{nt_n}\)

where \(Q_{ij}\) signifies the \(j\)-th possible query for \(X_i\). Here \(t_i\) signifies the number of queries to be executed for \(X_i\).

So for each name \(X_i\), there are multiple possibilities which we should look for to cover all the possible results. Once we have the queries ready for each of the names, we proceed towards the second task which is to include the information about the common relation which is present among the names in the input collection. Here, that relation means the graduate school where the people went to. After careful observation of how the graduate school is represented in the social network, we come up a set of strings which represent the graduate school in the social network. For example, University of Texas can be represented by ‘ut’, ‘uot’, ‘u. texas’, ‘university of texas’, ‘ut dallas’, ‘ut austin’, ‘uni of texas’, etc. So basically, the relation here, or the graduate school, can take up multiple forms as it can be represented by multiple strings:

\(GS \rightarrow GS_1, GS_2 \ldots GS_u\)
The graduate school relation will remain same for the input collection which will lead to a fix $u$ number of possibilities after careful analysis. Till now we had $t_i$ possible queries for name $X_i$, but now with the introduction of $u$ different possibilities due to the graduate school information, the number of queries become $t_i.u$. Now the next step is to include this knowledge of the relation in the query which will bring down the number of search results substantially. Let us say that for each query $Q$, we introduce the knowledge of the relation, which is graduate school here, transforms the query $Q$ to $Q'$. Till now, we had $t_i$ queries for $X_i$, but now we have $t_i.u$ queries for each name in the input collection because for each query, we have $u$ possibilities because of $u$ number of different ways in which we can query for the graduate school GS. Please note that during our experiments, $u$ would remain constant. Also note that $SR$ corresponds to the search results obtained from query $Q$ ($SR'$ corresponds to $Q'$).

$$X_i \rightarrow Q'_{i1}, Q'_{i2} \ldots Q'_{iu.t_i} \rightarrow SR'_{i1}, SR'_{i2} \ldots SR'_{iu.t_i}$$

We run them through the social network to get the search results. For each of the input names in the collection $X_i$, we get a list of queries to be run and each of the query would produce a list of search results form the social network. So we can finalize the below mentioned mapping between the input collection, queries and the search results.

$$X_i \rightarrow LQ'_i : Q'_{i1}, Q'_{i2} \ldots Q'_{iu.t_i} \rightarrow SR'_{i1}, SR'_{i2} \ldots SR'_{iu.t_i}$$

We can represent the input collection names and their corresponding search results in the following form:

$$X_i \rightarrow SR'_{i1}, SR'_{i2} \ldots SR'_{iu.t_i}$$
Please note that we have reserved the notation $SR$ for representing the search query results of $Q$ and we will use $SR'$ here to depict that the search results of query $Q'$ which contains the knowledge about the relation. We know that $SR'_{ij}$ is a list of search results for j-th query which intends to search $X_i$ from the social network. Let us define the size of this search result as $V_{ij}$. So the total number of search results we get for $X_i$ are:

$$\text{Number of possible social network profiles for } X_i = V_{i1} + V_{i2} + ... + V_{it_{1.u}}$$

We just now derived the number of social network profiles which are possible against a name with the knowledge of relation. In the next section we will see its importance before proceeding to the heuristics or algorithms.

2.2 Knowledge of the Relation(s)

While formulating our heuristics or algorithms for finding the correct people from the social network, we kept them relation-independent. Its an additional piece of information which can lower down our computations, but we can still find the correct people without its knowledge. As we know from the previous section,

$$\text{Number of possible social network profiles for } X_i = V_{i1} + V_{i2} + ... + V_{it_{1.u}}$$

We saw in the previous section that when we include the knowledge of the relation, our query becomes $Q'$ from $Q$ (does not contain information about graduate school). Consider the case of John Smith again. If we search John Smith in the social network, we are expected to get a very large number of results because this name is very common and we are searching it on one of the most popular social networks. Now, suppose we want to introduce the knowledge component to the query which states that John Smith attended University of
Texas. In that case, we will try to formulate many possible queries as mentioned in the previous section:

'john smith university of texas'
'john smith ut'
'john smith uot'
'john smith ut dallas'
'john smith ut austin'
'j. smith university of texas'
... and so on.

These queries are bound to retrieve very few search results compared to a plain John Smith query:

'john smith'

Similar to the analogy where we have $Q$ (without information about relation) and $Q'$ (with information about relation), we define their search results as $SR$ (search results of query $Q$) and $SR'$ (search results of query $Q'$) respectively. As we know that the query containing the relational information is bound to retrieve a very condensed set of profiles, let us assume that the search results get condensed by $\alpha$. This means that $SR$ will have way more search results than $SR'$ because the corresponding query $Q$ is more generic and query $Q'$ is more specific. This implies that the corresponding size of the list $SR$ will be more than $SR'$. As mentioned above, the sizes are denoted by $V$. This implies that:

$$V' = \alpha \cdot V$$
where \(0 \leq \alpha \leq 1\)

Revisiting the previous equation where we said:

