PEOPLE AND REVERSE PEOPLE SEARCH ON THE INTERNET

BY

LINGYONG WANG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Advisor:

Professor Kevin Chen-Chuan Chang

ABSTRACT

Company has needs to maintain their contact information database up to date and manually update contact record is error prone and time consuming. We created an automatic system to keep contact information up to date without any human effort. It is the first system that search and analysis unstructured online data to generate up to date contact information. We combine many different techniques like entity resolution, clustering, random walk, personalized PageRank and etc. System is practical and produces good results.

TABLE OF CONTENTS

Introduction

Nowadays, many companies have a contact database that stores person's name, person's job title, company name and etc. In the business world, contact is very important. For example, if you want to post advertising on CNN website, you want to contact the media director at CNN, having the contact name is crucial here. Because by having the person's name, you can easily find this person's phone number or email address using search engine or searching through yellow page, but having the person's name is the first critical step.

It is also very important to keep all these contact information up to date. Change job is common in real world, so database like this is usually outdated. Manually update is very expensive because human takes a lot of efforts to find person's name that is currently on the job is hard and time consuming. People usually use search engine and go through the search result to first find out possible person's name and for each name they need to do more search to make sure the person is currently holding the job. In the meanwhile, user also need to keep many information in mind like target job title, different job title associated with the person, time and more. Update one record is already time consuming and contact database usually contains hundreds of records and sometime even thousands.

To address this hard job, we create a fully automatic system which can find person names on the job and presenting the results a rank of name from high to low confidence. Based on our experiments on real data, the performance of the system is good enough to be practical.

Motivation and Challenges

Before I started designed the system, I first studied how will I solve this problem and what are the challenges. I first search the job title in search engine, go over the search result to find candidate names that looks like the person who is on the job. After went over five result pages, I start to get tired and missing name in the search result. For each name, I opened each page that contains these names, analysis each name trying to figure out when the page is published, because I need the person is still holding the job. Figuring out the published time for each page is hard, because the time evidence can appear in any place and more often there is just no time information indicating the published time, then I have to discard the page because I am not sure if the information is still up to date.

After all these hard work, now I have many candidate names. For each name, I need farther verify on each person's name, because the evidences in the search result is not convincing enough for me. I farther search each name with the job title trying to find out more evidences to support my decision.

First challenge is that there are too many candidate names, verify on all names is too much effort, usually there are only few people on the same position, there are clearly some noisy names in the candidates. But I had a hard time to flitter names, because keeping track all the possible names is already hard, decide which names are worth to do verify is even harder, usually after scanning many search result pages, I can't remember which names looks like having larger chance in the result pages.

Next I begin to verify the name I think is most likely to be the person by searching name and job title using search engine. I still having the old challenge that I need to make sure the information are up to date. During verification, only reading the search result is not enough anymore, because text snippets in the search result don't contain enough information. I need to process each page's content in order to get more evidences, first determining whether this page is talking about the person, because two different person have same name. This is hard to do with just looking at the page, I need to open multiple pages, comparing between pages, find the pages looks different from other pages. For example, most pages are talking about this person works at Ohio, but couple of pages talking about this person working at Illinois, I think I can’t trust information in these pages because these pages talk about another person.

Now I have pages that are talking about the person based on my judgments, I need to parse out useful information from pages. I first locate the name by search in the page content, then looking at the surrounding text around the name, find the job information from the surrounding text and determine it is the same as my target job title. It takes some time for me to finish all the pages.

After all previous hard work, it has been 10 minutes since I start, and this is only verification for one name. I usually have to verify 4-5 names before I made my decision. Update one record, only the preparation takes more than half hour to do, usually there hundreds of record in the database I need to update.

By having all these evidences for each name, now I need to make my decision that who is currently on the job. This is even harder to do, I have to take so many different features into consideration. In the end after spending more than half hour on it, I am still not confident with my decision.

Base on the observation, I realize I can mimic this procedure and computer can easily do this faster and better. The system flow is very similar, first system searches the job title using search engine. Parsing names from the search result is one challenge, using name tagger can overcome this challenge. After parsed out names, next challenge is determine what names have larger chance, system addresses this issue by calculating relevance score for each name which will be explained in later section, and only keep names have high relevance score. Now, we have the names that are worth doing farther verification. System can generate all these names without any human effort.

Now we are doing the verification for each name. First challenge is determining whether the page is talking about the person we are searching. We create search, crawl, parse, decode and entity resolution modules to mimic this procedure. Entity resolution module can explicit solve this challenge by clustering pages, detail will be explain in the entity resolution section. Another challenge for human is to process each page content, the crawl, parse and decode module will address this issue, crawl will scanning the page content, parse will pares out name surrounding text and decode will decide what are the text talking about.

