© 2016 Nikita Valeryevich Spirin. All rights reserved.

OPTIMIZING SEARCH USER INTERFACES AND INTERACTIONS WITHIN PROFESSIONAL SOCIAL NETWORKS

BY

NIKITA VALERYEVICH SPIRIN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Doctoral Committee:

Associate Professor Karrie G. Karahalios, Chair and Director of Research
Professor ChengXiang Zhai, Co-Director of Research
Professor Jiawei Han
Doctor Daniel Tunkelang, LinkedIn

Abstract

Professional social networks (PSNs) play the key role in the online social media ecosystem, generate hundreds of terabytes of new data per day, and connect millions of people. To help users cope with the scale and influx of new information, PSNs provide search functionality. However, most of the search engines within PSNs today still provide only keyword queries, basic faceted search capabilities, and uninformative query-biased snippets overlooking the structured and interlinked nature of PSN entities. This results in siloed information, inefficient results presentation, and suboptimal search user experience (UX). In this thesis, we reconsider and comprehensively study input, control, and presentation elements of the search user interface (SUI) to enable more effective and efficient search within PSNs. Specifically, we demonstrate that: (1) named entity queries (NEQs) and structured queries (SQs) complement each other helping PSN users search for people and explore the PSN social graph beyond the first degree; (2) relevance-aware filtering saves users' efforts when they sort jobs, status updates, and people by an attribute value rather than by relevance; (3) extended informative structured snippets increase job search effectiveness and efficiency by leveraging human intelligence and exposing the most critical information about jobs right on a search engine result page (SERP); and (4) non-redundant delta snippets, which different from traditional query-biased snippets show on a SERP information relevant but complementary to the query, are more favored by users performing entity (e.g. people) search, lead to faster task completion times and better search outcomes. Thus, *by modeling the structured and interlinked nature of PSN entities, we can optimize the query-refine-view interaction loop, facilitate serendipitous network exploration, and increase search utility.* We believe that the insights, algorithms, and recommendations presented in this thesis will serve the next generation designers of SUIs within and beyond PSNs and shape the (structured) search landscape of the future.

To my wise and inspiring grandfathers Igor and Fedor, and the Universe.

Acknowledgments

On these pages, I want to express my deepest gratitude to all people, who helped me during my graduate school time and made significant contributions to my development as a researcher.

I am lucky to have two world-class advisers Dr. Karrie Karahalios and Dr. ChengXiang Zhai, who always provided their support and guidance during the foggy moments of the graduate school.

Karrie, I am very thankful that you agreed to accept me to the Social Spaces Research Group and become my adviser despite my interest in startups, which is known to be a huge distraction during the PhD. It was risky to tell you as a potential adviser about it, but I am very glad that I did, you made a step forward, and we established such a high level of trust from the very beginning. All of our communication since then went like a charm. I appreciate your emotional support and your openness to new ideas. By working with you, I realized that we, as technologists, have the utmost responsibility to think about the influence of our work on society. From now on, it is on my checklist when I start a new "project". I also highly appreciate that you taught me how to think like a social scientist and how to embrace the complexity of human beings and interactions among them. I love social computing now.

Cheng, I am very happy that you agreed to become my adviser. While we didn't work with you as long as I would love, I am very thankful for all great things that I was able to learn from you as a researcher and as a person. Every paper that you co-authored has such a crystal line of thinking that it is hard to describe in words. For example, I like how you built up a powerful model from a very simple concept in [90]. I am thankful for being able to work with you and pick up this skill. I also highly admire the way you work with the students. Every time a student, including me, comes to your office, s/he is guaranteed to learn something new and insightful and get a boost of inspiration. Your care for the students, deep desire to help others, and humble attitude brightly complement your solid research contributions. When in doubt, I will always remind myself about such a great man like you.

I am thankful to my committee members Dr. Jiawei Han and Dr. Daniel Tunkelang for finding time to read through the thesis drafts and attend my presentations despite their busy schedules. However, most importantly I am thankful for being able to learn so many valuable things from both of them.

Dr. Han, everybody knows you as a very distinguished researcher in the Data Mining field. This is unquestionable. Your wisdom, sharp insights, and the focus on high impact research always did and continue to inspire me. That said, I want to highlight the other remarkable feature of yours — your ability to create collaborative research teams and environments and manage people. Being welcomed at the Data Mining Research Group meetings, I was able to literally feel it. Your group functions like a living organism, and because of this serendipitous yet carefully designed process the world sees great research innovations. By experiencing it myself, I hope I will be able to create in the future something as beautiful from the management point of view as this. I am glad I had an honor to be around you.

Daniel, it is fantastic to have you as a committee member. Your insights and feedback coming from the industry experience at Endeca, Google, and LinkedIn are invaluable and uniquely complement the opinions of all other committee members. I really appreciate your comments as they are always actionable and clearly formulated. Our (infrequent) interactions, online or offline, are always a bliss. I admire the positive energy that you radiate. Plus, every time we communicate, there is something to think about. I especially remember the moment when I asked you to join in as a committee member. It was busy time for you and chances were slim that you would accept the invitation because you had to "focus" on the startup. This word stuck in my head and this very interaction had a profound impact on me. That "focus" did help me finish this work and I will continue to "focus" moving forward. I am very glad that you agreed to join in.

I would like to thank my early advisers Dr. Kevin Chang and Dr. Konstantin Vorontsov for introducing me to the research as a way of life and for laying the foundation for my research endeavours.

Kevin, you are the first person from UIUC, who believed in me, and I am very thankful that you took me under your wing in the Forward Research Group when I began my PhD. I learnt a lot from you. You sharpened my thinking and cultivated patience, persistence, and attention to the details through multiple meetings and discussions in the first few years of my graduate school. Despite having Cazoodle company and three kids, you always could find extra time to meet if it was the right thing to do. Your high personal standards, integrity, and strong belief in what you do even when the entire world goes the other way are contagious. I will always remember the stories that you shared with me and "think deep" while changing the world with the solid systems and principled frameworks.

Konstantin, you are my first research mentor and adviser and I am very happy that I had a chance to work with you at Moscow Institute of Physics and Technology as an undergraduate student before I left for UIUC. It is through your excellent book and lectures on machine learning that I became excited about this fascinating field of science. And it is thanks to your openness to new ideas that I ended up working on search ranking, which is very vaguely related to the combinatorial theory of overfitting interesting for you at that time. Still, you were able to provide top notch guidance and had a unique perspective even on this novel for you topic. I also appreciate the support that you provided while I was preparing the application for the graduate school and I am thankful that you let me go without trying to keep in your research lab. I admire your work ethic and that you are always open to debate with your students as with the peers, without statuses and other social biases. Your dedication to the scientific method and logical thinking are remarkable. I cherish every interaction we have — it always provides food for the brain and for the soul.

The connections that we have with other people define who we are [24]. I am very lucky to be a member of such a bright and creative community as UIUC Computer Science Department. In particular, I am thrilled to be around wonderful people from Social Spaces (Danish Chopra, Sanorita Dey, Motahhare Eslami, Cole Gleason, Jennifer G. Kim, Hidy Kong, Manoj Krishnan, John Lee, Mary Pietrowicz, Kristen Vaccaro) and Forward (Mangesh Bendre, Yuan Fang, Manuel Lagang, Rui Li, Aston Zhang, Mianwei Zhou) Groups, who made my PhD journey more fun. I am also thankful to my collaborators Maxime Boucher, Mike Develin, Sylvia Ding, Hadi Hashemi, Junfeng He, Pavel Izhutov, Jaap Kamps, Julia Kiseleva, Alexander Kotov, Michael Kuznetsov, Vassil Mladenov, and Yaroslav Spirin. I learnt a lot by working with you. Special thanks are due to Junfeng He and Rajat Raina for being such great hosts during my research internship at Facebook and to Lada Adamic, Kevyn Collins-Thompson, Nick Craswell, Susan Dumais, David Hawking, Andrew Tomkins, and Eugene Agichtein for thoughtful discussions during WSDM2016 DC. The list would be incomplete without my roommates Fardin Abdi, Ore Alebiosu, Jason Croft, Tao Feng, Siddharth Gupta, Alex Gyori, John Hall, Luke Leslie, David Lundgren, and Kuanyu Tseng, who let me work on my thesis in a quiet place. There are many more friends that I would love to mention here, but there is not enough space to list all great people who surrounded me during my graduate school time.

This project would not be possible without a fantastic group of people from the Academic Office of our Computer Science Department. I am very thankful to Holly Bagwell, Kim Bogle, Cynthia Coleman, Donna Coleman, Candy Foster, Mary Beth Kelley, Viveka Kudaligama, Kara MacGregor, Rhonda Kay McElroy, Kathy Runck, and Elaine Wilson for the great service and support. Special thanks are due to Holly, Kara, Kathy, and Mary Beth for making every visit to the Academic Office a positive experience.

Many thanks to my friends from Moscow Institute of Physics and Technology, to anonymous reviewers for CIKM, SIGIR, WWW, and to the partners from the industry for sharing problems and data sets that improved the ideas presented in this thesis.

The last but not the least are many thanks to my family, Olga Spirina, Valery Spirin, Yarolsav Spirin, and my grandparents for being so understanding and supportive of my crazy ideas and for always being so near while apart. It is the first PhD thesis in our extended family. I hope it is not the last.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

"The greatest pay-off for information science will come if and when it successfully integrates systems and users research."

— Dr. Tefko Saracevic, Gerard Salton Awardee '1997

Long ago researchers recognized the importance of social networks and started to study their emergent properties [24] such as diffusion of innovations [104], resilience to shocks [95], and information propagation [156], to name just a few. In the last two decades modern technologies, notably wikis, email, and real-time messaging, further enhanced the power of social networks in information dissemination [31, 48, 55, 148], viral marketing [23, 35, 73], network search [1], expert search [71, 160] and led to the emergence of a new concept — an online social network (OSN). OSNs made the revolution in the way people communicate with each other by connecting millions of individuals on the same platform, simplifying maintenance of connections with significantly more individuals (weak ties [47]) than it is evolutionary possible [39], and enabling many engaging and useful applications such as photo sharing, real-time chatting, virtual collaboration, etc.

As we spend most of our time at work and professional activities represent such a significant part of our life, professional networking has gotten into prominence, and dedicated professional social networks (PSN) have become a sweet spot in the OSN ecosystem. PSNs are especially valuable since weak ties play a crucial role in professional networking and job search [47], i.e. PSNs can be considered as a "killer application" for online social networking. Thus, LinkedIn, undoubtedly the largest PSN, has grown ten times in the last five years and connected more than 433 million professionals[1] as of May 2016. Likewise, Facebook, less often considered as a PSN but still actively used for this purpose [4, 15, 33, 115], has grown three times since 2010 and scored 1.65 billion monthly active users[2] as of March 31, 2016. And, according to Reid Hoffman [56][3], Tim O'Reilly [96][4], and many other network thinkers, value is yet to come as OSNs, PSNs, and, more broadly, networked platforms grow, densify [77], and penetrate deeper into our society.

[1]`https://press.linkedin.com/about-linkedin`
[2]`http://newsroom.fb.com/Key-Facts`
[3]A co-founder of LinkedIn and Chair of the Board.
[4]The founder and CEO of O'Reilly Media and one of the key Web 2.0 evangelists.

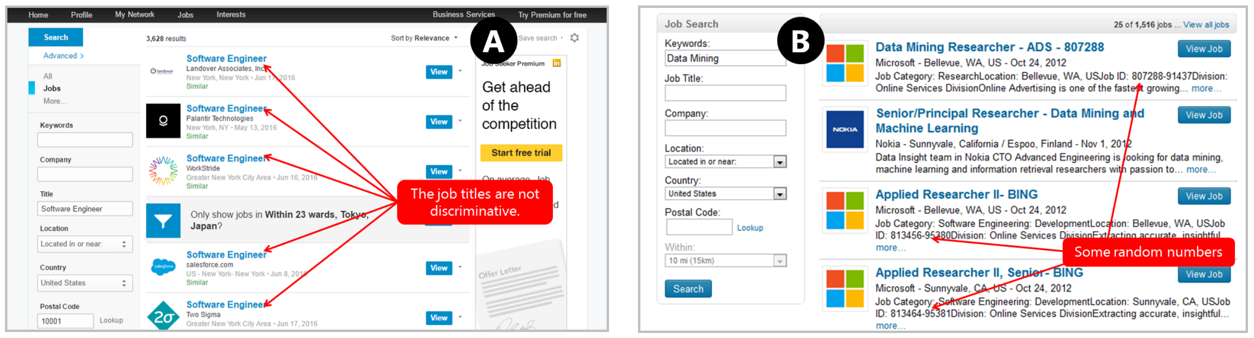


Figure 1.1: Existing job SUIs: (A) LinkedIn's SERP for the query *"Job title: Software Engineer; Location: New York"* as of June 2016. (B) LinkedIn's SERP for the query *"Data Mining"* as of March 2013.

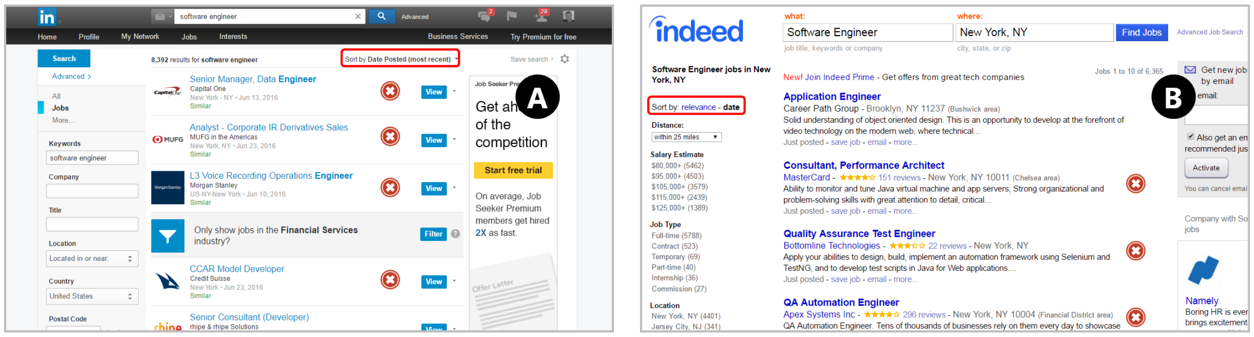
People use PSNs to quickly share personal and professional updates, participate in topical discussions in groups, form partnerships, post jobs, and market their services. This ease of content production enabled by Web 2.0 technologies and coupled with the scale of PSNs have resulted in the exponential growth of information [70] — popular PSNs generate hundreds of terabytes of new data per day [8, 119, 149].

To help users cope with the immense scale and influx of new information, PSNs provide search functionality. Below, we present a fictional job search scenario to illustrate this functionality, typical PSN search tasks, search user interfaces and provide the necessary background for the readers unfamiliar with PSNs.