Number of possible social network profiles for \(X_i\) is:

\[= V_i + V_2 + ... + V_{it_i,u}\]

\[= V_i'/\alpha_i + V_{i2}'/\alpha_i + ... + V_{it_i,u}'/\alpha_{it_i,u}\]

\[\geq V_i' + V_{i2}' + ... + V_{it_i,u}'\]

because \(0 \leq \alpha \leq 1\)

As all the components have been divided by \(\alpha\), we can deduce that (also observed practically):

\[V_i + V_2 + ... + V_{it_i,u} \gg V_i' + V_{i2}' + ... + V_{it_i,u}'\]

This clearly shows that if we have information about the relation (like graduate school), the number of possible candidate social network profiles reduce drastically. This is the reason why we assume the knowledge of relation during our experiments as it decreases the number of computations but our heuristics or algorithms work independent of the fact whether we have the information about the relation or not.
2.3 Searching the Social Network

Social Searching is mostly used by users to search for other people adhering to certain criteria. As we talked about it in chapter one, social search engines provide variety of facets which when applied, return us the relevant social network profiles. But in our problem statement, we do not have exact users to be searched for but some names which can correspond to multiple social network profiles. In the previous section, we saw that the number of possibilities which the user has to consider are enormous and the user alone cannot search for them manually. It would be very time consuming and an endless job. This is the motivation behind building a tool which will take names in the input collection and search for all the possibilities against them in the given social network. Following is the working flow:

![Diagram of working flow for Searching Related People]

**Figure 2.1. Working flow for Searching Related People**

Following is a brief summary of the flow:

a) User feeds the input collection list to the search engine.
b) The input collection list is a list of strings where each string is a unique name. If we already have the knowledge of the relation (college name etc.), we just collect the relational information and go to the next step.

c) Queries will be formed with all possible combinations of names and their relational information (for eg. college name etc.).

d) Social media crawler will take all the queries, execute them, collect the data extracted and store it into NoSQL database.

e) Algorithms (coming chapters) will be run to give rankings to the profiles with the most relevant profiles on the top.

f) Human intervention in the form of feedback is required to collect few ground truth profiles, which will in turn help generate more verified profiles.

g) Prepare results: verified list, non-verified list, no users found.

2.4 Data Storage and Management

We use NoSQL database for storing the social media data which we are collecting using the social media crawler. Following is a brief description of the format in which we have stored our data:

a) Each record is in the form of a document identified by a unique key. Document contains information about the user-name to be searched. It contains the search results which were found against him while searching the social media. This is basically a dictionary which contains person name as the key and the search results found against him/her in the form of a list.
b) The list against each person contains social media profiles which were found during the search result. Each profile has the basic information about the user (which user displays on his/her profile like education, location, work history, current work etc.). It also contains some of the top connections (friends) of each profile.

c) Each connection variable inside profile will contain link to the social media profile of that particular friend. We can easily transform that into the form of another dictionary but as of now we are not analyzing the social graph beyond this depth.

The above information is stored in MongoDB [20] which is a NoSQL database. MongoDB has several advantages due to which we decided to go forth with it. As mentioned in c), we may need to change the data model in the future if we make friend’s data into another profile object instead of just a link, MongoDB will let us do that with minimal changes.

Figure 2.2. Advantages of using MongoDB (NoSQL) database [20]
2.5 Challenges while Extracting the Data

There are many challenges encountered while extracting the data. Most of them are either related to the social media crawling policy or to the privacy settings of the users. Following are the major challenges which were encountered:

a) Crawling social media with very high frequency: Crawling social media profiles one after another with very less or no time delay causes a problem. Social media servers assume this to be an attack and they might start blocking anymore requests. We need to introduce some time delay between subsequent social media profile visits to ensure that we respect the politeness policy of the social media. Usually, social networks have a minimum threshold amount of time which we should spend between two profile visits to prevent classifying our social media visit as some form of web attack.

b) Crawling large number of profiles with same set of credentials: A large amount of activity observed from one social media user might again be a problem. Even if we crawl the social media with the required time delay, we still might get blocked because of excessively using one particular login credential. In the coming chapter, we will see how we have tackled this problem.

c) User not search-able on the social media: Many people tend to hide their social media profiles from social media search results. It is difficult to find such users because some of these profiles can be the correct profiles which we are looking for. To tackle this scenario, we start analyzing the friends of people who are search-able. We will talk about this in detail in the coming chapters.

d) User’s friend list not available: In some of the social media profiles, friend list is not available for viewing because of access restrictions. Not much can be done about it but if we know our target audience very well, we may develop some apps and request the target audience to give the permissions to the specific apps.
3.1 Social Media Search

In the previous chapters, we have seen that we have to build an enormous number of queries for searching a set of names from social media. One name cannot uniquely identify a person. A person’s name may or may not be consistent across the social media. For example consider a person named ‘Md. Nurul Huda’. ‘Md.’ is an abbreviation for Mohammad or Mohammed. This name can take the following types of strings:

Md. Nurul Huda
Md Nurul Huda
Mohammad Nurul Huda
Mohammed Nurul Huda

If our queries also hold the knowledge of the relation, for example college name, our queries can have different forms on the base of relation. For example, consider the institution named Bangladesh University of Engineering and Technology (BUET) \[7\]. Spending some time on the social media will make us realize that the college name can take the following forms here:

Bangladesh University of Engineering and Technology
Bangladesh University of Engineering & Technology
BUET

As we saw above, the name as well as the college information can take up multiple forms. One name search can lead us to $3 \times 3 = 9$ possible queries.
Let us make the following assumptions:

Number of forms which each name can take: $m$

Number of forms taken by the relational information in the query: $n$

Number of search results given by each query (on average): $p$

This leads to the number of search results against each name: $mnp$.