The hardest part is having all these evidences for all the names and then ranks them. When I did it in person, it is hard to make a ranking decision which leverages all evidences we have found so far. Our system uses random walk personalized PageRank to rank names, experiment shows our algorithm outperformances other baselines. To make this system full automatic, we made some assumptions when designing the system.

Assumptions

According to our study with real world data on the problem, we made few assumptions to support our system design

Assumption 1 - If person is still on the job, there are pages that published in the last two years contain evidence information. We want to find out the person who is currently on the job. There has to some information about that person on the job published online, otherwise even human can't do anything more. This information also needs to be published within last two years. If the information is too old, we as human can't decide if the information is still true now. According to our experiment, two years if the best cut off point, if we only search the pages in last year, we don't have enough evidences to support the decision. If we search older pages, we can't make sure the information is up to date.

Assumption 2 - If the person changed the job, there are pages published in last two years talk about this person with the new job. If the person changed the job, there has to be some information online talks about the person with the new job. If there is no such information online, human can't know whether this person changed the job or not. Similar to previous assumption, we only need recent information to make sure our decision is up to date.

Assumption 3 - The publish date of web pages on Google is very close to their real publish dates. We need to make sure the information is up to date, so our system needs to figure when the information is published online. There are multiple ways to do it. One way is to process the web page's html data and retrieve time information. But there are many noisy time information in the page, it is hard to identify which time stamp is the publish time. Some pages just do not contain any time information. Another way is using the last modified date of the web page, experiments show that the last modified date is often not the published date of the web page and most pages don't set the last modified date.

The most accurate way to retrieve time information is using published date of the webpage on Google, for all the pages we can find publish date is the same as the Google published date. Base on that, systems searches only the web pages that published in last two years on Google.

Based on these three assumptions, we build our system; I will first explain the whole system flow and go into detail explaining each module.

System Architecture and Data Flow

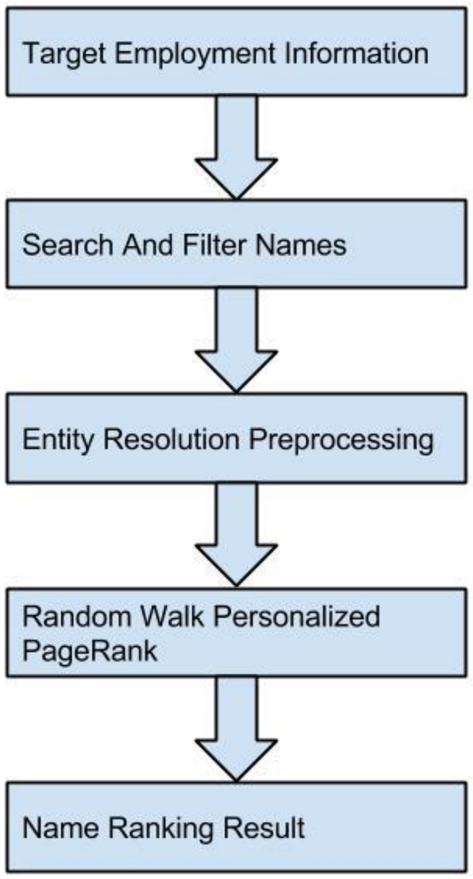
Target Employment Information
Search And Filter Names
Entity Resolution Preprocessing
Random Walk Personalized PageRank
Name Ranking Result

Fig. 1. System architecture and data flow, each box represents one module, the first box is input module asking user for input, the last box is output module presenting the final ranking, in between, are system modules which will be detailed explain in their section.

The system is design by mimicking procedure what human did. First system searches the job title and company name using search engine. The goal of our system is to provide the person's name that is currently on the job. We need to make sure the information we are gathering online is up to date. Our system will only search the pages published in last two years. Next we will parse names out from the search result. Then we will rank the names base on their relevance score that will be explained later at relevance score section. When system will filter out names base on the relevance score and threshold that user entered. Because next two modules take long time and people only need one name that is currently on the job instead of the entire name. After filtering, we will do entity resolution preprocessing on each name. Detail will be explained in the “entity resolution preprocessing” section, basically this module will find all the information related to this person from online unstructured data. Based on these information, we will first build a directed fully connected graph. Then we will do random walk personalized PageRank on it to calculate the ranking score for each name. Detail of this module will also be explained in “Random Walk Personalized PageRank” section.