1.1 Scenario: Search Within Professional Social Networks Today

Alice is a new computer science graduate, and she is looking for a software engineering job in New York. Being a computer geek, she goes online to conduct her job search. She opens the *"Jobs"* tab on LinkedIn and sees the input interface with two fields for a job title and a location. She enters *"Software Engineer"* and *"New York"* in the corresponding fields and hits the *"Search"* button. As output, the job search engine returns a SERP with ten results (Figure 1.1, A). Each result on the SERP is represented by a snippet containing a location, a company, a publication date, and a company logo (some job search engines additionally show a short textual job description, e.g. Figure 1.1, B). The results are quite relevant. However, the SERP is not informative overall as it is difficult for Alice to discriminate one result from another. It is because the titles for all retrieved jobs are the same and the snippets repeat information from the query. Therefore, Alice has to click on each result one-by-one. Many of her clicks might be wasteful.

Next, Alice decides to change her search strategy. Instead of sorting jobs by relevance, she tries to resort them by date because jobs have very short lifespans and HRs usually more actively reply to the job inquiries



software engineer

8,392 results for **software engineer**

Sort by **Date Posted (most recent)**

Keywords: software engineer
Company
Title
Location: Located in or near:
Country: United States
Postal Code

Senior Manager, Data **Engineer**
Capital One
New York - NY • Jun 13, 2016
Similar

Analyst - Corporate IR Derivatives Sales
MUFG in the Americas
New York, NY • Jun 23, 2016
Similar

L3 Voice Recording Operations **Engineer**
Morgan Stanley
US-NY-New York • Jun 10, 2016
Similar

Only show jobs in the **Financial Services** industry?

CCAR Model Developer
Credit Suisse
New York • Jun 23, 2016
Similar

Senior Consultant (Developer)
rhipe & rhipe Solutions

Job Seeker Pre
Get ah
of the
competition

On average, Job Seeker Premium members get hired **2X** as fast.

what: Software Engineer
job title, keywords or company
where: New York, NY
city, state, or zip

Software Engineer jobs in New York, NY

Sort by: relevance - **date**

Distance: within 25 miles

Salary Estimate
$80,000+ (5462)
$95,000+ (4503)
$105,000+ (3579)
$115,000+ (2439)
$125,000+ (1389)

Job Type
Full-time (5788)
Contract (523)
Temporary (69)
Part-time (40)
Internship (36)
Commission (27)

Location
New York, NY (4401)
Jersey City, NJ (341)

New! Join Indeed Prime - Get offers from great tech companies

Jobs 1 to 10 of 6,365

Application Engineer
Career Path Group - Brooklyn, NY 11237 (Bushwick area)
Solid understanding of object oriented design. This is an opportunity to develop at the forefront of video technology on the modern web, where technical...
Just posted - save job - email - more...

Consultant, Performance Architect
MasterCard - ★★★★☆ 151 reviews - New York, NY 10011 (Chelsea area)
Ability to monitor and tune Java virtual machine and app servers; Strong organizational and problem-solving skills with great attention to detail, critical...
Just posted - save job - email - more...

Quality Assurance Test Engineer
Bottomline Technologies - ★★★☆☆ 22 reviews - New York, NY
Apply your abilities to design, build, implement an automation framework using Selenium and TestNG, and to develop test scripts in Java for Web applications....
Just posted - save job - email - more...

QA Automation Engineer
Apex Systems Inc - ★★★★☆ 296 reviews - New York, NY 10004 (Financial District area)
QA Automation Engineer. Tens of thousands of businesses rely on them every day to showcase

Get new job
by email
email:

✓ Also get an em
recommended jus

You can cancel emai

Company with So
jobs

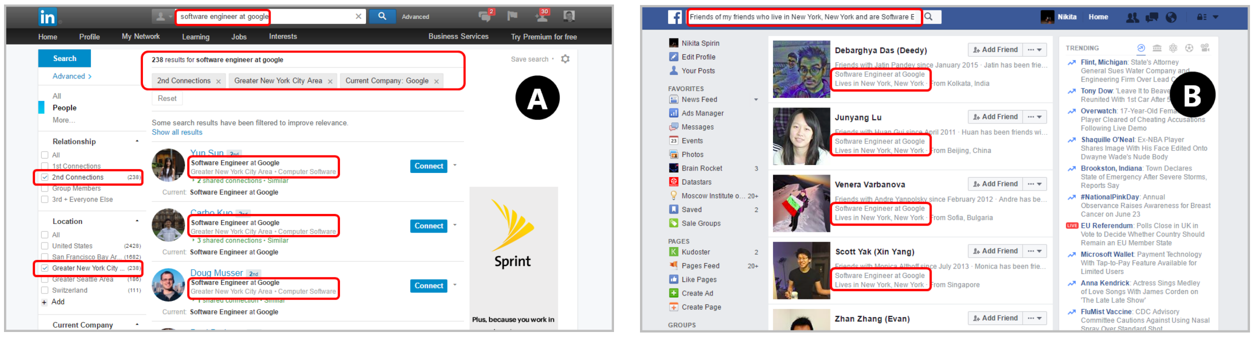
Namely
Boring HR is ever
brings excitement

Figure 1.2: *(A)* LinkedIn's job search results for the query *"Software Engineer in New York"* sorted by *"date"*. *(B)* Indeed's job search results for the query *"Software Engineer in New York"* sorted by *"date"*. While sorting by relevance is accurate, the results sorted by date are hardly relevant for the query.

about the recent jobs. First, she does this search on LinkedIn (Figure 1.2, A). Unfortunately, the results shown at the top of the list are hardly relevant to the query when she applies results resorting. Some recently added but weakly relevant jobs from the bottom of the result list sorted by relevance jump to the top of the list resorted by date. Trying to avoid this problem, she repeats the same search using the Indeed's job search engine (Figure 1.2, B). However, the Indeed's results sorted by date aren't relevant, either. It is because nowadays relevance is not taken into account in such cases or very simplistic heuristics are used (e.g. resort jobs by an attribute value). Despite all these problems she selects a few jobs interesting for her.

Next, Alice wants to learn more information about the jobs and companies she selected. To accomplish this, she switches to the LinkedIn's people search tab and searches for friends who work in relevant companies as software engineers. However, none of her friends work in the companies she selected. Therefore, rather than looking for friends, she starts looking for friends of friends. For that, she uses faceted search functionality [132, 152] and applies the *"2nd degree"* filter by clicking on the corresponding checkbox. She also specifies a company and a location to make sure that search results match her current search goal. The search engine successfully returns a list of results (Figure 1.3, A). But it is again difficult for Alice to differentiate one result from another since all people look the same — they all work at Google as software engineers and live in New York (the redundant information is highlighted in red). Alice executes the same people search strategy on Facebook but experiences the same difficulties (Figure 1.3, B).

Suddenly, a friend calls Alice and invites her to attend an invite-only tech event. Alice finds this opportunity exciting and decides to join in. During the event she makes new friends and, luckily for Alice, some of them work at the companies she selected earlier. They commit to answering her career questions. Unfortunately, she forgets to ask for their names. On the way home, she opens the LinkedIn's mobile application and tries to find these people by describing them using the information she remembers. She specifies



software engineer at google

238 results for software engineer at google

2nd Connections × Greater New York City Area × Current Company: Google ×

Some search results have been filtered to improve relevance.
Show all results

Yun Sun 2nd
Software Engineer at Google
Greater New York City Area • Computer Software
2 shared connections • Similar
Current: Software Engineer at Google

Carbo Kuo 2nd
Software Engineer at Google
Greater New York City Area • Computer Software
3 shared connections • Similar
Current: Software Engineer at Google

Doug Musser 2nd
Software Engineer at Google
Greater New York City Area • Computer Software
1 shared connection • Similar
Current: Software Engineer at Google

Friends of my friends who live in New York, New York and are Software E

Debarghya Das (Deedy)
Friends with Jatin Pandey since January 2015 · Jatin has been frie...
Software Engineer at Google
Lives in New York, New York · From Kolkata, India

Junyang Lu
Friends with Huan Gui since April 2011 · Huan has been friends wi...
Software Engineer at Google
Lives in New York, New York · From Beijing, China

Venera Varbanova
Friends with Andre Yanpolsky since February 2012 · Andre has be...
Software Engineer at Google
Lives in New York, New York · From Sofia, Bulgaria

Scott Yak (Xin Yang)
Friends with Monica Althoff since July 2013 · Monica has been frie...
Software Engineer at Google
Lives in New York, New York · From Singapore

Zhan Zhang (Evan)

Flint, Michigan: State's Attorney General Sues Water Company and Engineering Firm Over Lead C

Tony Dow: 'Leave It to Beave Reunited With 1st Car After 5

Overwatch: 17-Year-Old Fema Player Cleared of Cheating Accusations Following Live Demo

Shaquille O'Neal: Ex-NBA Player Shares Image With His Face Edited Onto Dwayne Wade's Nude Body

Brookston, Indiana: Town Declares State of Emergency After Severe Storms, Reports Say

#NationalPinkDay: Annual Observance Raises Awareness for Breast Cancer on June 23

LIVE EU Referendum: Polls Close in UK in Vote to Decide Whether Country Should Remain an EU Member State

Microsoft Wallet: Payment Technology With Tap-to-Pay Feature Available for Limited Users

Anna Kendrick: Actress Sings Medley of Love Songs With James Corden on 'The Late Late Show'

FluMist Vaccine: CDC Advisory Committee Cautions Against Using Nasal Spray Over Standard Shot

Figure 1.3: (A) A LinkedIn's SERP as of May 2016 for the query *"Software Engineer at Google; Location: Greater New York City Area; Degree: 2nd"*; (B) A Facebook's SERP for the query *"Friends of my friends who work at Google and are Software Engineers and live in New York"* as of June 2016.

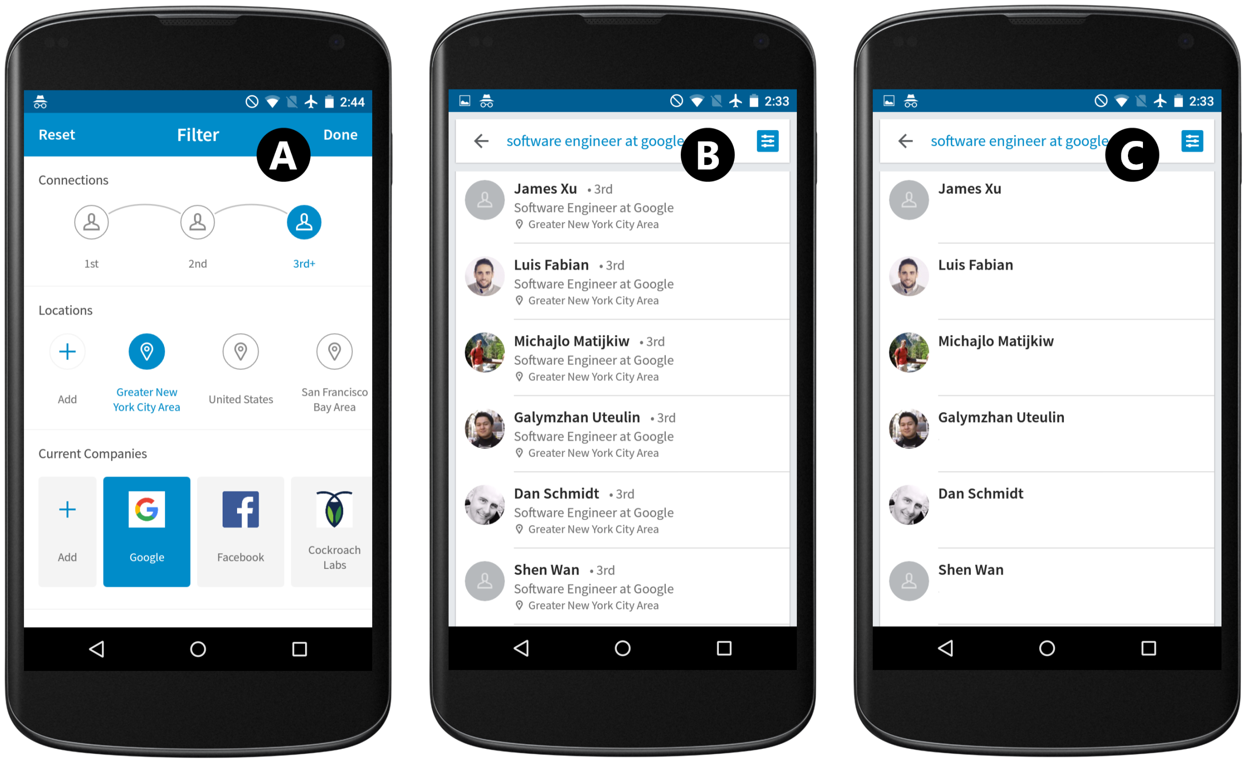
company, location, and profession filters and hits *"Search"* (Figure 1.4, A). The search engine returns the SERP where all results look the same (Figure 1.4, B). Moreover, the original seemingly information-rich SERP actually presents the very minimal amount of new information as indicated by blank areas on the modified SERP with the redundant information removed (Figure 1.4, C). Again Alice has to click on each result one-by-one following the inefficient "hub-and-spoke" SERP interaction pattern.

Eventually, having checked many profiles, Alice finds the people she met at the event, asks the necessary questions, and decides on her career move. She applies for several software engineering jobs in New York.

Although Alice's story is fictional, the tasks she engaged in and the attempts she made to satisfy her information needs illustrate the challenges faced by PSN users today. Based on her example, we can see that search functionality within PSNs is still very limited, which leads to siloed information and suboptimal search user experience. The SUI for PSN search resembles the one used for web search, which has minimal number of elements and is optimized to reduce complexity (a common belief among web search engine designers is that users are non-experts with very minimal computer skills, and hence, all intelligent work should be off-loaded to a search system [54], Chapter 1). We still have only a simplistic input interface allowing to submit keyword queries and a faceted search interface [132, 152] allowing to filter results based on attribute values. The presentation of results is far from perfect. Snippets are generated in a query-biased manner as it was also found to be useful in a web search scenario [131, 145]. There is a lot of redundancy on a SERP and it is hard to differentiate one search result from another. The structured and interlinked nature of PSN entities and unique aspects of structured search[5] are ignored or severely underutilized.

In this thesis, we present novel insights, algorithms, and recommendations that address these limitations and improve UX by increasing search relevance, adding more interactivity, and minimizing users' efforts.

[5]when both queries and data are structured and the matching is exact and not probabilistic.



James Xu • 3rd
Software Engineer at Google
Greater New York City Area

Luis Fabian • 3rd
Software Engineer at Google
Greater New York City Area

Michajlo Matijkiw • 3rd
Software Engineer at Google
Greater New York City Area

Galymzhan Uteulin • 3rd
Software Engineer at Google
Greater New York City Area

Dan Schmidt • 3rd
Software Engineer at Google
Greater New York City Area

Shen Wan • 3rd
Software Engineer at Google
Greater New York City Area

James Xu

Luis Fabian

Michajlo Matijkiw

Galymzhan Uteulin

Dan Schmidt

Shen Wan

Figure 1.4: The LinkedIn's Android mobile app in May 2016: (A) the structured query formulation interface with the query *"Software Engineer at Google; Location: Greater New York City Area; Degree: 3rd+"*: (B) the original SERP; (C) the same SERP with the redundant information removed from the snippets.