Now, suppose we have a dataset of more than 500 names which we are looking for.

This leads to a total of $500mnp +$ search results which we need to process in order to find the correct social network profiles against their corresponding names in the input collection.

This is a very high number of search results to be analyzed manually in order to classify them as correct or incorrect against their corresponding names in the input collection. Can we automate this completely? Obviously, a complete automation is not possible. Because if a combination of the right name and the right relational information (we will mostly encounter several search results even with the relational information) yields multiple search results, we need some human intervention or feedback at some point of time to know whether he/she is the person which the human is looking for. For example if we are searching for a person named 'Mahmudul Hasan’ studying at BUET [7], one of the possible queries will be:

'mahmudul hasan buet'

Some of the search results of the above query are as follows:
Even though we are providing the relational information which is the college name, we will still end up having numerous search results. If we try to solve this problem, it would be a very time consuming and inefficient task. If the name is very common, it would lead to a spike in the search results which is very difficult to handle.
There is a need for a semi-automated framework which can automate this task with some human intervention in the form of feedback. Once the framework starts generating best possible search results, some human feedback would be required which will classify some of the initial profiles as positive or negative. Our algorithms are based on the social graph of the people we are looking for in the social media. Even small amount of feedback from humans will boost up the scores of correct profiles and help us to reach the solution set.

As we have seen, there is a large number of possible search queries, which in turn will produce a large number of search results. There has to be an automated mechanism through which the queries are executed and the search results are collected. The search results are nothing but the social media profiles of the candidates from which we have to chose the correct profile corresponding to their name in the input collection list. If we have generic names and very less relational information, this transforms into a big data problem. But we will keep out focus mostly from the algorithmic point of view. We will definitely keep our project flexible enough so that it can be extended to implement big data technologies. For instance, we will be using a NoSQL database throughout our project. We will see other aspects as well as we progress towards the next chapters.

This leads to the need of a social media crawler which can execute all possible queries, collect the search results, visit all candidate profiles, extract all the required data and finally store it into a NoSQL database. We developed one such crawler which we will describe in the next section. Basically, the crawler should have the following properties:

\( a \) Generic in nature, which means it has all the required properties so that it can be used by other social media like LinkedIn, Twitter, etc. as well.

\( b \) Should be efficient and polite enough to avoid blocking by the social media.

\( c \) Should have real-time integration with the NoSQL database to ensure zero data loss.
d) Hosted from unrecognizable local commodity hardware so that crawler’s host IP does not get blocked (for eg. AWS servers [21] are very easily blocked by social media as they are easily recognizable). All major cloud services can be easily blocked by the social media so we should take care of such situations.

3.2 Social Media Crawler

As explained in the previous section, there is a need for an automated social network crawler which can crawl through the social media to extract the relevant data. Functionally, it should be able to perform the following tasks:

a) Authenticate with the social network.

b) Query through social network’s search.

c) Collect the search results and store in a database, efficiently.

d) Visit the social network profile.

e) Extract the required data from the social network profile.

f) Store the collected social media information in database, efficiently.

g) All these steps should keep on going until all the profiles are crawled, without getting blocked.

We used Python’s Mechanize library [22] which offers a Stateful programmatic web browsing in Python. It allows easy HTML form filling, convenient link parsing and following. It also has the provision to store browser history (.back() and .reload() methods). Another benefits are automatic observance of robots.txt and automatic handling of HTTP-Equiv and Refresh.
Along with the functions specified above, it should take care to perform the following tasks:

a) Politeness should be ensured by specifying appropriate time delay.

b) Authenticated session details should be stored/cached to reduce the number of login hits, which can make the social network suspicious.

c) Ensure least number of hits possible on the social network.

d) Use multiple authentications to avoid making the social media suspicious.

We recommend to use above mentioned Python library: mechanize.

We will now talk about the algorithmic details of our social media crawler.

Consider a program `performTestCrawl(n, td, cred)` which crawls over `n` social media profiles with time delay of `td` with the login credential `cred`. We chose `n` to be any sufficiently high number of profiles such that if we crawl `n` profiles without any time delay, we will definitely get blocked. For Facebook, 1000 is a very reasonable number.

While crawling through the data of one of the most prominent social networks, we used to get blocked by the social media. After our experimentation, we came up with the following
algorithm for the safe and un-blocked crawling of the social media.

**Result:** Minimum time delay $t_d$ below which we get blocked.

$t_d = 0$

$n = 1000$

$cred = \text{fetchCredentials}()$

```python
while True:
    result = \text{performTestCrawl}(t_d, n, cred)
    if result is blocked:
        $t_d = t_d + 1$
    else:
        break
```

**Algorithm 1:** Determining the optimal time delay for safe crawling of social media

In the above algorithm, we start from 0 seconds and we keep on increasing the time delay until we reach the politeness expected by the social media. Suppose we start with 0 seconds and we break the loop at 3 seconds (we are not getting blocked at 3 seconds), we will chose 3 seconds as our politeness delay. We can refine the above algorithm further by incrementing by say 0.2 seconds, instead of 1 second, and we may come up with a number like 2.8 seconds which is more optimal than 3 seconds. We have left this up to the programmer while we use integral number of seconds.