Search and Filter Names

After we search job title and parse out person names from the search result, we end up with many candidate names. Doing verification on all the names is time consuming and inefficient. One observation we found is there are usually few people on the same job title, and parsed names from search result usually involves lots of noise, we can easily identify these noise names. For example, if the name only appears once, it is very likely it is a noise. Or if the name usually appears with other job title or company name, it is usually noise. Or if the name appears very in pages rank very low in the search result, it usually means the name is likely to be a noise.

By taking these features, co-occurrence frequency, surrounding text relevance and ranking into consideration, we calculated relevance score. The relevance score is indicating how relevance the name to the job title. The higher the relevance scores the higher chance that this person is still on the job.

First we need to find out the words that are actually talking about this person. Closeness is one measure, the closer the words to the name, it is more likely these words are talking about this person. There are couple of exceptions, one, in the Google search result, some snippets contains points of ellipsis (...) which indicating Google omits some words at this position and words after ellipsis should not consider as close to the name. Another exception is that the text snippet can contain more than one name and the words after the other name should consider only relevance to this name.

We define the words that is relevance to the name called relevance block. Relevance blocks starts with the start of the text snippet, person name or point of ellipsis, it ends with the end of text snippet, person name or point of ellipsis.

For example, the text snippet is “Moderator: Sean Corcoran, Senior Vice President, Director of Digital Media & Social ... -Jeff Zannella, Vice President, Associate Media Director, Hill Holliday.” There are two names in the text snippet, “Sean Corcoran” and “Jeff Zannella”, The relevance block for “Sean Corcoran” is “Moderator: Sean Corcoran, Senior Vice President, Director of Digital Media & Social“ and the relevance block for “Jeff Zannella” is “Jeff Zannella, Vice President, Associate Media Director, Hill Holliday.”

The relevance score for one name is the sum of the cosine similarity score between the target employment information and all relevance blocks of this name.

Entity Resolution Preprocessing

Now we have names that are possible on the job, we want to verify each name whether that person is currently on the job. We want to find all the pages about that this person and see if we want find out some evidences saying this person is still on the job. Entity Resolution Preprocessing will accomplish this job, by taking input as a name and job title, output all the evidences that this person is on the job or changed to another job. This module has five smaller modules, search, crawl, parse, decode and entity resolution.

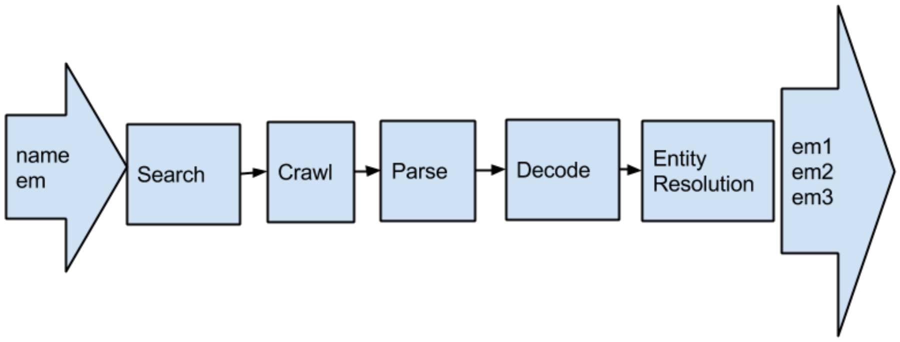

Fig. 2. Entity resolution reprocessing module takes name and target employment information as input and output all the possible employment information.

Search

The intuition behind search module is that if the person changed job, there should be some connections between the old job and new job. For instance, the connection can be two job titles are mentioned in the same page. Base on this observation, we set our search query to name and job title instead of search the name only. According to our experiment, this query works the best.

We also set the search limited to only pages published in last two years base on our assumptions. This can make sure all the information we are getting for this person is up to date. We also filter out pages like job search websites (LinkedIn, Indeed) and data websites (ZoomInfo). Search module will output the entire page URLs that related the person.

Crawl

Crawl module crawls HTML source code for each page’s URL.

Parse

Parse module parses out text surround the person name, it turns each page into many text snippets (name surrounding text).

Decode

Decode module decodes each text snippet and decide what does each text snippet talks about. The category includes Biography, Employment, Education, Award, Presentation, Phone, Fax, Email, Address, Research, Others, Course, Entrepreneurship and Interesting Others. [1, 2, 3]

Entity Resolution

Now we break each page into multiple text snippets, each snippet contains some information about the person. Then we do entity resolution by comparing similarity between text snippets in two pages, if the similarity is above certain threshold, two pages belong to the same entity. [1, 2, 3] After entity resolution, now we have multiple entities. Each entity contains multiple pages. Each page contains multiple text snippets.