1.2 Thesis Statement

Search within PSNs is fundamentally different from web search and traditional information retrieval. First, PSN entities aren't independent of each other but form an entity graph, which opens up the opportunity to utilize the rich graph structure to perform navigation, filtering, and exploration of a social network. For example, we can search for an entity by expressing constraints on the entities connected to it. Moreover, since users represent a part of this graph, local graph (sub)structures become as important as the global structure [18]. Personalization is possible not only based on the history of interactions with a search system but also based on the neighbourhood structure in the social graph. Second, the units of retrieval in PSNs are structured and typed entities rather than documents. It allows to perform more accurate semantic analysis at all stages of the search process such as query formulation, result manipulation, result presentation, and ranking [19,112,123,146]. Queries could be more expressive and precise, snippets could be more informative, and PSN users could have more control over the search process. For example, we can perform exact matching, like in databases (DB), rather than probabilistic, like in information retrieval (IR). In turn, because all results exactly match a query, snippets are needed mostly to differentiate the results on the SERP and not to communicate their relevance. Traditional query-biased snippets create redundancy in the case of people/job search — information from a query is repeated as an attribute value for each search result (in web search, if the keywords from a query are repeated, the SERP is still informative since the content is diverse). Plus, specifically in the case of job search, job postings are quite regular (not applicable for documents on the web), and hence, rather than showing a generic summary, we can accurately parse job postings and generate more informative structured snippets helping address specific user tasks and needs. Third, sorting is possible not only by relevance (typical for web search) but also by an attribute value, e.g. sort by publication date or salary for job postings, by date for status updates or resumes. Finally, rather than providing services to a mass market, the PSNs' target audience are knowledge workers, who are more open to innovation and have the potential to expend cognitive and physical energy for increased search utility [83, 84, 157]. Therefore, there are many opportunities that we can leverage to facilitate more effective search within PSNs. We demonstrate that *by modeling the structured and interlinked nature of PSN entities, we can optimize the query-refine-view interaction loop, facilitate serendipitous network exploration, and increase search utility.*

1.3 Research Agenda

The primary motivation for this thesis is to improve search user experience within professional social networks. However, as a research agenda, we consider a more specific research question: *How can we optimize search user interfaces and interactions within professional social networks?* We focus on the SUI because of the following reasons. While a lot of work has been dedicated to mining social and information networks [51], effective ranking and recommendation algorithms [121, 122, 155], the optimization of SUIs within OSNs and PSNs, in particular, is still in its infancy. At the same time, several influential IR researchers noted that "system-driven IR" focused on indexing and ranking algorithms isn't enough to deliver high-quality search user experience and called for "user-centered IR". In his ECIR 2008 keynote Nicholas Belkin[6] stressed:

> *"...it is clearly the case that the new models and associated representation and ranking techniques lead to only incremental (if that) improvement in performance over previous models and techniques, which is generally not statistically significant (e.g. Sparck Jones, 2005); and, that such improvement, as determined in TREC-style evaluation, rarely, if ever, leads to improved performance by human searchers in interactive IR systems..."* [10].

[6]Gerard Salton Awardee 2015 for *"significant, sustained and continuing contributions to research in information retrieval"*.

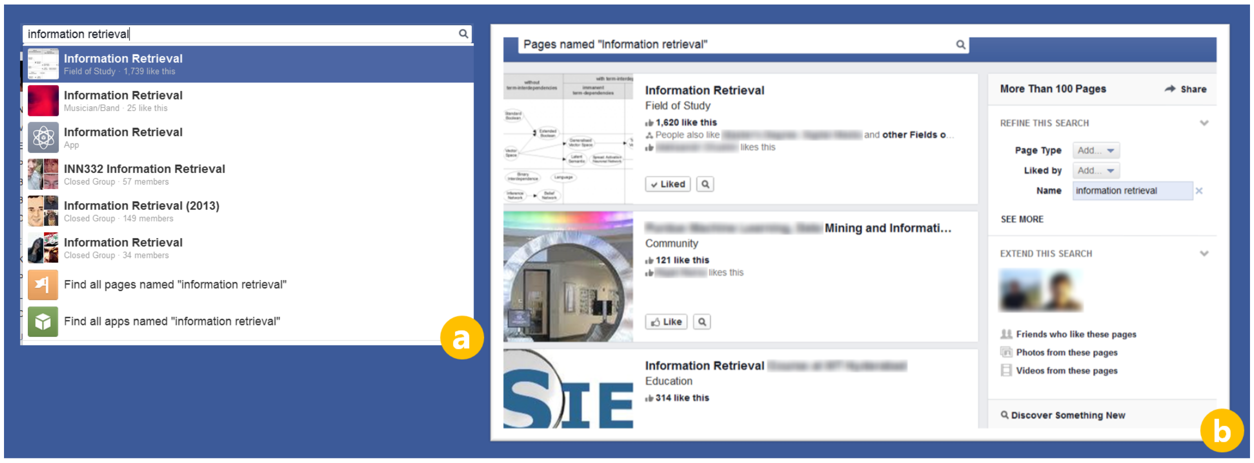


Figure 1.5: FBGS UI: (a) Typeahead, which performs query suggestion for both named entity and structured grammar queries. It handles named entity queries by sending directly to a clicked entity page. (b) Browse, which presents results for structured grammar queries as a standard list with links and snippets.

We contribute to this new emerging research topic and invite more IR researchers and practitioners to join the flock. We use the SUI design framework proposed in [112, 146] to frame the research questions.

1.3.1 Improving *Input* Elements of the Search User Interface

We start our investigation by focusing on query formulation with the goal to understand how can we design this element of the SUI in a native way taking into account the peculiarities of the search vertical (search within OSNs and PSNs). In 2013 Facebook introduced its innovative Graph Search product (Figure 1.5) aiming to take the search user experience to the next level and facilitate exploration of the Facebook Graph beyond the first degree. Uniquely, Facebook Graph Search (FBGS) allowed users to search for entities and relationships between entities and, following the terminology from [117, 153], supported: (a) *interactive free-text queries*, like "*Lady Gaga*" (navigational queries for one entity by name or Named Entity Queries, NEQs); (b) *interactive structured queries*, like "*Photos of people who live in China*" (exploratory queries for filtering entities with conditions on attributes; the instances of a rich query grammar or Structured Queries, SQs); (c) *one-shot free-text queries*, like "*query log mining*" (limited to users' status updates). In other words, FBGS supported more sophisticated queries and provided more control and agility than traditional SUIs. In turn, it demanded people to take responsibility for this control by expending cognitive and physical energy [83, 84]. This made FBGS the very system to study to deepen our understanding of how to search within a social network and guide the future design of interactive SUIs, especially on graphs.

Inspired by many existing query log studies and being in a unique position to have access to both search logs and the social graph, we designed this study around the following research questions:

- **RQ1:** How does search behavior differ for NEQs and SQs? For example, do people use NEQs to find different people compared to SQs?
- **RQ2:** How does search behavior depend on the graph search distance? For example, do people search more for friends compared to non-friends?
- **RQ3:** What search patterns are typical for users from various demographics? For example, are women or men more engaged with Graph Search? What influence does age have on search product usage?
- **RQ4:** How do people search for people using structured grammar queries?

The work is especially notable because FBGS is the first search engine, which offered structured querying capabilities for non-engineers at scale. We address **RQ1-RQ4** in Chapter 2.

1.3.2 Improving *Control* Elements of the Search User Interface

Following the information seeking journey, our second step is to improve the control aspects of the SUI. Specifically, we focus on results (re)sorting by an attribute value, which is important in the context of PSN search since jobs, people, and other PSN entities are structured and can be sorted based on many meaningful attributes different from relevance, e.g. sort job postings by time or salary or people by years of work experience or distance from an office location. As we described in Alice's scenario, nowadays PSNs address this aspect in a simplistic manner — they merely resort entities by an attribute value. However, sorting purely by an attribute value is not the best approach since at the top of the list users might find irrelevant results (Figure 1.2). It motivates the following two related research questions:

- **RQ5(a):** Can the quality of results sorted by an attribute value be improved by incorporating relevance into the ranking process?
- **RQ5(b):** What is the best way to accomplish it?

Apparently, by adding relevance, we can increase ranking quality. For example, we can combine relevance with other features of an entity using machine learning. The challenging part of this work is how to increase ranking quality and *satisfy the relative ordering constraints imposed by an attribute value to communicate to the user that she controls the SUI and the search process*, which is impossible to ask from a machine learning-based solution. In Chapter 3, we address **RQ5(a)** and **RQ5(b)** and propose the theoretically optimal algorithm to perform relevance-aware search results filtering that directly optimizes a given search quality metric and preserves strict ordering constraints imposed by an attribute value.

1.3.3 Improving *Presentation* Elements of the Search User Interface

The next step is to improve the presentation of search results. An ideal search system presents the best results at the top of the SERP. However, due to complexities of the natural language, such as lexical ambiguity and vocabulary mismatch, existing systems cannot guarantee perfect retrieval results. Therefore, relevant results might appear in any position on the SERP. To address this issue, search engines "cooperate" with the users via intelligent SUIs — on the SERP each result is represented as a title/name and accompanied with a snippet, which contains the key information about the result in a summarized form. The snippets serve primarily two purposes: (1) help assess result relevance; (2) help differentiate one result from another.

The de facto method to generate search snippets is based on the extraction of sentences or attributes containing query terms [60, 131, 136, 139, 145]. Initially introduced for full text search [131], this method is used nowadays for all sorts of search applications, data formats, and across a wide range of search verticals. However, such an egalitarian approach might be suboptimal, and some search verticals might benefit from dedicated solutions. By reasoning from basic principles and analyzing Alice's scenario, we identified that query-biased snippets aren't effective for job and people search — it is very hard to differentiate one result from another when such snippets are used. Moreover, when the matching is exact (e.g. faceted search), we observe query-snippet duality, i.e. *the longer is the query/filter set, the more redundant and less informative is the query-biased snippet.* This might demotivate users from submitting longer queries, which are known to be more effective [11]. Therefore, we explored new alternative snippet types, methods to generate them, and conducted user-centric evaluation studies.

Extended Informative Structured Snippets for Job Search

Inspired by the recent success of [151], who studied the problem of expert selection in the enterprise and found that extended snippets help employees find the right experts more effectively, and of [49], who introduced the concept of enhanced snippet for web search and demonstrated its utility via an ad-hoc user study and an A/B test, we proposed the concept of *extended informative structured snippet* for job search. The idea is to bring the most important information about a job (responsibilities and requirements) directly to the SERP and present it in a structured form. In this case, users will incur extra effort (since they will have to process more information), yet they will be able to scan this information faster, make more thoughtful choices, and click only on job postings that are genuinely attractive for them. In other words, the addition of such information to the SERP should result in faster and more effective search. This hypothesis became the driver for the project, and we formulated three main research questions to test it. First, we conducted the user study to de-risk the project and understand:

- **RQ6(a):** What information is necessary for users to decide whether they want to apply for a job?
- **RQ6(b):** What information is necessary for users to decide which results to click on a SERP?

The results of this user study turned out to be positive (the participants did want to see the information about responsibilities and, especially, requirements on the SERP; they consistently ranked these two attributes among the top most critical attributes of a job). Therefore, a more pragmatic question became relevant. How can we generate such snippets? One can build an information extraction model following the ideas from [94, 147]. However, despite aiming to minimize labeling efforts, this approach still requires some training to be conducted manually. The problem becomes more severe if we take into account the fact that major job search engines and PSNs operate internationally, and hence, training sets must be created for each language, which is costly. Ideally, we should be able to perform information extraction and generate structured snippets in an unsupervised way or with minimal supervision. It shapes two more research questions for this project:

- **RQ7(a):** How can we automatically extract job responsibilities and requirements from an unstructured job posting with minimal supervision?
- **RQ7(b):** How can we generate an extended informative structured snippet for a job posting having a list of job responsibilities and requirements in a structured form?

Finally, having built the algorithm to generate extended informative structured snippets, we have to evaluate the utility of such snippets. Therefore, the last two research questions in this project were:

- **RQ8(a):** Do extended informative structured snippets improve search user experience for job search?
- **RQ8(b):** How do users behave when such structured snippets are used?

We address **RQ7-RQ9** and describe three stages of this project (the user study, the algorithm design, and the user-centric evaluation) in Chapter 4.

Non-redundant Delta Snippets for Job and People Search

Researchers proposed multiple approaches to generate non-redundant and discriminative snippets. The main idea behind these approaches is that we can utilize space on the SERP more effectively by showing in the snippets information that helps differentiate the results better, and hence, complementary to the query. For example, there are methods to generate informative [29, 30] and diverse [79, 91] snippets for tuples from a relational database and for XML search [59]. Recently, [49] demonstrated the advantages of enhanced structured snippets for web search.

Despite the fact that query-biased snippets are suboptimal from a purely information-theoretical perspective (the information density per pixel of screen space is higher for non-redundant snippets), many popular (structured) search engines today still use them in practice (e.g. Figure 1.3 and Figure 1.4). There is a possible explanation which is related to the human factor. According to the SUI design guidelines proposed in [112], we should: (1) *strive for consistency* so that *users* could comfortably switch from one search engine to another without loss of productivity; (2) *offer informative feedback* so that *users* are informed about all aspects of the search they are preparing to do (the sources, fields, what is being searched for, and what variants are being allowed); (3) *reduce short-term memory load* to help *users* accomplish their search tasks with minimal efforts and higher satisfaction. Query-biased snippets satisfy these design guidelines — most of the search engines today use query-biased snippets; query-biased snippets explicitly repeat information from a query making users confident that the results match the query constraints; with query-biased snippets rather than trying to recall the query constraints, users only engage in the recognition process, which is known to be less cognitively demanding [41].

To summarize, there is an inconsistency between theory and practice and between database and human-computer interaction research. On the one hand, there are very powerful methods to generate non-redundant, informative, and discriminative snippets for structured data. Unfortunately, none of them were evaluated in a user-centric fashion. On the other hand, query-biased snippets are used in many search engines and satisfy seminal SUI design guidelines. We can only speculate which version is better.

In Chapter 5, we resolve this inconsistency and provide new knowledge for the development of more effective and useful SUIs within and beyond PSNs. Specifically, we consider non-redundant delta snippets, which show information relevant but complementary to the query, as an alternative to traditional redundant query-biased snippets and pose the following research questions:

- **RQ9:** What kind of snippets make users more productive and effective when performing structured search (e.g. job/people search) on mobile devices? (objective evaluation)
- **RQ10:** Do users prefer non-redundant delta snippets or query-biased snippets based on their subjective feelings? (subjective evaluation)

While the problem in question is relevant for both web and mobile devices (e.g. Figure 1.3 and Figure 1.4), we focus on mobile search because mobile devices have smaller screens and, hence, it is more critical to understand whether showing redundant query-biased snippets is really necessary. Plus, mobile applications and search are getting more and more popular among users [34, 97].