Time complexity is $O(t_d c)$, where $t_d$ is very small (single digit seconds). This is because the social networks usually keep the time delay of not more than 2 or 3 seconds. The users are expected to browse social media (which means switching from one social media page to another) with around this much range (2-3 seconds) but not substantially less than that. Also note that here, $c$ means the time taken for performing the test crawl using $t_d$ seconds over $n$ profiles.
Once we have the optimal $t_d$ below which we will be blocked by the social network, we need to find the maximum number of profiles which we need to crawl above which we start getting blocked using the same authentication credential. We try to approach the $n$ in reverse way where we initialize $n$ to a high number which is let us say 1000 for Facebook. We are very much sure that crawling 1000 profiles at once with a single credential will lead to blocking. We then follow a similar approach for this algorithm as well:

**Result:** Maximum number of profiles $n$ to be visited above which we get blocked.

$n = 1000$

while $n > 0$ do

| result = performTestCrawl($t_d$, $n$, $cred$) |
| if result is blocked then |
| $n = n/2$ |
| else |
| break |
| end |

end

$n_{high} = 2n$

while $n \leq n_{high}$ do

| result = performTestCrawl($n + 1$) |
| if result is not blocked then |
| $n = n + 1$ |
| else |
| break |
| end |

end

Algorithm 2: Determining the optimal $n$ for safe crawling of the social media
In the above algorithm, we first start with a \( n \) which is a large number and keep halving it until we do not get blocked by that \( n \). In each loop, we are halving it instead of decrementing it linearly. This is because if we are getting blocked after crawling 1000 profiles, we are very much likely to get blocked by crawling 999 profiles as well. So the number which we are going to check is not 999 but 500. If we are not getting blocked at 500 (which is not likely the case in social media), we know that our target number lies between 500 and 1000. However, this is again a large number and we won’t practically encounter this kind of scenario. Usually, we will encounter a scenario where our number lies between some range like 32 - 64 (after getting halved again and again starting from 1000). We assign 32 to \( n \) and then linearly traverse from 32 to 64 to find the optimal \( n \) in the second part of the algorithm.

Another reason why we are halving is ’politeness’. Let us assume we find the optimal value to be 37 (while traversing from 32 to 64 linearly). Moving from 1000 to 37 will surely get us blocked pretty soon. The best way to reach 37 is in the following steps: 1000, 500, 250, 125, 64, 32, 33, 34, 35, 36, 37. This is a total of 11 steps. In the other case, we would have to resort to 1000 - 37 = 963 steps. We cannot afford to perform test runs 963 times.

Time complexity is \( O(\log n) + O(cx) \), where \( x \) is a small number. This is because if we are trying to find the optimal \( n \) which is lying between 64 and 32 (consider 64 as \( n_{high} \) and 32 as \( n \), then we just need to traverse from 32 to 64 which makes the maximum value of \( x \) as 32.

The above two algorithms were implemented to know the following components:

- Optimal (minimum) politeness time delay \( t_d \) seconds.
- Optimal (maximum) number of profiles \( n \) to be crawled before getting blocked (using same credential).

Once this is done, our next goal is to safely crawl through the social media. The above constants have been derived after running tests using the algorithms mentioned above. Now,
we will use these constants to formulate an algorithm which will ensure un-blocked and safe crawling of the social media in an uninterrupted way. This involves preparing a list of login credentials \( L \), which can authenticate with the social network. Consider a program `fetchCredentialsRandomly(L)` which returns a random login credential (username/password) from a list of available credentials \( L \). We will be choosing credentials randomly after a certain number of social network profile crawls \( p \) (period after which we need to switch credential), so that we do not get blocked. \( p \) signifies the number of profiles after which we should change the login credential by randomly selecting from the list \( L \). Consider the case where \( n \) is 100, so we can sufficiently chose \( p \) to be around 20. The algorithm may fail under the condition where we get the same login credential 5 times leading to a crawl of \( 20 \times 5 = 100 \) profiles with the same credential. But the probability of this happening is very less. We will leave this selection of \( p \) using \( n \) through the function `selectPeriod(n)`, up to the user. In our case, we just chose \( p \) as \( n/5 \) which means that `selectPeriod(n)` will just return \( n/5 \). Following is the algorithm:

**Result:** Un-blocked safe social media crawling

\[
\text{noProfilesVisited} = 0
\]
\[
\text{cred} = \text{fetchCredentialsRandomly}(L)
\]
\[
p = \text{selectPeriod}(n)
\]

**while** more profiles are left **do**

\[
\text{visitProfileAndExtractData()}
\]
\[
\text{wait for } t_d \text{ time}
\]
\[
\text{if } \text{noProfilesVisited is divisible by } p \text{ then}
\]
\[
\text{cred} = \text{fetchCredentialsRandomly}(L)
\]
\[
\text{end}
\]
\[
\text{noProfilesVisited} = \text{noProfilesVisited} + 1
\]