Now we want to choose the entity that is the person we want. So we go through the text snippets in each entity, compare each text snippet with the target employment. If the cosine similarity is above certain threshold, we choose this entity. And if there are more than one entity achieving the threshold, we simply merge two entities into one.

With relevance score entity for each name, we can build the graph and finally rank the names.

Random Walk Personalized PageRank

Graph Structure

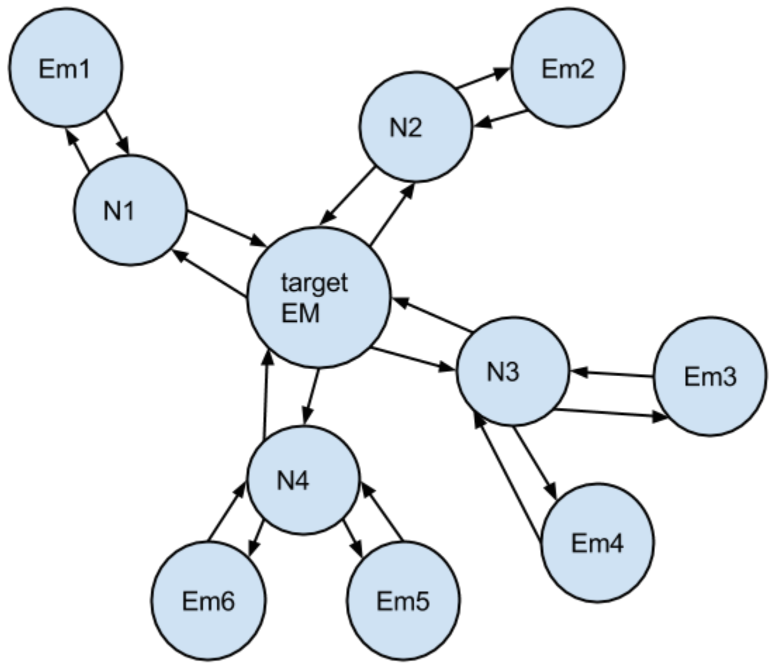

Fig. 3. Graph structure build from processed data. TargetEM is user's input job title which user wants to find person who is currently holding the job, N are names parsed from search result, EM are employment information associated with each name after entity resolution preprocessing.

Our network is directed and fully connected graph. The central node is the target employment information that user entered. Around the central node are the names that system parsed out from search result. Each name node is connected with zero or more employment information that is the output of the entity resolution reprocessing, these are the possible employment information associated with the person.

The graph is defined by the edge weights, in this chapter, I will explain how the edge weights are defined and intuition behind the idea.

Edge: TargetEM -> Name

$$E_{t->n} = e\^(-1*(average\ ranking / 10) \quad (1)$$

The names are parsed out from the text snippets in the search result, each text snippet has ranking in the result. The average ranking is the average ranking of all the text snippets the names appeared. Intuition: This indicates that, if a job title co-occurs with a name in higher rank, then there is a higher chance for the surfer to walk from the job title to the name, because this name is more likely to be the person's name who is currently on the job.

Edge: Name -> TargetEM

$$E_{n->t} = Relevance\ Score \quad (2)$$

The way to calculate relevance score is explained in the relevance score section. Intuition: Relevance score indicates how relevance between name and job title. The higher the relevance score, the higher chance that the name is closely related to the job title and higher chance that the person is on the job, then there is higher chance for surfer to walk from the name to the target employment information.

Edge: Name -> EM

$$E_{n->e} = number\ of\ ER\ snippet\ contains\ EM / total\ number\ of\ snippet \quad (3)$$

After the entity resolution reprocessing, we end up with snippets that include person's possible job information. Intuition: This indicates that, if a name co-occurs with a job title more frequently, then there is a higher chance for the surfer to walk from the name to the job title, because this name is more likely to be the name that is currently on the job.

Edge: EM -> Name

$$E_{e->n} = e\^(-1*(average\ ranking + time\ factor) / 10) \quad (4)$$

Ranking of the web page in the search result that employment information shows up. The employment information can appear in multiple pages, our system uses the average ranking.

Time factor is words that indicating the information are old. For example, “former” is a time factor word, if this words appears closely to the job title, it can indicate there is higher chance that this person is no longer on the job, because it is usually the case that “XX is former CEO of XX company”. We manually created a dictionary for the time factor words.