1.4 Key Contributions

Taken together, this thesis is a step forward to the better search within PSNs. It could have a massive positive impact on the lives of people making the search for a dream job or professional career advice more effective. Broadly speaking, this thesis provides new insights and techniques for the design of search engines on top of structured and networked data. In particular, the thesis makes the following key contributions:

- **A large scale analysis of Facebook Graph Search query logs.** Having access to anonymized query logs, an anonymized social graph, and an anonymized set of user profiles, we examined how search behavior differs for different query types, how it changes for users from various demographic groups, and how it depends on the graph distance between a searcher and a person to be searched. The analysis revealed many exciting findings and suggested numerous design implications. For instance, structured query usage behavior has a wider variation across different demographics, and hence, it makes sense to focus search personalization efforts on this query type; users search more for friends using named entity queries and for non-friends using structured queries, which shows their complementary roles in enabling effective search. The crux of this work is that users do benefit from more control and highly interactive query suggestions — they engage in a new type of search behavior (exploratory search for non-friends). See Chapter 2 for details.

- **An algorithm for relevance-aware search results filtering**, which addresses the problem that nowadays users see many irrelevant results at the top of the SERP when they select sorting by an attribute value, e.g. salary, date, price, etc. It is an efficient and theoretically optimal algorithm based on the dynamic programming [12], which directly optimizes a given search quality metric, like for the relevance-based sorting order. According to our extensive experiments, the algorithm outperforms all baselines and leads to 2-4% increase in search quality. See Chapter 3 for details.

- **A user study and an algorithm for extended informative snippet generation for job search.** To motivate this project, we conducted the user study using survey and interview methods. Then, we designed the effective algorithm. The algorithm allows generating informative snippets that contain the key information (responsibilities and requirements for a job) right on the SERP and help users make clicks mostly on relevant results. Plus, it helps optimize content for mobile devices and avoid irregularities of job postings coming from multiple websites by converting them into the structured representation. The algorithm leverages the power of big data to minimize supervision required for model training (2-3 words per language) and could be easily deployed by a job search engine operating internationally. We conducted a series of offline and online A/B experiments and found that: (1) the algorithm achieves high extraction accuracy (86% precision at 94% coverage for English language and 97% precision at 100% coverage by a highly-tuned enterprise-grade model for the Russian language); (2) extended informative structured snippets improve the majority of search quality metrics and decrease SERP click entropy, which implies that we can collect accurate data to further enhance search utility (e.g. via relevance feedback) with minimal intervention. See Chapter 4 for details.

- **A comparative user study of query-biased and non-redundant delta snippets for structured search on mobile devices**. To investigate what kind of snippets are better suited for structured search on mobile devices, we built an experimental mobile search application and conducted a task-oriented interactive user study. Four different versions of the SERP were compared by varying the snippet type (query-biased vs. non-redundant) and the snippet length (two vs. four lines per result). We adopted a within-subjects experiment design and made each participant do four realistic search tasks using different versions of the application. During the study sessions, we collected search logs, "think-aloud" comments, and post-task surveys. Each session was finalized with an interview. We found that with non-redundant delta snippets the participants were able to complete the tasks faster and find more relevant results. The participants preferred non-redundant snippets more and wanted to see more information about each result on the SERP for any snippet type. At the same time, the participants felt that the version with query-biased snippets was easier to use. Based on the study results, we proposed a set of design recommendations on how to improve structured search (job and people search within PSNs), especially on mobile devices. See Chapter 5 for details.

1.5 Thesis Outline

The rest of the thesis is organized as follows. In Chapter 2, we present the query log analysis study of Facebook Graph Search. In Chapter 3, we present the algorithm for relevance-aware search results filtering via a direct optimization of search quality metrics. In Chapter 4, we describe the user need elicitation study, the algorithm for extended informative structured snippets generation for job search, and the A/B experiment conducted to evaluate the utility of such snippets. In Chapter 5, we describe the user study in which we compared the utility of query-biased and non-redundant delta snippets for structured (e.g. people) search on mobile devices. We conclude and present ideas for future work in Chapter 6.

Chapter 2

Named Entity and Structured Queries for People Search

In this Chapter 2, we analyze large scale anonymized query logs generated by users of Facebook Graph Search with the aim to understand whether structured querying capabilities are beneficial for the users of PSNs and answer our **RQ1-RQ4**. First, in answering **RQ1** (*"How does search behavior differ for NEQs and SQs?"*), we studied named entity and structured query usage behavior and established their complementary nature showcasing the importance of each query type for enabling flexible search and exploration within PSNs. Second, in answering **RQ2** (*"How does search behavior depend on the graph search distance?"*), we shared unique insights about people search on Facebook related to the graph search distance via anonymized query logs and an anonymized social graph mining. Third, in answering **RQ3** (*"What search patterns are typical for users from various demographic groups?"*), we performed privacy-preserving demographic profiling and discovered demographic-specific people search patterns. Finally, in answering **RQ4** (*"How do people search for people using structured grammar queries?"*), we presented insights about the structured grammar usage, which are unique to the Facebook Graph Search product but have potentially much broader application. By the end of this Chapter 2, we will better understand how to design interactive query formulation interfaces for PSNs and, more broadly, for structured networked data.

In the next sections, we cover related work (Section 2.1), necessary definitions and background material (Section 2.2); describe the data sets (Section 2.3); and summarize high level properties of logs (Section 2.4). Starting from Section 2.5, we share many insights about Facebook Graph Search usage. We present design implications relevant for the design of future SUIs and interaction techniques within PSNs in Section 2.7 and discuss the limitations of this work, thereafter.

2.1 Related Work

Research on people search has a long history. Early works studied ways to automatically assign reviewers to papers [38] or find topical experts [118] by representing people via documents they are associated with and framing people search as a traditional information retrieval or recommendation problem. However, as [129] noticed, it fundamentally changes the nature of the problem when the objects are people rather than documents. People form social relationships, and therefore, ranking people is qualitatively more complex than ranking textual documents. As a result, [129] proposed the concept of social matching to emphasize the social dimension. In turn, that led to the research and development of social matching systems, such as Referral Web [71], Expertise Recommender [87], and Aardvark [57], which return relevant people taking into account the social similarity between a candidate result and a searcher. [1] studied the fundamental properties of social networks making social search possible and concluded that "*where the data is incomplete or reflects non-hierarchical structure, tools that support social search should assist users by either providing a broader view of their local community or directly assisting users through a global analysis of the network data*" pointing out to the importance of search within online social networks (OSNs). Recently, exploratory people search was investigated in the context of a PeopleExplorer project [52], which allows users to explicitly model their search preferences using sliders. Based on extensive experiments, the authors concluded that it is crucial to model task difference and user variance in people search.

Looking at the related work from a query log mining perspective, there is a large body of research dedicated to the analysis of the web search engine usage. Starting from the influential taxonomy of web search queries [19], researchers studied query logs to understand how users search, analyzing the length [11, 63, 113, 128], topical distribution [9, 116], and temporal patterns [9] of queries. Query logs were also used to understand search sessions [68] and re-finding [126, 134].

Because we are exploring people search, which is a type of vertical search operating in the people vertical, it is important to consider studies of vertical search engine logs. [92] studied queries issued to a blog search engine, and found that people were particularly likely to search for named entities, e.g. people and blogs on a topic of interest. [120] compared blog queries with news queries, observing that queries often refer to people and temporally relevant content. [128] compared microblog search and web search, and found that Twitter users similarly search for temporally relevant information and people. A study of a web people search query log [143] revealed that a significant number of users type just one query, that people search has lower click-through rates (CTR) compared to web search and that the most popular results come from social media (OSNs). Recently, people search behavior and the role of graph distance in name and non-name queries was explored using the LinkedIn log [58]. It was reported that for name queries users primarily click on only one of the results and a shorter graph distance leads to higher CTR, while for non-name queries users are more likely to click on multiple results that are not among their existing connections, but with whom they have shared connections, i.e. the second degree connections.

Search logs were also used to uncover relationships between the search behavior and demographic char-

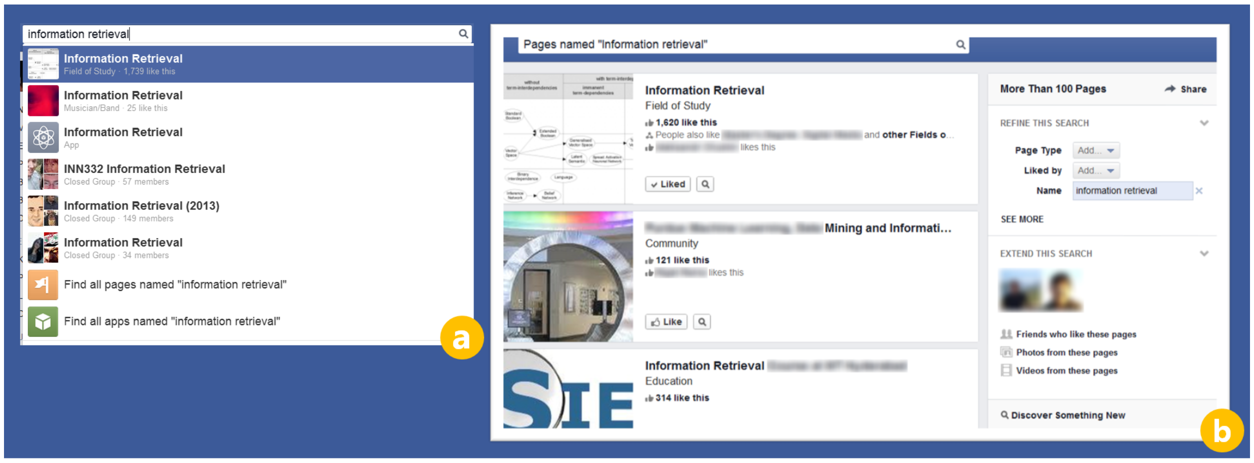


Figure 2.1: Facebook Graph Search user interfaces: (a) Typeahead, which performs query suggestion for both named entity and structured grammar queries. It handles named entity queries by sending directly to a clicked entity page. (b) Browse, which presents results for structured grammar queries as a standard list with links and snippets.

acteristics of users. [14, 140] described the methodologies for usage of query logs and demographic profiles for search personalization. [142] presented the demographic-specific insights about search sessions and query topics. Several studies explored the influence of gender and age on search behavior. It is reported based on a series of interviews that males used search engines more than females in 2004 [44], but in 2012 the numbers equalized [101]. By analyzing web search engine usage, [80] found that females write longer queries than males. [36] investigated the search behavior of users retrieving information for children.

The work described in this Chapter 2 complements and extends existing studies from three perspectives. First, while there is a large body of work on the topic of people search from an algorithmic side, there are only two people search query log studies [58, 143] and only one of them is about search within a PSN [58]. We continue this line of work and answer previously untouched questions, e.g. "How does search behavior vary with age?". Additionally, we present novel insights about the structured grammar usage unique to Facebook Graph Search, e.g. "What are the strategies for a name disambiguation?". Second, we extend the research on social search by studying query logs of an entity search system. Existing query log studies in the blogosphere [92], web [143], and Twitter [128] focused on document search systems. Third, our analysis uses logs generated by the system supporting several different query types, and therefore, we can do cross-type search behavior comparisons. Typically, search systems support only one query type.

2.2 Background Information

In this section, we describe the specifics of the search user interface and provide the necessary definitions.

2.2.1 Structured Search User Interface

The search process starts by navigating to a Typeahead interface (Figure 2.1(a)). In the null state, before any symbol is typed, Typeahead presents 7-8 query suggestions personalized for a searcher. These might include named entity queries or simple structured grammar queries. On every keystroke, the Typeahead results get updated, and the user is presented with the input-dependent suggestions. At any point in time, the user can pick a relevant result among the suggested options or keep typing. Suggestions, representing named entities, serve as search results directly, and upon click redirect to the corresponding entity page. Structured grammar queries lead to a standard search engine results page, called a Browse interface (Figure 2.1(b)), where the user is presented with a list/grid of entities matching the query conditions. For example, "*Lady Gaga*" is a named entity query, which upon click will navigate a searcher to the page maintained by Lady Gaga. "*People who like Information Retrieval and live in USA*" is an example of a grammar query, with three predicates (`people`, `like`, `residents`) and two entities (*USA*:Country, *Information Retrieval*:Field_of_Study). To effectively explore entities on Facebook, users may construct sophisticated structured grammar queries by concatenating predicate-entity pairs using a boolean `AND` operator. The search engine supports queries for entities in dozens of categories such as *Apps*, *Pages*, *People*, *Posts*, and others.

2.2.2 Definitions

Before we proceed to the analysis, it is useful to define some terminology. To make the definitions clear, where necessary we reference an example shown in Figure 2.1.

Person: In our case, people are represented as entities from: (a) a $User$ category with all standard capabilities, such as friending, commenting and so forth; (b) human-like subcategories of a *Page* category such as *Athlete*, *Music Band*, *Politician*, and others – a broadcast-style account, typically used by celebrities to interact with their fans. One individual could have a $User$ account and be an admin for several *Pages*. Unless otherwise stated, in the rest of this Chapter 2 we focus on queries for Person from the $User$ category. We only consider queries for Person from the *Page* category while discussing celebrity search in Section 6.

Celebrity: According to [85,133], a celebrity is a "*highly visible in the media and overly public individual, who usually has emerged from the entertainment or sports industry and whose private life attracts greater public interest than the professional life*". For consistency, we define a celebrity as a Person with more than 10000 friends, fans, or followers.

Functional Predicate: In typed logic, F is a functional predicate with a domain type T and a codomain type U if, given any object X of type T, $F(X)$ is an object of type U [62]. The type of the predicate coincides with the codomain type. Similarly, a query type has the type of the result. Search grammar

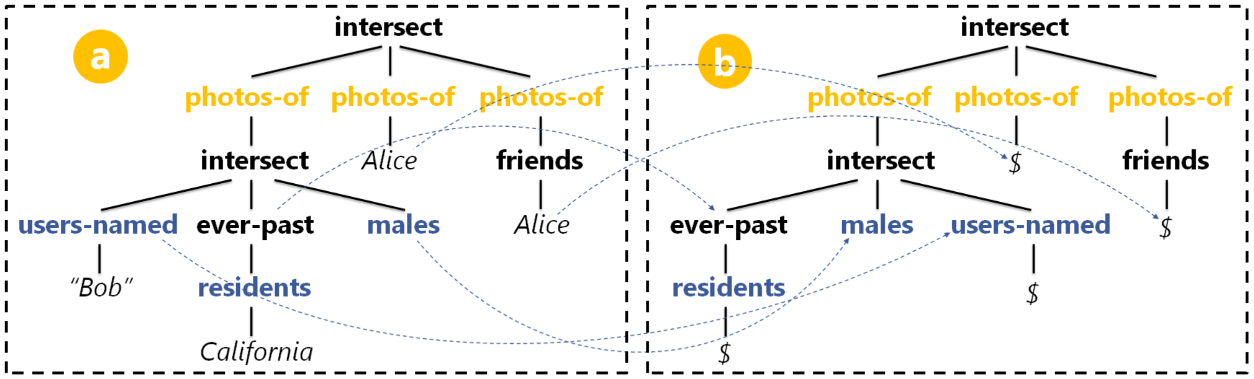


Figure 2.2: A grammar query (a) after a few transformations becomes a Semantic Query Template (b).

consists of numerous functional predicates, which perform typed mappings defined on the entities. For example, `friends` is a functional predicate that, given an entity of type *User*, produces a set of friends for this user and, hence, is a *User* predicate; `photos-in`, given an entity of type *Location*, produces a set of photos taken in that location and, hence, is a *Photo* predicate.