**end**

**Algorithm 3:** Polite Social Network Algorithm for safe crawling of the social media
Time complexity is $O(k(d + t_d))$ where $d$ is the time required to visit a profile and extract data from it. Here, $k$ denotes the total number of profiles to be crawled. Recall that previously, we estimated this number to be more than $500mnp$. Most of the working of the algorithm has been explained above. We will just briefly visit the case where the algorithm might encounter a situation where it selects the same credential again and again (through random selection) for a series of time, which might lead to blocking. Suppose the size of the list containing credentials is $size(L)$. Probability of selecting a credential from the list is $1/size(L)$. Suppose our $p$ is chosen as $n/5$ which means that we need the same credential to be selected 5 times to get blocked. Probability of this happening:

$$\frac{1}{size(L)} \cdot \frac{1}{size(L)} \cdot \frac{1}{size(L)} \cdot ... \cdot \frac{1}{size(L)}$$

Suppose $size(L)$ is 10, we get the above value to be:

$$\frac{1}{(size(L))^5} = 0.00001$$

which is very less.

There is a very less probability of failure of this algorithm, even with just 10 login credentials.

### 3.3 Social Graph Algorithm

This is the first step in finding relationships among the input collection names. Recall that we have a total of more than $500mnp$ candidate profiles (assuming we have more than 500 names in the input collection). Here $m$ denotes the different possible forms of name, $n$ denotes the different possible forms of the relational information and $p$ is the average number of search results returned by each query.
We try to find all possible friendship connections between all the candidate profiles. This is done by analyzing the friends of all the candidate profiles.

**Result:** Social Graph

\[
db = fetchPointerToDatabase() 
\]

\[
profiles = db.readAllProfiles() 
\]

**while more profiles are left do**

\[
profile1 = getCurrentProfile() 
\]

**while more profiles are left do**

\[
profile2 = getCurrentProfile() 
\]

\[
profile2Friends = profile2.getProfileFriends() 
\]

**if profile1 is present in profile2Friends then**

\[
\text{Boost score for profile1} 
\]

**end**

**end**

**Algorithm 4:** Social Graph Algorithm for boosting the relevance scores

Time complexity is \(O(n^2)\), as we are double iterating over all the profiles to find out all possible friendships between them. There might be a person who is not at all connected with any other in the list, such person will have a default score of minimum or 0. A person having exactly one connection with any other profile in the list will have the boosted relevance score. A person having more than one friend in the graph will have more relevance score than someone with only one friend. More the number of friends you have in the social graph, more is the relevance score which a profile has. Suppose we have a list of names which we are looking for: 'Moshfequr Rahman', 'Akbar Rana', 'Nurul Faiz'. All of them went to BUET. So our search queries will encompass all possible combinations of person’s name and college’s name. We know that this above set of three people who went to the same college, attended it
during some time frame (say 1988 - 1993). How many of such sets of these three people exist who went to the same college? The probability is very less. What if we increase the number of names? Suppose now we have 50 such names. There is a very less probability that another set of 50 names exists who went to the same college during the same or different time frame. During our experiments, we used a dataset of more than 500 names. There is a very less probability that another set of 500+ names will exist who went to the same college during the same time frame. We use this basis as the foundation point of our first algorithm, which is Social Graph algorithm.

In the above algorithm, we double iterate over the more than 500 $mnp$ profiles to check all possible friendships between them. If a profile among these 500 $mnp$ profiles has a friendship with any other profile from the rest 500 $mnp - 1$, then we boost its relevance score. Basically we start off with a graph where we have more than 500 $mnp$ disconnected points and then we check for a possible friendship in all of them to build edges between them which correspond to their social network friendships. More the number of edges with other nodes in the graph, more is the relevance score for that particular node. The graph might look somewhat like below diagram:

![Social Graph](image)

Figure 3.4. Social Graph
People are usually connected to each other based on some relationship. Relationships can be of following forms:

a) People who went to BUET.

b) People who live in Dhaka.

c) People who live in Bangladesh.

d) People belonging to the age group of 35-50.

d) People who would pursue engineering as their career.

As shown above, relationships can be of multiple types. We need to extract people with a certain relationship. In our case, we want people who go to the same educational institution which in this case is BUET. But how does our algorithm know which relationship we want to extract? Here is the situation where we need some human intervention in the form of feedback to specify ground truth which will help us identify the correct set of people. We ask some people belonging to the same relevant relationship which we are interested in, i.e. BUET alumni to provide us with some ground truth where they classify the candidate profiles as positive or negative, based on their knowledge. As we collect more and more ground truth, BUET relationship will keep getting highlighted instead of other relationships. This will start building a sub graph which will be connected to ground truth specified by the humans who are BUET alumni.

Mathematically, we are trying to extract the relevant cluster ’people who studied at BUET’.
We will look more into this ground truth collection in the next section:

Here is why our algorithm should work:

Suppose we are searching for someone called ‘John Smith’ in the social media. The name is very common and this will lead to a lot of search results. Now consider we know that ‘William Grant’ also studied with him. Now, using social graph algorithm we are only interested in those people named ‘John Smith’ who are friends with ‘William Grant’. This will tighten the number of search results for ‘John Smith’ and very few people named ‘John Smith’ will get their scores boosted.