Intuition: This indicates that, if a job title co-occurs with a name in higher rank, then there is a higher chance for the surfer to walk from the job title to the name, because it is more likely that this person is currently on the job. And the more time factor words shows up, then there is a lower chance for the surfer to walk from job title to the name, because it is more likely the person is no longer on the job.

Now we have the graph and all the edge weights calculated, we will do random walk personalized PageRank on the graph.

Random Walk

$$v = (1-c)vP + cr \quad (5)$$

The known parameters are c is a teleportation parameter, P is transition matrix as defined by the graph edge weights, r is the preference vector. The only unknown parameter is v. Compute v, and rank the name according to their values in v. [4]

For computing preference vector, we maintain a list of EM nodes (denoted as EMList) that are similar to the target employment information according to some similarity function and a threshold. We use both cosine similarity and edit distance as similarity function. Cosine similarity doesn’t handle misspelling cases well. For example, “Hill Holliday” and “Hill Holiday”, the second phrase is missing a letter l on the word “Holliday”. These two phrases mean the same thing, it is just misspelling. But the cosine similarity score between “Hill Holiday” and “Hill Holliday” is 0.28 which letting our system thinks these two phrase are very different, actually they are not. To address this issue, we also use edit distance as another reference. For misspelling cases, the edit distance will be much lower.

Another interesting observation we found is when people writing a long company name online, they usually use abbreviation of the company name. For instance, people usually write “CVB” for “Clermont County Convention & Visitors Bureau”. This bring up another problem, the abbreviation is very different from the original company name. So we let user enter the abbreviation of the company name and we also compare that too.

According to our experiment, the best threshold for cosine similarity over 0.7, for edit distance under 3 and for cosine similarity with the company name abbreviation is also 0.7.

When computing preference value for each node, with only those nodes in EMList having value 1 and the other nodes having value 0.

Experiment

In this section, I will first describe the baselines we used to measure system performance. Then I will use couple of study cases to do more detail comparison.

Baseline1 - Rank names use only Google search result with time. Rank names only base on their appearances in the Google search result. Use this baseline to show that calculating relevance score is necessary.

Baseline 2 - Rank names use only relevance score. Rank names only base on each name's relevance score. Use this baseline to show that doing personalized PageRank is necessary.

Baseline 3 - Rank names use only ranking score from non-random walk approach. Rank names only base on the each name's non-random walk ranking score. This baseline only count each name's score based on its connection to the employments that are in the preference vector. For example, one name is directly connected with four job titles; two of them are in the preference vector, so the ranking score for this name is 2. The purpose of this baseline is to show that random walk information propagation is necessary.

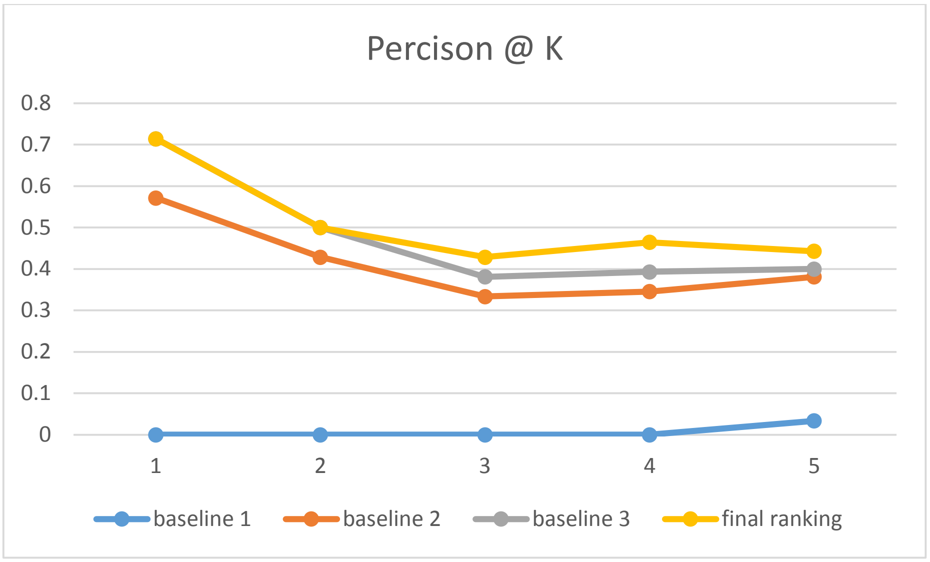

Fig. 4. Precision at K graph, X-axis is K value and Y-axis is the precision value at K.