Semantic Query Template: Grammar queries consist of keywords, entities, and functional predicates, which can be combined using a boolean `AND` operator and functional superposition. An example query "*Photos of Alice and friends of Alice and males named Bob who live in California*" has a parsing tree shown in Figure 2(a). It has functional predicates as inner-nodes and keywords (a quoted string) and entities of types *User* and *State* as leaves. Because Facebook Graph Search is a highly personalized search engine, almost any two query parsing trees are different. However, the semantics of these queries might be similar. For example, users may search for their friends by name, but names of the friends are likely to be different. Therefore, to study search patterns at a more general level, we categorize all grammar queries into factor classes [82] based on the structure of the corresponding parsing tree. First, we replace all leaves with the generic sentinel, e.g. "*Alice*", "*Bob*", and "*California*" are mapped to "*$*". Second, we sort tree nodes level-by-level using a lexicographic ordering on node names. Such factor classes are called Semantic Query Templates and each query goes to one class. An example is illustrated in Figure 2.2.

Graph [Search] Distance: The entities of all types (vertices) and various relationships between them (edges) form a Facebook Entity Graph (a Social Graph is an undirected subgraph of the Entity Graph made of only *User* vertices and `friends` edges). We use a traditional graph-theoretic definition of the graph distance as the minimal number of edges connecting two given vertices [16]. We categorize all *User* queries into three major classes based on the graph distance between a searcher and a result:

- *Self*: queries for herself/himself;

- *Friend*: queries for friends, relatives, and many other entities connected to the searcher by an edge;
- *Non-friend*: out-of-network queries for friends of friends and many other entities not connected to the searcher.

For example, a user searching for a friend of a friend by name, e.g. "*John Smith*", does a *Non-friend* query; on the other hand, *"My Friends who live in California and like Computer Science"* is a *Friend* query.

It is important to note that while named entity queries are not ambiguous based on the categorization above, the degree of a structured query is not strictly defined since it might contain multiple entities and various predicates. Therefore, for structured grammar queries we use the following distance calculation algorithm. We only consider grammar queries involving at least one $User$ entity, such as "*Photos of **User1** and **User2***" or "*Places visited by **me***". We then look at each entity involved in a query and assign it a distance using the Social Graph and functional superposition of $User$ predicates. Finally, we compute a bit vector with the three components, one for each of the three classes of graph distance, and normalize it by the number of non-zero components. Therefore, a grammar query might contribute to the weighted count for each of the graph distances.

For clarity, let us apply this algorithm to the query in Figure 2.2(a) by enumerating hypothetical searchers. Each $User$ entity participates in a path from the root to this entity in the parsing tree: `intersect` → `photos-of` → `friends` → *Alice* and `intersect` → `photos-of` → *Alice*. If the searcher is Alice, the output vector is $(0.5, 0.5, 0)$, a half for herself and a half for friends of Alice. If the searcher is a friend of Alice, the output vector is $(0, 1, 0)$ because both paths are about Alice, who is one edge apart from the friend. If the searcher is not a friend of Alice, the output vector is $(0, 0, 1)$.

This and other custom data processing pipelines were implemented as MapReduce jobs [32]. Simple aggregation statistics were computed using Hive [130].

2.3 Data Sets

We study people search behavior using four data sets collected at Facebook: (a) one which gives insight into the navigational search – an anonymized log of named entity queries; (b) another which gives insight into the exploratory search – an anonymized log of structured queries; (c) one which allows us to calculate graph distances – an anonymized Social Graph; (d) one which allows us to discover demographic-specific patterns – a set of anonymized User Profiles. Query logs were collected in the second half of 2013. The Social Graph and User Profiles were captured on 2013-10-17. The sizes of the data sets are given in Table 2.1.

Data Set	Attribute	Count
Named Entity Query (NEQ) Log	Users	$3M$
	Queries	$58.5M$
Structured Query (SQ) Log	Users	$3M$
	Queries	$10.9M$
Social Graph	Vertexes	$858M$
	Edges	$270B$
User Profiles	Users	$858M$

Table 2.1: Basic statistics about the data sets used.

2.3.1 Named Entity Query (NEQ) Log

The log contains named entity queries for *Person* from three million randomly sampled `en_US` (English, USA) users, who made at least one such query both in the month preceding the study and during the study periods. This allowed us to level novelty effects and observe a stable search behavior representative of a general population. Finally, after filtering we worked with 58.5 million queries. Each query log record includes such fields as an anonymized *id* of a searcher, an anonymized *id* of an entity to be searched, a time stamp, an entity type, and the query metadata.

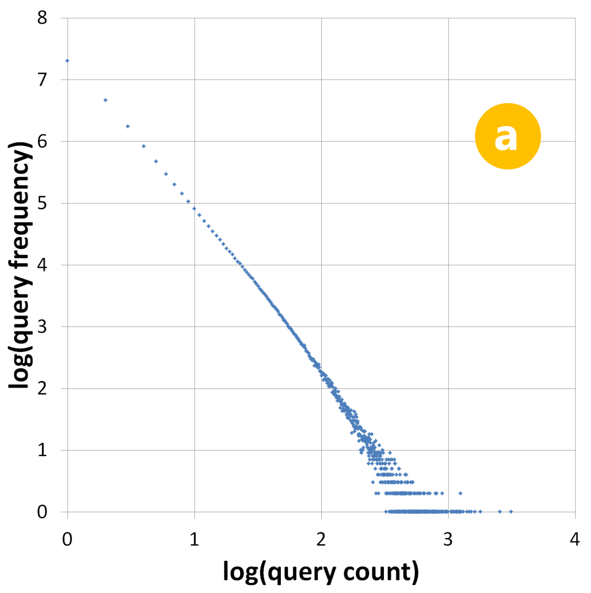
2.3.2 Structured Query (SQ) Log

The log contains structured grammar queries of three million randomly sampled `en_US` (English, USA) users, who made at least one such query both in the month preceding the study and during the study periods. Unlike named entity queries, which are handled purely by Typeahead and always require a query writing or a suggestion selection, each structured grammar query and the corresponding Browse search engine results page (SERP) gets a unique URL, which can be shared and accessed without writing a query. We excluded such records from the log to make sure that it contains only authentic user-generated queries. Finally, after filtering we worked with 10.9 million structured grammar queries.

Users in the Structured Query Log are independent of the users in the Named Entity Query Log. Any overlaps are due to coincidence. Although the samples are independent and we don't join them, this doesn't prevent us from discovering general insights for both query types.

2.3.3 Social Graph and Demographic Profiles

We used a snapshot of the anonymized Social Graph made of 858 million entities and 270 billion edges. To gauge an understanding of search patterns for different demographic slices, we used an anonymized data set with the four user profile attributes: age, gender, number of friends, and celebrity status. Using information from the profiles, we performed checks for representativeness of our samples by comparing



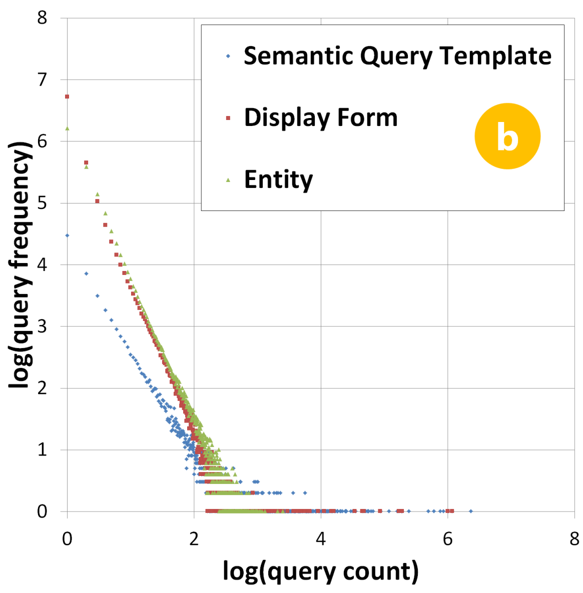


Figure 2.3: Power law graphs for query frequency of Named Entity (left) and Structured (right) Queries.

the key statistical properties of attribute value distributions for a sample and all 858 million profiles. The differences were insignificant because of the large sample size, which aligns with the reasoning on significance for big data [140].

2.4 First Order Analysis

Characteristic of online usage behavior, we observed power law distributions for query popularity and user activity. Figure 2.3(a) shows the query frequency distribution in a log-log scale for NEQs, which follows a power law with the slope $\alpha = 2.63$. Figure 2.3(b) shows three query frequency distributions in a log-log scale for SQs. Each graph corresponds to a different definition of a unique grammar query: (a) one which at a display form level (the query string), e.g. "*Photos of **Alice** and **Bob***", has the slope $\alpha = 2.38$; (b) one which at an entity level, e.g. {***Alice***, ***Bob***}, has the slope $\alpha = 2.08$; (c) one which at a semantic template level, e.g. "*Photos of **$** and **$***", has the slope $\alpha = 1.15$. As we can see, most of the queries are issued only a few times; however, there are some very popular queries. The power law also holds for user activity with the slopes $\alpha = 2.43$ for NEQs and (a) $\alpha = 2.08$, (b) $\alpha = 1.90$, and (c) $\alpha = 1.13$ for SQs, which shows that there are some very avid searchers. This observation is predictable and aligns with existing log studies [5, 76, 143].

To shed light on the semantics of the most popular people queries, we computed the frequency distribution over *Person* subcategories of the *Page* category for the top-1000 celebrity *Page* queries for both NEQs and SQs, shown in Table 2.2. Both distributions are quite similar to each other and follow a power law. Musicians,

Rank	NEQs	Percentage of Queries	SQs	Percentage of Queries
1	Musician/Band	**32.2%**	Musician/Band	**27.7%**
2	Public Figure	**19.4%**	Actor/Director	**26.9%**
3	Actor/Director	**17.8%**	Public Figure	**19.1%**
4	Entertainer	8.2%	Entertainer	8.4%
5	Artist	7.4%	Artist	6.4%
6	Athlete	7.3%	Athlete	5.2%
7	Character	2.5%	Character	2.3%
8	Comedian	2.2%	Politician	1.8%
9	Politician	1.9%	Author	1.3%
10	Author	1.1%	Comedian	0.8%

Table 2.2: Semantics of the top-1000 celebrity queries.

public figures, and actors represent the three most popular celebrity *Page* categories.

Contributing to the line of work on repeat queries [106, 126], we looked at the query frequency from a different perspective and computed an average query repetition ratio per user, i.e. a fraction of unique queries out of all queries. Since in web search navigational queries are repeated differently from others [106], it is worth seeing how NEQs and SQs compare to each other in our scenario. We found that the repetition ratio for NEQs is 0.56. For each definition of a unique SQ, the repetition ratios are (a) 0.72, (b) 0.62, and (c) 0.47, respectively. Therefore, users *search for different people using SQs more than NEQs* and *repeat Semantic Query Templates to learn about different people.* Surprisingly, we found that the repetition ratios both for NEQs and SQs stay the same for all three classes of queries based on the graph distance. One interesting implication from it is that users repeatedly search for non-friends without adding them as friends. The repetition ratios stay the same for various demographic slices.

2.5 Graph Search Distance

Unique to this study are the insights about people search for various demographics and graph distances. The graph distance is the key parameter to quantify users' tendency for OSN exploration via search. The distribution over the graph distances is presented in Table 2.3. Other alternatives to quantify the OSN exploration via search are the number of unique queries or the repetition ratio (see previous Section 2.4).

Users search for friends using NEQs and for non-friends using SQs. This demonstrates the importance and utility of having both query types to enable more effective exploration of the social graph. *Self* queries are negligible compared to an overall query volume, yet there is a significant difference between NEQs and SQs *Self* queries. Users search for themselves more using SQs. We think that this is because SQs are used to curate personal data published on Facebook, like "*My Photos*", while there are many other ways to navigate to a personal profile beyond using NEQs for yourself.

Query Type	Self	Friend	Non-friend
NEQs	0.6%	**57.6%**	41.8%
SQs	**5.2%**	31.2%	**63.6%**

Table 2.3: Query distribution over graph distances.

2.5.1 Influence of Demographic Characteristics on Graph Search Distance

Drilling down, we study people search behavior for different demographic slices. Among many available options, we picked the four attributes of the user profile: age, gender, number of friends, and celebrity status. Age and gender were used in multiple existing log studies [14, 37, 140–142], and it is interesting to compare their findings with the ones for Facebook. Number of friends and celebrity status are new attributes unique for this log study, which allow to capture the social dimension specific to search within OSNs.

We present a series of figures for each of the attributes in question. In the first column, we show fractions of the *Friend* queries out of all non-*Self* $User$ queries for a set of bins; in the second column, we show the search trends for Named Entity $User$ queries; in the third column – for Structured $User$ queries. We focus on *Friend* and *Non-friend* queries, since $Self$ queries are not common.

Age

Figure 2.4 presents how the graph distance varies with age. Looking at Figure 2.4(b), we see that *Friend* queries are more popular among NEQs for all age bins. On the contrary, according to Figure 2.4(c) the graph for SQs is bi-modal. While *Non-friend* queries prevail for the younger group of users, *Friend* queries prevail for the older group of users. One can also notice a clear cyclic pattern in a combined Figure 2.4(a), which depicts the fractions of *Friend* queries out of all non-*Self User* queries for NEQs and SQs. Both graphs have one valley for users in their $30s$ and one peak for users in their $70s$, which shows that the younger users more actively search for *Non-friends* and the older users more actively search for *Friends* relative to an average user. The graph for SQs has a higher variation than the graph for NEQs suggesting that the search personalization is more beneficial for SQs than NEQs.

Gender

It is reported in [44] that males searched more than females in 2004 but in 2012 the usage of search engines converged to the same level [101]. At the same time, females communicate more and use the web less than males [61, 74]. We compared the search usage for female and male users in our case and found three insights. First, females write more queries than males (Figure 2.5(b,c)). Second, females more actively search for *Friends* using NEQs compared to an average user (Figure 2.5(a)). The fraction of *Friend* NEQs to the

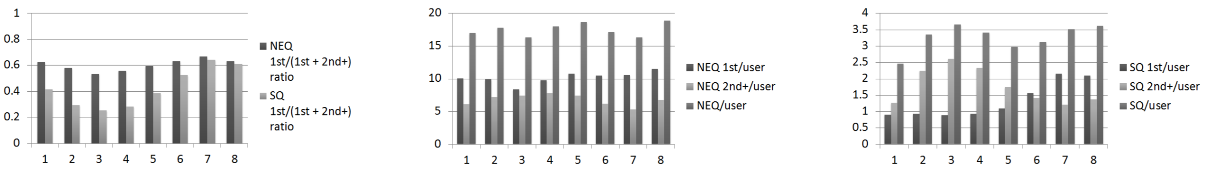


Figure 2.4: Search distance vs. Age (ten-year bins): (a) fraction of 1st degree queries out of all non-*Self User* queries; (b) average number of NEQs per user; (c) average number of SQs per user.