Now consider the case where we have 500+ such names. Even if they contain very generic names like ‘John Smith’, ‘William Grant’, ’Mark Henry’, etc. which can individually lead to a lot of search results, we are exploiting the fact that the collective presence of all of them at once (and being related to each other through some relation, like college name) is very rare.
3.4 Specifying Ground Truth

Once we have the social graph for all the 500 $mnp$ profiles, the next step is to ask some of the humans who are BUET alumni, to specify some ground truth which will lead us to finding the relevant profiles which we are looking for. This is done with the help of an online website portal where we present all the candidate profiles to the user and the user has to mark the candidate profiles as positive or negative. The positive examples will go to the list of verified profiles and the negatively marked profiles will be discarded.

In the next page, we can see the screen shot of the portal where the user is presented with candidate profiles against each name in the input collection list. The number of names in the input collection list is huge and it is not possible for the users to find and mark all the verified profiles against their names. This is the reason why we are trying to build this semi-automated framework which will have some human feedback as the starting point and our correct results will start growing on the top of this initial feedback. We already generated the social graph in the previous section where we know which of the 500 $mnp$ profiles are connected to each other and which are not. Using the information of ground truth collected through the online portal, we would get a starting point in the social graph where we can say that we have got an idea about where the verified profiles are located in the social graph.

**Result:** Social Graph (boosted scores for verified profiles)

```
while more profiles are left do
    profile = getCurrentProfile()

    if profile is verified then
        Boost score for profile
    end
end
```

**Algorithm 5:** Boosting the score for verified profiles
How do we select the users who are going to provide the ground truth information? During our experimentation, we were interested in finding out about the alumni of BUET belonging to a certain class. So the human feedback which will be required here is also from the *most well connected and social people who attended BUET from the particular class in which we are interested in*. This is called ground truth. Using the same people we can extract various other relationships which we want for eg.

a) People who live in Dhaka.

b) People who live in Bangladesh.

c) People who are from Dhaka and live in USA.

d) People who are from Bangladesh and pursue engineering as their career.

Accordingly, our ground truth will also vary. For extracting people related to each other using relationship a), we need ground truth from *some well connected people who live in Dhaka*. Similarly, to extract people who are engineers and they are from Bangladesh, we need similar set of *well connected people who are engineers and share a common home country Bangladesh*. So for the above relationship(s) extraction, we need the following set of people for ground truth collection (respectively):

a) Well-connected users who live in Dhaka.

b) Well-connected users who live in Bangladesh.

c) Well-connected users who are from Dhaka and live in USA.

d) Well-connected users who are from Bangladesh and pursue engineering as their career.
Given an option, we tend to look for most well-connected users for human feedback. In the following images, we will some screenshots from the online website portal which we developed. In the first image you can see that the user was presented with the candidate profiles for Manoj Kumar Roy and he marked the first candidate as positive example, rest of them automatically get classified as negative (unless someone has multiple social media profiles). In the second image, we can see that the verified list keeps growing.

Figure 3.6. Candidate profiles presented to the user for 'Manoj Kumar Roy'

Figure 3.7. Online website portal showing the verified list
3.5 Combining Social Graph and Ground Truth

We collected ground truth so that we can highlight only the relevant relationships which we want. Here, we want to highlight those social network profiles who went to BUET. So we asked some of the BUET alumni to provide some ground truth in the form of human feedback. We already had a set of scores for the candidate profiles which we determined from the social graph.

As of now we have the following candidate social media profiles (decreasing order):

a) Profiles which are verified (boosted score to the highest level).

b) Profiles with some connectivity in the social graph (connected to at least one other social media profile).

c) Profiles with no connectivity in the social graph but have some search results.

d) Profiles with no search results.

The highest score is obtained by the verified profiles which have been verified by the user that they are the correct profiles. Then come to the profiles which are connected to each other by some means. They still have a good chance of being the candidate profile because of the same logic we proposed in the previous sections.

There is a very less chance that a large number of people which we are looking for (say 'John', 'William', 'Julie' etc.) exist somewhere who went to the same school which we are looking for (say University of Texas) and they were classmates at that time (say 1985-1990), and they still are not the solution set we are looking for. Basically our algorithm will fail only in that when there exists another set of 'John', 'William', 'Julie' ... say 500 names, who went to University of Texas during 1985-1990 (same batch or class which we are interested in). This has a very negligible probability and this scenario is very unlikely to happen.
The third scenario is the profiles who are not connected to each other. We will just keep those profiles against the user names on the online portal and see if the users can identify any of them. We will talk about this category later.

The last category is the set of users who were not found at all. They have 0 search results against their queries.

As we have started getting the verified social media profiles, we propose the following algorithm for combining the social graph as well the ground truth approach. Here we check if there is any social media profile which is friend of anyone from the verified list. If that is the case we will boost up the profile score as there is a very high chance that a candidate profile which is connected to the verified profile, will be the correct profile for the corresponding user.

**Result:** Social Graph (boosted scores for people connected to verified profiles)

```plaintext
while more profiles are left do
    profile = getCurrentProfile()
    while more verified profiles are left do
        verifiedProfile = getCurrentVerifiedProfile()
        friends = getFriends(verifiedProfile)
        if profile is present in friends then
            Boost score for profile
        end
    end
end
```

**Algorithm 6:** Boosting the score for verified profiles

This will lead to boosting up the score of the profiles connected to the verified profiles.