In calculating precision, “relevant document” is define as the person is currently on the job. Usually there only few person on the same job title, as K increase, precision goes down because the total number of relevant document is small. In the experiment, we tested our three base lines and compare with our final ranking result, our algorithm outperforms all the baselines.

Baseline 1 is only using google search result, the precision is very low because there are just too many noise name in the search result. Baseline 2 is only use the relevance score, it performances better than baseline 1, but there are cases like John and CNN both like this article, relevance ranking with treat John with high relevance score because the name appears very close to the target company name, but actually John is just a random name who happens like the same article with the target company. Baseline 3 performances better than baseline 1 and 2, by doing entity resolution, it fixes the error that baseline 2 has, but it still have some case it won’t handle correctly, because this baseline only considering the entity resolution result instead of considering other features like time factor (“previous”, “former”), ranking the search result, relevance score and etc.

Our algorithm performances the best, it takes good things from other baselines and added more features into consideration. Due to the nature of the application, there are usually few person on the same job, so with K value increase, the precision decreases because in many cases, there is only one related person. When K with low value like one or two, the precision is high, the person name that ranks the first has more than 70% chance that the person is on the job.

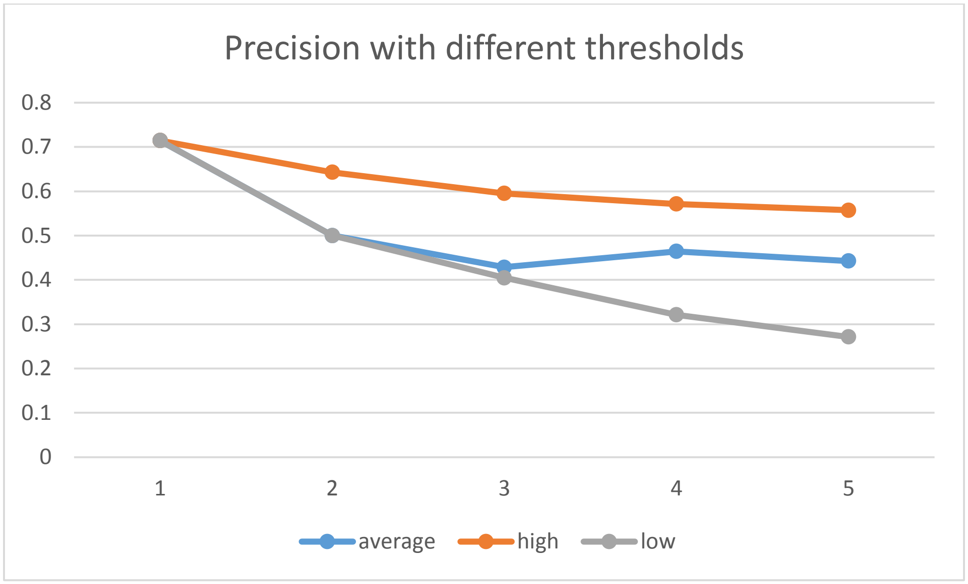

Fig. 5. Precision at K graph with different thresholds for filtering names, X-axis is K value and Y-axis is the precision value at K.

We compared three different settings for threshold which used to filter out noisy names. Average line using the threshold that is the mean of names' relevance scores, high line uses threshold that is 90% higher than average and low line uses threshold that is 80% less than the average. In the result, the high threshold performance the best, with high threshold, system has less noisy names, as K increase, the precision is still higher than other. There are cases that with high threshold, system filter out the target name, but usually system keep the target name, and it shows that our relevance calculating give well scores for names. Low threshold performances the worse at expected, even though there are cases that with lower threshold, we retrieve the target name with low relevance score, but these cases are rare, they doesn't affect result much.

Average threshold is the threshold used default in the system, an interesting observation is that at K = 4, the precision increase. By having more names involve, other names can affect the ranking result. With high threshold, there are few names need to do ranking, sometime, just one name, if this name is the target name, the precision score will be very high for these cases. But with average threshold value, more names are involved in ranking, the target name may rank in lower position instead of the first position, so as K increases, the precision goes up.