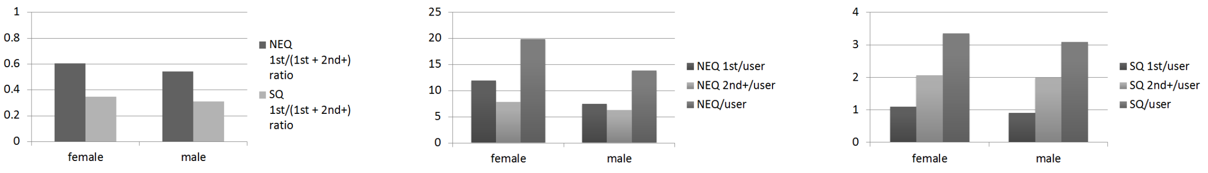


Figure 2.5: Search distance vs. Gender: (a) fraction of 1st degree queries out of all non-*Self User* queries; (b) average number of NEQs per user; (c) average number of SQs per user.

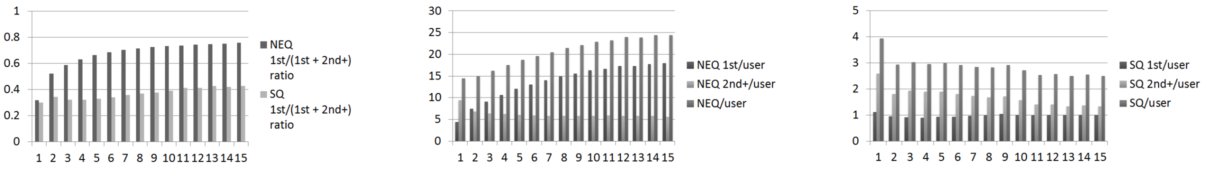


Figure 2.6: Search distance vs. Number of searcher's friends (100-friend bins): (a) fraction of 1st degree queries out of all non-*Self User* queries; (b) average number of NEQs per user; (c) average number of SQs per user.

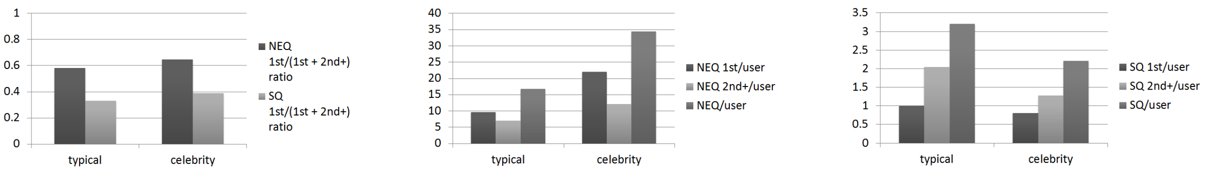


Figure 2.7: Search distance vs. Celebrity status:(a) fraction of 1st degree queries out of all non-*Self User* queries; (b) average number of NEQs per user; (c) average number of SQs per user.

sum of *Friend* and *Non-friend* NEQs for females is 0.605, while for males it is 0.542. Third, according to Figure 2.5(a), we also found that males search more actively for *Non-Friends* using SQs compared to the mean for the population. The fraction of *Friend* SQs to the sum of *Friend* and *Non-friend* SQs for males is 0.328, while for females it is 0.348. The ideas based on these insights are discussed in Section 2.7.

Number of Friends

In Figure 2.6 we present how the graph search distance depends on the number of friends. First, note that the ratio of *Friend* to *Friend* and *Non-friend* $User$ queries reaches its saturation level at around 0.75 for NEQs and 0.43 for SQs (Figure 2.6(a)). Interestingly, the graph for SQs almost reaches its saturation already for the first bin (less than 100 friends), while the graph for NEQs grows steadily and saturates at around 10th bin (1000 friends). Second, the more friends a user has, the more *Friend* NEQs the user writes (Figure 2.6(b)). This suggests that users actively use NEQs to find information about their existing friends. On the contrary, the trend for *Non-friend* NEQs declines slightly with more friends. Therefore, we speculate that in this case people have already "friended" a good amount of users they want in their network. Third, it is worth noticing the volatility of graphs for different graph distances. The trend for *Non-friend* NEQs is flat, while *Friend* NEQs contribute to the growth of the query volume (Figure 2.6(b)). The trend for *Friend* SQs is flat, while the volume of *Non-friend* SQs changes depending on the number of friends (Figure 2.6(c)). This again suggests that NEQs are fundamental for searching for friends and SQs are fundamental for searching for non-friends. Together these two query types enable both navigational and exploratory people search.

Celebrity Status

In Figure 2.7 we extend the analysis presented in the previous section and consider how celebrity status influences search behavior. This is an important question to study since search effectiveness and, hence, usage could be different in that extreme case (larger search space of friends). Because having $10,000$ friends is a rare event, there may only be a few celebrities in the original sampled query logs. Therefore, we considered a complete set of non-novice celebrity users of Facebook Graph Search and analyzed their anonymized queries during the same time intervals as for the original NEQ and SQ logs. According to Figure 2.7(b,c), on average celebrity users submit more NEQs and fewer SQs than typical users, which is in agreement with the findings presented in the previous section regarding the correlations between the number of queries and the number of friends. Moreover, from Figure 2.7(a) we conclude that the ratio of *Friend* queries to the sum of *Friend* and *Non-friend* queries for celebrities is biased towards the first degree connections compared to typical users. These findings suggest that celebrities are not as interested in exploring the graph and mostly use Facebook Graph Search for navigational purposes.

We deepen the analysis further by segmenting queries into celebrity and typical groups analogous to the approach we used for the searchers, i.e. we check whether a query contains an entity which has at least 10000 friends or fans. We computed the ratios of celebrity queries out of all non-*Self* $User$ queries for NEQs

Type	Ratio	NEQs	SQs
Typical	Celebrity/Typical among friends	0.001	0.001
	Celebrity/Typical among non-friends	0.009	0.002
Celebrity	Celebrity/Typical among friends	0.167	0.067
	Celebrity/Typical among non-friends	0.247	0.123

Table 2.4: Relationship between a celebrity status of a queried entity and a celebrity status of a searcher.

and SQs, two demographic groups (typical and celebrity), and graph distances (*Friends* and *Non-friends*). The results are presented in Table 2.4. It is worth highlighting that this experiment and the corresponding insights are about searches for celebrities as $Users$ and not about users searching for celebrities as $Pages$.

According to Table 2.4, celebrities search more for other celebrities than typical users. To see whether this was a result of celebrities having more celebrity friends, and therefore, simply searching for friends, we calculated the average number of celebrity friends for a celebrity and found that it is 0.016. The ratio of celebrity queries is higher than this, i.e. celebrities search for other celebrities disproportionally more than for typical users. Two other interesting findings are that both typical users and celebrities are more likely to search for a celebrity when they write a *Non-friend* query relative to a *Friend* query and using NEQs relative to SQs. Therefore, the surplus might be achieved by suggesting *Non-friend* celebrity NEQs to Facebook users in Typeahead.

2.6 Structured Grammar Usage

A prior query log study of a web people search engine [143] reports that users write queries with additional keywords to disambiguate a name of a person to be searched or to find relationships between people. The most used keywords were city names, jobs, and activities. However, the advanced functionality of that system was limited only to such keywords. We push this line of work forward and share insights about people search on Facebook, where users can write non-ambiguous grammar queries using a rich set of predicates.

2.6.1 Finding People Using Functional Predicates

To understand how SQ length depends on the query popularity, we computed the average number of functional predicates for top-100 and top-1000 most frequent semantic query templates (SQ-A). We didn't use traditional definitions of the query length such as the number of letters or words [11, 63, 113] because structured queries consist of indelible entities and predicates. We additionally computed the lengths of SQs with at least one (a) `friends` predicate (SQ-F), (b) `friends-of-friends` composite predicate (SQ-FF), to understand how SQ length depends on the graph distance. The former serves as a proxy for *Friend* queries,

SQ Type	**top-**100	**top-**1000
Structrue Query All (SQ-A)	**1.64 ± 0.59**	2.00 ± 0.72
Structured Query Friends (SQ-F)	**1.01 ± 0.61**	1.70 ± 0.77
Structured Query Friends of Friends (SQ-FF)	**1.63 ± 0.76**	2.02 ± 0.89

Table 2.5: Average structured grammar query length measured as the number of functional predicates.

while the latter for *Non-friend* queries. Table 2.5 summarizes the results.

As we can see, shorter SQs are more popular because the average query length for top-100 is smaller than for top-1000. This is a predictable finding because shorter queries are likely easier to write. More interestingly, users write shorter queries when they search for the first degree connections. We have two ideas to explain this. First, the number of friends is much smaller compared to the number of non-friends, and hence, it takes less information to encode them (search entropy is lower). Second, according to [134], "*repeat web queries are shorter and more effective*". Analogously, users might be more effective in formulating queries about their friends because they know more about them.

To understand what predicates people use to disambiguate SQs while searching for people, we computed top-10 predicates most frequently co-appearing in SQs with at least one *User* predicate having a clear distance semantics, i.e the same SQ-F and SQ-FF sets of queries. Table 2.6 summarizes the results. An exemplary SQ for the second column is "*Photos of **my friends** who like CIKM2014*"; for the third column – "***Friends of my friends** who are not my friends*".

The users disambiguate queries using predicates in the following groups: location – `visitors`, `residents`, `home-residents` (e.g. "*Friends who visited Dublin*"), affiliation – `employees`, `students`, `employer-location`, `members` (e.g. "*Employees of Tesla Motors*"), interest – `likers` (e.g. "*People who like Hadoop and Pig*"), gender – `females` (e.g. "*Females who are single*"), and relation to other users – `non-friends(me)` (e.g. "*Friends of Bob who are not my friends*"). They also submit queries for people without providing any predicate, and some queries that have predicates are more popular than queries without them. However, we found that on average, shorter queries are more popular. To investigate why some longer queries are more frequent than their shorter counterparts, we introduced the concept of a lift predicate.

A **Lift Predicate** is a predicate that, when used as part of a query, increases the frequency of the query compared to the query without this predicate. For example, if we have the query *"Users named Alice who live in California"* with the frequency 100 and the query *"Users who live in California"* with the frequency 50, then `users-named` is a lift predicate because the former query is more frequent than the latter.

The lift predicates are insightful because unlike in traditional query frequency analysis, which tells us *what* queries are popular, they help us understand *why* some queries are popular. Top-10 lift predicates are presented in the rightmost column of Table 2.6. In addition to the groups of predicates discussed above,

Rank	in SQ-F	in SQ-FF	Lift predicates
1	*No predicates*	non-friends(me)	users-named
2	likers	residents	residents
3	residents	students	photos-of
4	employees	employees	friends(me)
5	members	likers	friends(X)
6	students	*No predicates*	non-friends(me)
7	females	users-named	females
8	visitors	friends(X)	photos-liked
9	home-residents	(employees, employer-location)	photos-by
10	(likers, likers)	home-residents	users-interested-in(males)

Table 2.6: Top-k predicates for different user queries.

users are interested in `photos`. We discuss how to leverage lift predicates in Section 2.7.

2.6.2 Functional Predicates and Graph Distance

In the previous section, we studied what predicates people use to disambiguate their SQs in order to find other people. Below we study what people want to learn about other people and whether it depends on the graph distance.

Similar to the distance calculation algorithm described in Section 2.2.2, we consider only queries with at least one entity. For each entity in a query we (a) identify the distance from the searcher using the social graph or functional superposition of the $User$ predicates, e.g. `friends(friends(me))` has a distance two; (b) find the closest non-$User$ predicate, for which it serves as an argument unwrapping the functional embedding. For example, a query `photos-of(friends(me))` is a *Friend* query and the closest non-$User$ predicate is `photos-of`; a query `videos-commented(123)` submitted by a friend of the user with the anonymous $id = 123$ is a *Friend* query and the closest non-$User$ predicate is `videos-commented`; a query `photos-in(places-liked(me))` is a *Self* query and the closest non-$User$ predicate is `places-liked`. The results for top-30 grammar predicates are presented in Figure 2.8.

Some predicates are “pure” because they can be applied only to the entities at a specific distance, e.g. `places-near`, while most of the predicates have all of the distance components. We observe a high variation from predicate to predicate and most of the predicates have a distribution over the graph distances deviating from the average distribution presented in Section 2.3. Users apply the `friends` predicate to non-friends, while interest-related predicates, e.g. `page-liked` and `videos-liked`, are biased towards friends. Users are also interested in knowing the locations of their first degree connections, e.g. `current-cities`. Job-related predicates, e.g. `employers`, are used more for non-friends, which aligns with Granovetter’s theory of weak ties [47]. Some predicates with the related semantics, like media, have drastically different

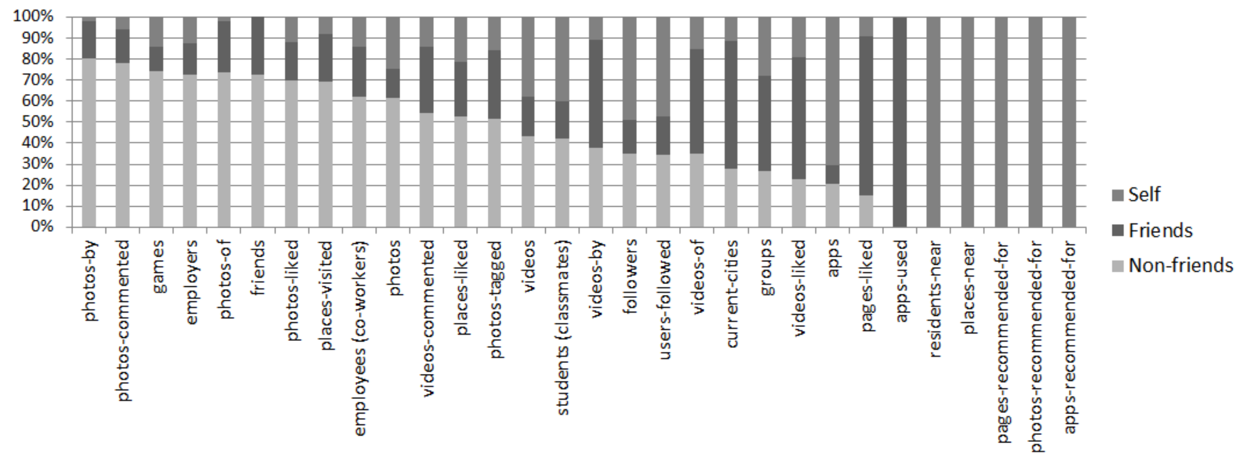


Figure 2.8: Distributions of graph distances for top-30 functional predicates. The distance is taken between a searcher and a user entity that serves as an argument for a functional predicate.

distributions: while `photos-of` is biased towards non-friends, `videos-of` is often used for friends.

2.7 Design Implications

In this section, we discuss what our findings suggest for the design of next-generation search products on top of structured and interconnected data sets and PSNs in particular.

2.7.1 Supporting Diverse Information Needs

Our key finding is that users search more for friends using NEQs and for non-friends using SQs (Section 2.5). Moreover, this behavior is consistent across numerous demographic slices (Section 2.5.1). Therefore, an interactive Typeahead interface supporting both NEQs and SQs facilitates navigation and exploration and makes information stored within the OSN useful and easy to search. Having these two query types tailored to a specific class of information needs within one system is beneficial for users of the OSN as these queries don't compete but rather complement each other. A similar idea was proposed based on the analysis of a LinkedIn people query log, where the authors found serious differences in post-search behavior for name and non-name queries.