We show this by an example (next page). Suppose two people are searched using relevant queries in social media and some search results were found. For Abdul Halim, some of its
candidate profiles are connected to the verified list and some are not. In case of Asif Mahmud, only one profile is connected to the verified list.

Figure 3.8. Search results for ’abdul halim buet’

Figure 3.9. Search results for ’asif mahmud buet’
As we can see that the two of the search results of the query ‘abdul halim buet’ are connected to the verified list. One of them is connected to two of the verified profiles but the other one is connected to only one. Both of their scores will be boosted but the one with two friends from the verified list will get a higher boost than the other one.

3.6 Seed’s Friends Algorithm

We need to handle the category d) described in the previous section, which is ‘Profiles with no search results’. We scan through the verified list and see who has one of the highest scores till now. We select top few of them and name them as seeds. This will lead us to some of the most well connected people from the verified list. We extract their friend lists using our social media crawler and try to check if our missing persons are present in those lists with similar names. For example suppose a person named ‘Sharif Shams Imon’ was not found in the social media and had 0 query results. This person might have not mentioned the relational information in his social media profile. Or he may have not allowed his profile to be searched on the social media (less permissions). Or the person is not having his/her social
media profile under that name but some minor variation of the name for eg.:

Kabir Sharif
Mohammed Sharif Ahmed
Mohammad Sharif Ahmad
... so on

A variation in name is a common problem because of which we are not able to find certain social media profiles. So we may not find the exact names of these people in the friends list of seed profiles. But we do need to check for minor variations of them and predict a closer match, and present it to the user. We employ the following algorithm which uses Jaccard Similarity \[ 8 \] to check if the missing person has a close match in the seed’s friend list or not:

**Result:** Dictionary containing all the non zero similarity matches from the seed’s friend list

missingPersonToBeFound

**while** more seed profiles **do**

seedProfile = getCurrentProfile()

**while** more friends of seedProfile **do**

friend = getCurrentFriendOfSeedProfile()

jaccCoeff = jaccard(friend, missingPersonToBeFound)

if jaccCoeff > 0 then

| store this friend in a dictionary

end

end

Algorithm 7: Storing the candidate’s Jaccard coefficients (with seed’s friends) if they are non-zero
The output is a dictionary which contains all the non-zero Jaccard similarity matches. If we sort this data in the decreasing order, we will get some highest matching profiles from the friend lists of the seeds. They are the top candidates which are very likely to be the people which we were not able to find using our queries in the social media. We suggest them to the user in decreasing order and observe that most of them are accepted by the user as the verified profiles.

Figure 3.11. Find no-result-users using Seed’s Friends Algorithm

3.7 Filtering using Machine Learning

Another crucial component of data which we have is the profile pictures of the people. Suppose we are looking for a graduate class from 1988-1993 batch. This means that the estimated age of the profile pictures which we will be looking at should be 35-50. We are keeping a low lower threshold considering inactive people also present in the social media who do not regularly post their recent pictures.
Age can be used for disambiguation purposes while selecting which profile is correct or not. We use IBM Watson’s Visual Recognition [9] used to predict the age from social network profile. Deep Learning based IBM Watson [10] is a very good option to predict age of a person from his/her profile picture. If we find the age of the person belonging to the bracket (35-50) we will further boost up the scores.

Deep Learning based IBM Watson analyzes images for scenes, objects, faces, text, and other subjects that can give you insights into your visual content. We can also train our own classifier using Watson but they already have a highly trained default classifier which can be used to predict age. It takes the input in the following format:

a) JPEG images.

b) PNG images.

c) Custom: Classifier name, JPEG images (positive examples for each class, and negative examples).

d) Custom: Collections - your own images that you want to search for similar images.

and gives the following output:

a) Class description.

b) Class taxonomy.

c) Face detection (Gender, age range, celebrity (very limited see docs).

d) Similar images with confidence scores.

In the next page, we can see the flow of working of IBM Watson’s Visual Recognition. We are mainly interested in the age component, so we will just focus on extracting the age data from the output.
As we can see above, we can also design our own classifier and train it with our positive and negative examples to predict age. But Watson already has a well-trained classifier (trained over a large dataset) that we do not need to create a classifier for our purposes. So we use the default Watson classifier for age prediction. After we are done running Deep Learning based Watson on all the social media profile pictures, we will analyze the results to see which of these profile pictures lie in the age range of 35 to 50. We are interested in only those age predictions which give a high confidence. A predicted value of age with low confidence would not be of much use to us. We will consider only those results which have been generated with high confidence and discard the output for those results which have low confidence. For example:
Age Prediction from Profile Picture

Female detected with age (18-24) with confidence of 0.7

Male detected with age (18-24) with confidence of 0.5

Decision: REJECT example - does not fall in age (35-50)

Figure 3.13. Example (reject)

Age Prediction from Profile Picture

Male detected with age (35-44) with confidence of 0.97+

Decision: ACCEPT example - falls in age (35-50)

Figure 3.14. Example (accept)
CHAPTER 4

RESULTS & DISCUSSION

We started off with the dataset of 500+ people who attended BUET around 1988 - 1993. Applying the relational information of the college name, we were able to extract more than 3000+ social media profiles using our social media crawler. We presented to the users and based on the user feedback, we were able to classify 100+ profiles as verified. Based on the current status of the verified list, we can see that the size of verified list has grown to 102.