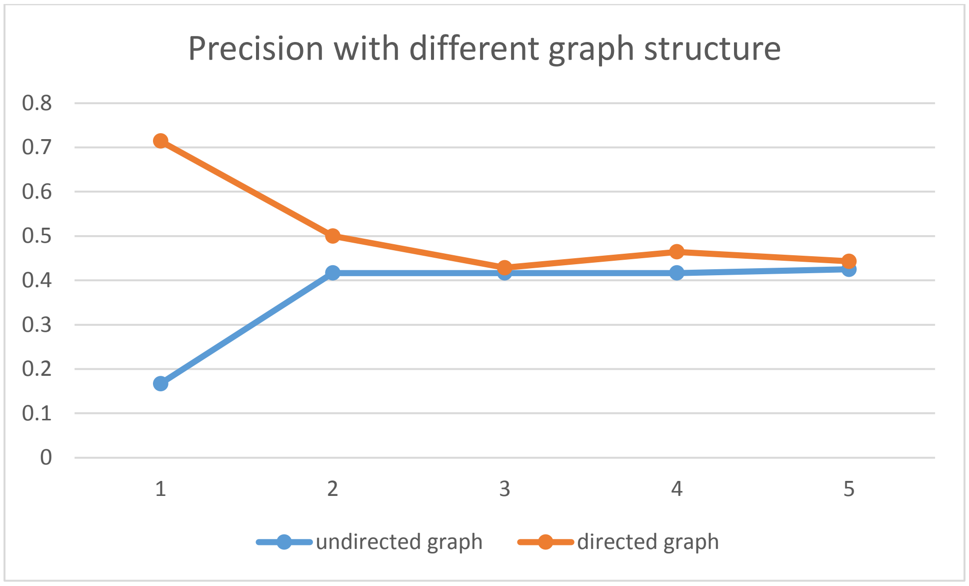

Fig. 6. Precision at K graph with different graph structures, X-axis is K value and Y-axis is the precision value at K.

We also tested our algorithm with different graph structure, undirected graph has the same structure as the directed graph, just with undirected graph edges. The edges are defined at combination of the current directed edge weights. For example, the edge weight between the target node to name nodes are 0.5 * edge weight from target node to name + 0.5 * edge weight from name to target node. Our experiment shows that directed graph significantly outperforms undirected graph. With undirected graph, algorithm performances badly on ranking the target name to the first place. Precision goes higher with K increases, because we are doing filtering before the ranking, so there is high chance there are only few number of names and target name is one of them. If we don't do filtering, the result will be worse.

Case Study

We retrieve the actual job title and company name from real Tripadvisor CRM database. For each job, we manually found the person who is currently on the job. If we can't find ground truth, because sometime there is just not enough data online to determine, we discard the data.

Table 1. "State Tourism Director" "Ohio Department of Development"

Baseline 1 – use only search result	Baseline2 – use only relevance score
Pat Barker	**Christiane Schmenk** (currently on the job)
Gerry Baker	Pat Barker
Melinda Huntley	
Paul Sherlock Award	
Cusick	
Rachael Rahrig	
Christiane Schmenk	
Mary Cusick	
John R. Kasich	
David Goodman	

Compare two results we can clearly see that using relevance score and threshold, we reduce number of names from ten to two which improves the system performance. Christiane Schmenk is the ground truth name, with only search result, this name ranks at seventh position, after we apply our relevance score ranking, we bring this name up to the first position. Using relevance score ranking, we can accurately filter out noisy names.

Table 2. "VP, Digital Associate Media Director" "Hill Holliday"

Baseline 2 – use only relevance score	Final Ranking
Jeff Zannella	**Jocelyn Molla**
Jocelyn Molla	**Jeff Zannella**
Renee Robertson	**Kristin Mollerus**
Kristin Mollerus	Renee Robertson
Brad Blake	Brad Blake

The difference between two results is that Renee Robertson moves down one position in the final ranking which is the ranking using random walk personalized PageRank algorithm. Renee Robertson was on this job before, final ranking brings this name down and bring up Kristin Mollerus who is currently on the job.

Table 3. “VP, Digital Associate Media Director” “Hill Holliday”

Baseline 3 – no random walk	Final Ranking
Kristin Mollerus	**Jocelyn Molla**
Jocelyn Molla	**Jeff Zannella**
Renee Robertson	**Kristin Mollerus**
Jeff Zannella	Renee Robertson
Brad Blake	Brad Blake

Using this comparison, I want to show that doing random walk is necessary. Baseline 3 is the result without random, it simply counts the number of related employment information we got for each name after entity resolution. The result clearly shows that without random walk, Renee Robertson ranks higher than Jeff Zannella, because Renee was on this job, and there are still many related information online. With random walk that takes more features into consideration, Renee’s position went down and Kristin’s position went up.

After we have relevance score for each name, we can filter out names that have low relevance score. We need a threshold for the relevance score. If we set the threshold high, fewer names will let for verification, this will speed up the procedure, but it will lower the precision because we may filter out target names. If we set the threshold low, we will have better precision, but it will slow down the system.