2.7.2 Improving the Quality of (Structured) Query Suggestions

We noticed significant changes in search behavior for users with different demographics. We found that the number of *Friend* queries grows as users gain more friends, while the number of *Non-friend* queries slightly declines (Section 2.5.1), that celebrity users search for celebrities more than typical users (Section 2.5.1), that females and users, who are older than 60, are more interested in the first degree connections compared to the rest of the users in our sample (Section 2.5.1), and several others. Therefore, we suggest to further innovate around personalized search query suggestions given our demographics' distinct people search patterns. It is worth mentioning that SQ usage behavior has a wider variation across different demographics, and hence, it makes sense to focus the efforts on that query type.

For example, we recommend building personalized grammar query suggestions at a Semantic Query Template level because we found that users reuse Semantic Query Templates to search for different entities. Moreover, users repeatedly search for non-friends without adding them to their friend network. This implies that query suggestions shouldn't be limited to friends only, but should also include some interesting distant vertices in the social graph.

We believe that the quality of search suggestions would improve using lift predicates by concatenating them to the $User$ queries to facilitate entity disambiguation. Additionally, it would speed up the query writing process. Lift predicates are advantageous over frequent query suggestions because the former can boost the popularity of a new query while the latter are limited to the set of existing queries.

We found that the distribution over graph distances varies from predicate to predicate. While some predicates are used primarily to explore information about friends, other predicates are used for non-friends. We propose ranking entities for a predicate using its graph distance distribution.

We discovered that users write shorter queries when they search for *Friends* and use more predicates to find *Non-friends*. Therefore, we propose generating interactive query suggestions by predicting an intended search distance and deciding whether (a) to add one more predicate to the original query to generate a list of longer queries or (b) to stop growing the query and iterate over the predicates applicable for the entity in question. For example, once a user specified a name of a friend, it makes sense to show other predicates used for friends, e.g. `pages-liked` or `videos-liked` (Figure 8). At the same time, if the user typed a name of a non-friend, one can extend the query using disambiguation or lift predicates, e.g. `employees` or `residents`.

2.8 Limitations and Future Work

We do acknowledge several limitations of this study. First, we used proprietary data sets, which makes the reproducibility of the study possible only by a selected few. However, we argue that the methodology underlying the reported analysis is general to allow similar analyses by other researchers with access to similar data sets. Moreover, it is the first large scale study of a search system providing structured querying capabilities for a casual user, which is important to distribute in the information retrieval community.

Second, throughout this Chapter 2 we considered three categories of queries based on the graph distance – *Self*, *Friend*, and *Non-friend*. However, we could have done more fine-grained analysis and calculated exact graph distances between entities by using more compute power. Third, some findings might depend on the concrete implementation of the Typeahead query suggestion algorithm, e.g. users might search for Non-friends more using SQs simply because they are shown such suggestions. However, the same applies to almost any query log study of a modern search engine. Finally, we used a quantitative approach, which knowingly has its own shortcomings. While it allows uncovering numerous data-driven insights at scale, it cannot uncover users' motivations and goals. Therefore, we can only speculate about the reasons underlying observed user behavior. Qualitative survey or interviews are necessary to backup our quantitative findings.

2.9 Conclusions

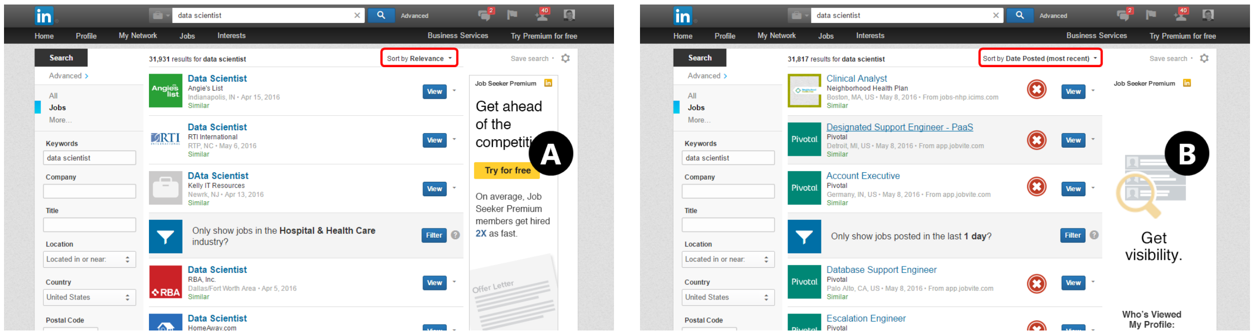
In this Chapter 2, we conducted the large scale analysis of Facebook Graph Search query logs. It is the first analysis that revealed insights on how demographic attributes and graph distance affect people search within an online social network. We presented the comprehensive overview of named entity and structured grammar queries usage and uncovered many new insights about people search within OSNs and PSNs. The key takeaways from Chapter 2 are two-fold. First, named entity and structured queries complement each other, and it is crucial to have them both in one system to address a diverse set of information needs of users. Second, search usage behavior changes a lot for users with different demographics, and it is important to model this variance to fulfill interests of everyone. We proposed numerous suggestions on how to develop better people search systems within OSNs and PSNs in particular, and, more broadly, how one can improve interactive structured search interfaces for networked data.

Chapter 3

Relevance-aware Results Filtering for Job and People Search

Sorting tuples by an attribute value is a typical search scenario within PSNs and many search engines within PSNs support such capabilities, e.g. time-based sorting for resumes or status updates, salary-based sorting for jobs, or distance-based sorting for events. However, sorting purely by an attribute value might lead to poor user experience because relevance is typically not taken into account. Hence, at the top of the list users might see irrelevant results (Figure 3). In this Chapter 3, we focus on the control aspect of the PSN search user interface with the aim to address this significant problem and answer our **RQ5(a)** and **RQ5(b)**. First, in answering **RQ5(a)** (*"Can the quality of results sorted by an attribute value be improved by incorporating relevance into the ranking process?"*), we developed a new filtering algorithm that directly optimizes a given search quality metric and uses predicted relevance scores in the process. Second, in answering **RQ5(b)** (*"What is the best way to accomplish it?"*), we provided an argument that the proposed algorithm is theoretically optimal as it is based on the dynamic programming framework and, hence, "virtually" enumerates all possible solutions. Finally, we performed a comprehensive evaluation study of this algorithm on synthetic and real learning to rank datasets. Based on our experimental results, we concluded that the proposed algorithm is superior to typically used heuristics and has clear practical value for search applications. By the end of this Chapter 3, we will better understand how to improve PSN search user experience when users perform structured entity sorting by an attribute value.

In the next Section 3.1, we describe an ad hoc search engine evaluation study that we conducted to motivate this project and understand how existing PSN search engines support sorting by an attribute value. Then, we cover related work in Section 3.2. We introduce our algorithm for relevance-aware tuple filtering, analyze its computational complexity, and explain why it is theoretically optimal in Section 3.3. After that, we turn to the experiments and describe the datasets used, the evaluation procedures, and the results in Section 3.4 and in Section 3.5. We end this Chapter 3 with the discussion of possible limitations that search engines ought to consider when deciding to use this approach in practice (Section 3.6).



data scientist

31,931 results for **data scientist** — Sort by Relevance

Keywords: data scientist
Company:
Title:
Location: Located in or near:
Country: United States
Postal Code

Data Scientist — Angie's List — Indianapolis, IN • Apr 15, 2016 — Similar

Data Scientist — RTI International — RTP, NC • May 6, 2016 — Similar

DAta Scientist — Kelly IT Resources — Newrk, NJ • Apr 13, 2016 — Similar

Only show jobs in the **Hospital & Health Care** industry?

Data Scientist — RBA, Inc. — Dallas/Fort Worth Area • Apr 5, 2016 — Similar

Data Scientist — HomeAway.com

Get ahead of the competiti

Try for free

On average, Job Seeker Premium members get hired 2X as fast.

Offer Letter

data scientist

31,817 results for **data scientist** — Sort by Date Posted (most recent)

Keywords: data scientist
Company:
Title:
Location: Located in or near:
Country: United States
Postal Code

Clinical Analyst — Neighborhood Health Plan — Boston, MA, US • May 8, 2016 • From jobs-nhp.icims.com — Similar

Designated Support Engineer - PaaS — Pivotal — Detroit, MI, US • May 8, 2016 • From app.jobvite.com — Similar

Account Executive — Pivotal — Germany, IN, US • May 8, 2016 • From app.jobvite.com — Similar

Only show jobs posted in the last **1 day**?

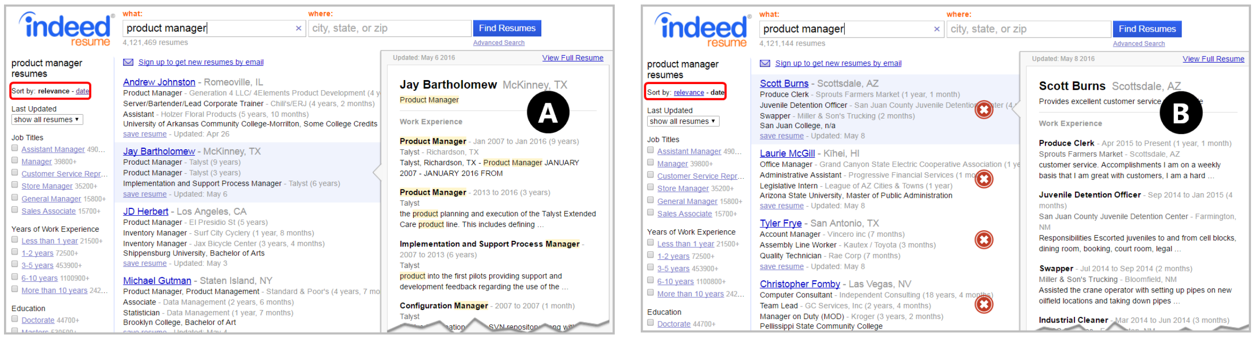
Database Support Engineer — Pivotal — Palo Alto, CA, US • May 8, 2016 • From app.jobvite.com — Similar

Escalation Engineer — Pivotal

Get visibility.

Who's Viewed My Profile:

Figure 3.1: *(A)* LinkedIn's job search results for the query *"data scientist"* sorted by ***"relevance"***.*(B)* LinkedIn's job search results for the query *"data scientist"* sorted by ***"date"*** Sorting by relevance yields relevant results. The results sorted by the attribute value are hardly relevant for the query.



product manager — 4,121,469 resumes

product manager resumes

Sort by: **relevance** - date

Last Updated: show all resumes

Job Titles: Assistant Manager 490... ; Manager 39800+ ; Customer Service Repr... ; Store Manager 35200+ ; General Manager 15800+ ; Sales Associate 15700+

Years of Work Experience: Less than 1 year 21500+ ; 1-2 years 72500+ ; 3-5 years 453900+ ; 6-10 years 1100900+ ; More than 10 years 242...

Education: Doctorate 44700+ ; Masters 530500+

Sign up to get new resumes by email

Andrew Johnston - Romeoville, IL
Product Manager - Generation 4 LLC/ 4Elements Product Development (4 y
Server/Bartender/Lead Corporate Trainer - Chili's/ERJ (4 years, 2 months)
Assistant - Holzer Floral Products (5 years, 10 months)
University of Arkansas Community College-Morrilton, Some College Credits
save resume - Updated: Apr 26

Jay Bartholomew - McKinney, TX
Product Manager - Talyst (9 years)
Product Manager - Talyst (3 years)
Implementation and Support Process Manager - Talyst (6 years)
save resume - Updated: May 6

JD Herbert - Los Angeles, CA
Product Manager - El Presidio St (5 years)
Inventory Manager - Surf City Cyclery (1 year, 8 months)
Inventory Manager - Jax Bicycle Center (3 years, 4 months)
Shippensburg University, Bachelor of Arts
save resume - Updated: May 3

Michael Gutman - Staten Island, NY
Product Manager, Product Management - Standard & Poor's (4 years, 7 mo
Associate - Data Management (2 years, 6 months)
Statistician - Data Management (1 year, 7 months)
Brooklyn College, Bachelor of Art
save resume - Updated: May 4

Updated: May 6 2016

Jay Bartholomew McKinney, TX
Product Manager

Work Experience

Product Manager - Jan 2007 to Jan 2016 (9 years)
Talyst - Richardson, TX
Talyst, Richardson, TX - Product Manager JANUARY 2007 - JANUARY 2016 FROM

Product Manager - 2013 to 2016 (3 years)
Talyst
the product planning and execution of the Talyst Extended Care product line. This includes defining ...

Implementation and Support Process Manager - 2007 to 2013 (6 years)
Talyst
product into the first pilots providing support and development feedback regarding the use of the ...

Configuration Manager - 2007 to 2007 (1 month)
Talyst

product manager — 4,121,144 resumes

product manager resumes

Sort by: relevance - **date**

Last Updated: show all resumes

Job Titles: Assistant Manager 490... ; Manager 39800+ ; Customer Service Repr... ; Store Manager 35200+ ; General Manager 15800+ ; Sales Associate 15700+

Years of Work Experience: Less than 1 year 21500+ ; 1-2 years 72500+ ; 3-5 years 453900+ ; 6-10 years 1100800+ ; More than 10 years 242...

Education: Doctorate 44700+

Sign up to get new resumes by email

Scott Burns - Scottsdale, AZ
Produce Clerk - Sprouts Farmers Market (1 year, 1 month)
Juvenile Detention Officer - San Juan County Juvenile Detention Center (4
Swapper - Miller & Son's Trucking (2 months)
San Juan College, n/a
save resume - Updated: May 8

Laurie McGill - Kīhei, HI
Office Manager - Grand Canyon State Electric Cooperative Association (1 ye
Administrative Assistant - Progressive Financial Services (1 mo
Legislative Intern - League of AZ Cities & Towns (1 year)
Arizona State University, Master of Public Administration
save resume - Updated: May 8

Tyler Frye - San Antonio, TX
Account Manager - Vincero inc (7 months)
Assembly Line Worker - Kautex / Toyota (3 months)
Quality Technician - Rae Corp (7 months)
save resume - Updated: May 8

Christopher Fomby - Las Vegas, NV
Computer Consultant - Independent Consulting (18 years, 4 months)
Team Lead - GC Services, Inc (2 years, 4 months)
Manager on Duty (MOD) - Kroger (3 years, 2 months)
Pellissippi State Community College

Updated: May 8 2016

Scott Burns Scottsdale, AZ
Provides excellent customer service

Work Experience

Produce Clerk - Apr 2015 to Present (1 year, 1 month)
Sprouts Farmers Market - Scottsdale, AZ
customer service. Accomplishments I am on a weekly basis that I am great with customers, I am a hard ...

Juvenile Detention Officer - Sep 2014 to Jan 2015 (4 months)
San Juan County Juvenile Detention Center - Farmington, NM
Responsibilities Escorted juveniles to and from cell blocks, dining room, booking, court room, legal ...