The list is still growing and the ideal target is to cover all of the users present in the input collection list. In the table we can see all the relevant information of the people present in the verified list: profile picture, name, educational background, work history, location etc. We also provide the active Facebook link of the person so that the user can connect with him/her. With further application of pruning algorithms and human feedback, our verified list will keep on increasing and the non-verified list will continue to decrease.

One important observation is that we can find the people with the same relationship even if they are not specified in the input collection list given by the user. We had a bunch of names of BUET alumni who attended the college during a certain time frame. Apart from getting the names which we want, we also got other social media profiles which followed the same relationships but were not specified in the input list. Suppose 1000 people attended BUET during 1988 - 1993 but we have the names of only 550. We were also able to find many people from the other 450 which are not present in the input list. This is very useful as we are able to extract the profiles of the people who follow the relation we are looking for, but they were not present in the input collection list.
<table>
<thead>
<tr>
<th>NAME</th>
<th>PROFILE</th>
<th>UNIVERSITY</th>
<th>INDUSTRY</th>
<th>LOCATION</th>
<th>WORK INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abdul Halim</td>
<td>Facebook Profile</td>
<td>Studies at University of Florida; Studied Civil Engineering at Bangladesh University of Engineering and Technology; Went to University Laboratory High School, Dhaka</td>
<td>Followed by 49 people; Former General Manager at Daeju Bangladesh Ltd., Union Group</td>
<td>Lives in Dhaka, Bangladesh; From Dhaka, Bangladesh</td>
<td>Followed by 49 people; Former General Manager at Daeju Bangladesh Ltd., Union Group</td>
</tr>
<tr>
<td>Abdullah Daud</td>
<td>Facebook Profile</td>
<td>Studied at Bangladesh University of Engineering and Technology; Went to Dharmmond Govt. Boys' High School, High School</td>
<td>Worked at The University of Texas at Dallas</td>
<td>Lives in Dallas, Texas; From Dhaka, Bangladesh</td>
<td>Worked at The University of Texas at Dallas</td>
</tr>
<tr>
<td>Ahsan Z. Talukder</td>
<td>Facebook Profile</td>
<td>Studied at Dhaka College; Went to Dharmmond Govt. Boys' High School, High School</td>
<td>Lives in Dhaka, Bangladesh; From Dhaka, Bangladesh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Akbar Rana</td>
<td>Facebook Profile</td>
<td>Studied at gov't college of Commerce; Went to Chittagong Govt. High School</td>
<td>Married to Rowshan Haque</td>
<td>Lives in Chittagong, Bangladesh</td>
<td>Married to Rowshan Haque</td>
</tr>
<tr>
<td>Anup Sen</td>
<td>Facebook Profile</td>
<td>Lives in Orlando, Florida</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ariful Kabir</td>
<td>Facebook Profile</td>
<td>Lives in Orlando, Florida</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asif Mahmood</td>
<td>Facebook Profile</td>
<td>Lives in Orlando, Florida</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Atia Rahman</td>
<td>Facebook Profile</td>
<td>Studied at University of Texas at Austin</td>
<td>Lives in Austin, Texas</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.1. Verified list improving with time
REFERENCES


[7] Bangladesh University of Engineering and Technology: http://www.buet.ac.bd/


[12] LinkedIn: https://www.linkedin.com/


[15] Bing:
   https://www.bing.com/

[16] BITS Pilani:
   http://www.bits-pilani.ac.in/

[17] University of Texas:
   http://www.utexas.edu/

[18] Social Graph:
   https://en.wikipedia.org/wiki/SocialGraph

[19] NoSQL:
   https://en.wikipedia.org/wiki/NoSQL

[20] MongoDB:
   https://www.mongodb.com/

[21] Amazon Web Services:
   https://aws.amazon.com/

[22] Python’s Mechanize Library:
   http://wwwsearch.sourceforge.net/mechanize/

[23] Python’s Mechanize Library (2):
   https://pypi.python.org/pypi/mechanize

[24] Relationship diagram for demonstration purposes:
   https://griffsgraphs.wordpress.com/2012/07/02/a-facebook-network/

[25] Facebook Graph Search:
   https://en.wikipedia.org/wiki/FacebookGraphSearch

[26] Facebook Graph Search meaning:
   https://en.wikipedia.org/wiki/FacebookGraphSearchOperation
VITA

Indervir Singh Banipal was born in Sangrur, Punjab, India on October 8, 1988. After completing his schooling at Sri Guru Harkrishan Sr. Sec. Public School, G. T. Road, Amritsar, he entered Birla Institute of Technology & Science, Pilani (BITS-Pilani), to complete his engineering. In May of 2010, he earned his Bachelor of Engineering (Electrical & Electronics Engineering) and in August of 2014, he entered The Erik Jonnson School of Engineering and Computer Science at The University of Texas at Dallas for his Master of Science degree.