User should be the person to decide the threshold, we let user enter their desired accuracy for the result, and system will set the result accordingly. System takes the average of relevance scores and multiple the accuracy rate as threshold. If some names’ relevance scores are a lot higher than others’, taking the average will emphasis more on these names and filter out other names, because these names are lot higher than other names. If most names’ relevance score are high, average will be high, we will consider more names with high relevance score, because this is likely the case that more than one person on this job. If most names’ relevance scores are low, usually it means we didn’t find much information online, then we want to verify more names to find out more evidences, average relevance score will give use that. And multiple with user’s input will take user’s preference into consideration.

In order to have a good performance with the random walk personalized PageRank, I need to build the correct graph. Two important factors are structure and edge weights. For structure, I need to make sure the graph is fully connected which is the requirement for PageRank. The graph is design base on how the data is generated, the center node is the target job title which user entered and we start with. Directly connect to it are the names parsed out from search result by searching the target job title. Directly connect to each name are the job information nodes that we generate for each name using the entity resolution preprocessing module.

Another important factor is graph's edge weights and I explained the intuition under each weight design, with the right design doesn't mean a good performance. After we did some experiments, the PageRank result still doesn't outperform the relevance score. I realize the formula gives too much weight on the edge between target job title and names and these edge weights are the relevance scores. This means our formula is not random enough, so I tuned the c value larger to give me randomness and the result becomes outperform relevance score.

Limitations and Future Works

Table 4. "Executive director" "Clermont County Convention and Visitors Bureau"

Baseline 2 – use only relevance score	Final ranking
Chris Smith (formerly on the job) Mark Calitri (currently on the job - newly named) John Krug	Chris Smith Mark Calitri John Krug

"Mark Calitri" is newly hired for this position, there are few information online about this. On the other hand, Chris Smith has been on the job for many years and just got retired. Even though there are pages using the words "former" or "interim", there are still many pages still think Chris is still on the job. In result, Mark doesn't move to the first place.

Table 5. "Director of Communications" "Maryland Live! Casino"

Baseline 2 – use only relevance score	Final ranking
Carmen Gonzales (Currently on the job) Mike Smith	Mike Smith Carmen Gonzales

Final result moves the correct person name to second place. This is because of the company name, "Live!" is an unusual company name and tagger doesn't recognize this as a company name. So in result, the employment nodes associated with "Carmen Gonzales" are not complete which make the result worse.

In the future, we can involve machine learning into system to help us better identify cases like "former CEO" or "interim executive director". When we are more confidence on find these cases, we can propagate this finding to other nodes that can help lower the ranking score for this name. To address the second issue, we can build a better classifier for the tagger to improve performance on identifying company names.

The way we designed our graph and algorithm that it works best when comparing names that are close related to the job title. For example, it works best when ranking names that are either currently on the job, or was on the job, or in the same company. It doesn’t work very well when comparing names that some names are not closely related to the job title. For example, when ranking names that one name is the person currently on the job, another name is a random person who don’t have any employment information online, usually for names like this, we didn’t get anything out from entity resolution. But this name is still appears in the relevance ranking because this name co-occur with the job title. For example, John and CNN both liked this article, the name appears closely to the company name we want to search, and then it ranks high in relevance score. When ranking names like this, sometimes, the algorithm ranks the random person higher, because in the graph, that name is still connected with the target employment node.

One observation I made is that this happens when there exist name that doesn’t have any employment node connected after entity resolution. If there is no employment information about this person online, system thinks this person in no on the job because there is no enough evidence to support that.

System filter out names that doesn’t have any employment node directly connected other than the target employment node. This is just a naïve fix for the problem, in the feature, I will better design the algorithm which can fix this error in the first place.

Conclusion

Keeping contact information up to date is a common request. Update one contact record manually is error prone and time consuming and nowadays the contact records can easily be hundreds and thousands. We created a fully automatic system to address this issue. System combines many state-of-art techniques, combining these techniques and keeping good performance as a whole is challenging and we make it happened. Even though there are still some limitations with system, it is still very useful. In the feature, we plan to involve more techniques to improve the performance.

References

1. K. Ganchev, J. a. Graça, J. Gillenwater, and B. Taskar. Posterior regularization for structured latent variable models. J. Mach. Learn. Res. (JMLR), 11:2001–2049, aug 2010.

2. A. Bakalov, A. Fuxman, P. P. Talukdar, and S. Chakrabarti. Scad: collective discovery of attribute values. In Proc. of Intl. Conf. on World Wide Web (WWW), pages 447–456, 2011.

3. J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In Proc. of ACL, pages 363–370, 2005.

4. Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web.