Swapper - Jul 2014 to Sep 2014 (2 months)
Miller & Son's Trucking - Bloomfield, NM
Assisted the crane operator with setting up pipes on new oilfield locations and taking down pipes ...

Industrial Cleaner - Mar 2014 to Jun 2014 (3 months)

Figure 3.2: *(A)* Indeed's Resume search results for the query *"product manager"* sorted by ***"relevance"***.*(B)* Indeed's Resume search results for the query *"product manager"* sorted by ***"date"*** Sorting by relevance yields relevant results. The results sorted by the attribute value are hardly relevant.

3.1 Motivation

To evaluate how attribute-based sorting is supported by PSN search engines today, we conducted the ad hoc evaluation of four popular job search engines[1]. For each search engine, we submitted 25 queries, applied the sorting based on one of the attributes (relevance or date), and judged the quality of results. The following queries were used: *"Account Analyst"*, *"Account Director"*, *"Account Manager"*, *"Analyst"*, *"Analytical Scientist"*, *"Business Analyst"*, *"Data Analyst"*, *"Database Administrator"*, *"Driver"*, *"IT Consultant"*, *"IT Manager"*, *"Java Developer"*, *"Junior Developer"*, *"Lead Software Engineer"*, *"Office Manager"*, *"Project Manager"*, *"Python Developer"*, *"Quantitative Analyst"*, *"Real Estate Manager"*, *"Registered Nurse"*, *"Research Assistant"*, *"Risk Manager"*, *"Sales Manager"*, *"Software Architect"*, *"Web Developer"*. The statistics from this ad hoc evaluation study is presented in Table 3.1.

We found that the ranking by relevance was of very high quality — the average Precision@10 was 0.86. On

[1]Indeed, LinkedIn, SuperJob, Monster for jobs.

Search Engine \ Metric	**Precision@1**	**Precision@5**	**Precision@10**	**"Weak" SERP**
Indeed	0.13 ± 0.34	0.28 ± 0.31	0.25 ± 0.24	78%
LinkedIn	0.26 ± 0.45	0.22 ± 0.27	0.26 ± 0.24	87%
SuperJob	0.52 ± 0.51	0.55 ± 0.39	0.58 ± 0.40	43%
Monster	0.83 ± 0.39	0.89 ± 0.25	0.91 ± 0.22	9%
Average	**0.46 ± 0.5**	**0.48 ± 0.40**	**0.50 ± 0.39**	**54%**

Table 3.1: The performance of different search engines when sorting by the attribute value is applied.

the other hand, we found that the search results were far from relevant when the attribute-based sorting was applied. For instance, across the sites the average Precision@1 was 0.46, Precision@5 was 0.48, Precision@10 was 0.50, and 54% of queries had the Precision@10 below 0.5 (denoted as "Weak" SERP in Table 3.1). By looking at individual SERPs (e.g. Figure 3), we realized that the main reason for failed searches was due to obviously irrelevant results jumping to the top because they had very high/low attribute values. It suggests that nowadays search engines don't take relevance into account when the attribute-based sorting is requested. Interestingly, we were able to increase the quality of results dramatically even using very simple heuristics (top-k threshold). On the pages to follow, we describe how we developed our optimal approach.

3.2 Related Work

This work is related to the research on search user behavior analysis, search metrics, and learning to rank. The proposed algorithm is based on the dynamic programming [12].

Researchers studied the way people interact with search engines by analyzing mouse movements, eye-tracking and click logs. Joachims et al. [66] discovered the *position bias* phenomenon, i.e. the results at the first two positions receive most attention, and then it quickly drops. Plus, on average users tend to read the results in a linear order from top to bottom. Craswell et al. [27] explored how the position bias might arise and proposed four hypotheses and the corresponding probabilistic click models. They found that the "cascade" model, where users view results from top to bottom and leave as soon as they see a worthwhile document, is the best explanation for position bias in early ranks. Dupret et al. [40] generalized this model by allowing for the possibility that a user skips a document without examining it.

Complementary to work on search models, a lot of attention has been devoted to the design and analysis of search metrics. Thus, in addition to the traditional metrics, like the Precision and the Recall, Järvelin and Kekäläinen proposed the (Normalized) Discounted Cumulative Gain (DCG) [64], Chapelle et al. — the Expected Reciprocal Rank (ERR) [22], to name just a few. Recently, Chuklin et al. [26] developed a unified framework to convert any click model into the evaluation metric. Essentially, all search metrics model the position bias and penalize the top ranked irrelevant results.

Numerous ranking algorithms have been developed to accurately predict the relevance of documents. Typically, these algorithms are based on machine learning and find the optimal parameters by optimizing the "surrogate" objective function. However, the solution to the approximation is not always optimal for the original ranking problem. Therefore, recently several approaches have been proposed that directly optimize a given search metric. For instance, Xu et al. [150] focus on the algorithms that optimize the objectives upper-bounding the original non-smooth search metrics. Tan et al. [125] proposed *DirectRank*, which is based on the iterative coordinate ascent with the smart line search procedure.

The attribute-based ranking, however, has been handled very differently. Rather than taking the relevance into account, search engines return the list of results sorted by an attribute value or suboptimal heuristics are used (Section 4.1). Inspired by the recent advancements in learning to rank, in this work we bridge the gap between the relevance-based ranking and the attribute-based ranking by proposing to do relevance-aware search results filtering which directly optimizes a given search metric when the sorting by an attribute value is requested. It is worth highlighting the difference between the proposed algorithm and the famous Threshold Algorithm (TA) by Fagin et al. [43]. While the TA algorithm finds top-k most relevant tuples by scoring them *individually*, we return the tuples which *cumulatively* optimize a given search quality metric.

3.3 Our Approach

We consider the scenario when a user requests the sorting of the search results by an attribute value, e.g. by salary or date (Figure 3). Our goal is to produce the final ranking that both satisfies strict ordering constraints and optimizes a given search quality metric (in turn, it minimizes the user's effort on finding relevant results). We only focus on results filtering and assume that relevance scores are already predicted by the ranking algorithm. Therefore, the formalization of our problem looks as follows.

Input: *a list of tuples* $\{(t_i, r_i)\}_{i=1}^{l}$*, where* t_i *is the attribute value and* $r_i \in \mathbb{R}+$ *is the relevance score predicted by the ranking algorithm; a search quality metric* Q.

Output: *a (sub)list of indices* J *delivering the maximum to the metric* Q *and totally ordered based on the attribute value, i.e.* $J = \arg\max Q(r_{j_i} | j_i \in J)$*, s.t.* $t_{j_1} < ... < t_{j_l}$.

Throughout this Chapter 3, we consider *DCG* as the search quality metric (although *ERR* or another metric could be used), date as the attribute, and the input sorted chronologically. However, these assump-

Algorithm 1 (A1) Relevance-aware filtering of totally ordered set via direct optimization of a search quality metric

```
Input: $DCG$ and $\{(t_i, r_i)\}_{i=1}^{l}$, s.t. $t_i < ... < t_l$ and $r_i \in \mathbb{R}+$
Output: $J = \arg\max DCG(r_{j_i} | j_i \in J)$, s.t. $t_{j_1} < ... < t_{j_l}$
1:  $M \leftarrow Matrix(l+1, l+1); M(:,0) \leftarrow 0; M(0,:) \leftarrow 0;$
2:  $Path \leftarrow Matrix(l+1, l+1);$ # to recover max sequence
3:  for $i$ in 1, ..., $l$
4:    for $j$ in 1, ..., $i$
5:      $gain \leftarrow \frac{2^{r_i}-1}{\log(j+1)};$
6:      if $M(i-1, j-1) + gain > M(i-1, j)$
7:        $M(i,j) \leftarrow M(i-1, j-1) + gain;$
8:        $Path(i,j) \leftarrow (i-1, j-1);$
9:      else
10:       $M(i,j) \leftarrow M(i-1, j);$
11:       $Path(i,j) \leftarrow (i-1, j);$
12: $(i,j) \leftarrow \arg\max M(l,:);$ # last element of solution
13: $J \leftarrow List(); J.append(j);$
14: while $i > 1$ and $j > 1$
15:   if $P(i,j).last < j$
16:     $J.append(P(i,j).last);$
17:   $(i,j) \leftarrow P(i,j);$ # jump to shorter subsequence
18: return $J.reverse()$
```

tions are made solely to simplify the presentation. It is worth mentioning that the formalization above covers the post-filtering scenarios as well, i.e. the input might consist of tuples that passed some other filtering algorithm.

Currently, this problem is solved heuristically. Mainly, there are two approaches built around the same idea of thresholding. We can take only the results that have a relevance score above the threshold. We can also sort the results by the relevance scores, take the top-k elements, and, finally, re-sort the list by date. While these approaches are easy to implement, they have two major drawbacks. First, it is not clear how to set the threshold. Second, the described approaches provide approximate solutions to our problem. Even the result set constructed from the top-k tuples sorted by relevance, being re-sorted by the attribute value, gets ordered randomly if we look at the relevance component.

The solution that guarantees optimality is to enumerate all possible subsequences, compute the metric for each one, and take the best. However, this approach is not tractable as the number of subsequences is exponential. We propose a polynomial algorithm based on the dynamic programming [12]. There are three key observations behind our algorithm: (1) natural enumeration order for subsequences; (2) additivity of the metric; (3) optimality of subproblems.

First, all subsequences can be partitioned into the factor classes based on their length, i.e. in each factor class there will be the subsequences of the same length. To enumerate all subsequences, we can iterate over the factor classes and within each factor class enumerate all subsequences. Second, search metrics are

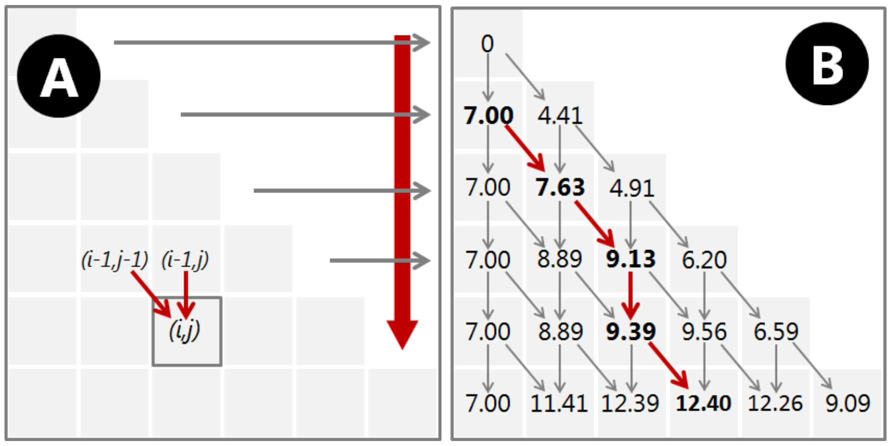


Figure 3.3: Dependencies in the memoization matrix, a legal evaluation order, and the optimal path.

additive and can be computed in linear time from the beginning of the list to the end [22]. It means that having a partial metric value for the prefix, we could compute the next metric value by simply adding the gain/utility provided by the current element. Third, the optimal subsequence for the prefix of length k is one of the optimal subsequences from each of the factor classes for the prefixes of length up to $k-1$ with or without the current element appended (proof by induction for the prefix length).

Combining the observations above, we present our algorithm and its analysis. It starts by initializing the memoization matrix to store the optimal *DCG* values for subproblems and the transition matrix to reconstruct the optimal subsequence. Then, it iterates over the prefixes of the input sequence in the outer loop and over the factor classes in the inner loop. The cell (i,j) is for the optimal subsequence of length j for the prefix of length i. At each step, we decide whether we should append the current element of the input sequence i to the optimal subsequence of length $j-1$ for the prefix of length $i-1$ (the recursion on lines 6-11). If we append the current element, we go diagonally. If we don't append, we keep the existing optimal subsequence of length j and stay in the same column. A legal evaluation order and dependencies between the cells are shown in Figure 3.3, A. Finally, to reconstruct the optimal subsequence, we find the maximum in the last row (the last element is always "in" since the elements are non-negative) and go backward in the *Path* matrix. If the line is diagonal, we take the element in the next cell. Otherwise, we skip. The *Path* matrix is depicted in Figure 3.3, B. The complexity (both time and space) of the algorithm is $O(l^2)$ because we have two nested loops, costing us $O(1)$ time at each iteration, and the square memoization matrices. It is guaranteed to deliver the optimum because we "virtually" enumerate all subsequences within the dynamic programming framework. For a toy example problem $\{(0,0),(1,3),(2,1),(3,2),(4,1),(5,3))\}$ the optimal solution is $\{1,3,4,5\}$ with the *DCG* equal to 12.40.

3.4 Experiments on Real Learning to Rank Datasets

In this section, we study how our approach contributes to the ranking quality using two real learning to rank datesets (MQ2007LETOR [102] and MSLR-WEB10K). We additionally investigate the effectiveness, efficiency, and selectivity of the proposed tuple filtering algorithm on four synthetically generated datasets.

3.4.1 Datasets

While there are more than ten publicly available learning to rank datasets that fit our evaluation needs, we decided to pick MQ2007 and MSLR-WEB10K as they are sufficiently diverse (the number of results per query, the dataset size, the number of features, the range of possible values for relevance judgments) and, hence, allow to evaluate the proposed algorithm from multiple points of view. Both MQ2007 and MSLR-WEB10K datasets can be obtained here[2] and here[3], respectively.

MQ2007

The MQ2007 learning to rank dataset is a sample of **1,600 queries** obtained from a retired training set of a commercial web search engine (Microsoft Bing) and annotated by professional annotators. It consists of feature vectors extracted from query-URL pairs along with relevance judgments and has a standard for machine learning applications sparse matrix format: each row corresponds to a query-URL pair, the first column is relevance label of the pair, the second column is query id, and the following columns are features.

The **relevance judgments** are integer numbers ranging from **0 (irrelevant)** to **2 (perfectly relevant)**. The features include the scores predicted by various information retrieval models, e.g. TF-IDF, BM25 [103], language models [99, 158, 159] for different web document fields (title, anchor, body, and etc.); quality indicators; click-based features [25]; PageRank [18], and other web graph measures. There are **46 features** in total. The dataset is pre-partitioned into five folds of equal size for consistent cross validation. Each fold contains approximately **335 queries**. On average there are **40 documents per query**.

MSLR-WEB10K

Likewise, the MSLR-WEB10K dataset is a sample of **10,000 queries** obtained from a retired training set of a commercial web search engine (Microsoft Bing) and annotated by professional annotators. It consists of feature vectors extracted from query-URL pairs along with relevance judgments and has a standard for

[2]http://research.microsoft.com/en-us/projects/mslr/
[3]http://research.microsoft.com/en-us/um/beijing/projects/letor/letor4dataset.aspx

machine learning applications sparse matrix format: each row corresponds to a query-URL pair, the first column is relevance label of the pair, the second column is query id, and the following columns are features.

The **relevance judgments** are integer numbers ranging from **0 (irrelevant)** to **4 (perfectly relevant)**. The features include the scores predicted by various information retrieval models, e.g. TF-IDF, BM25 [103], language models [99, 158, 159] for different web document fields (title, anchor, body, and etc.); quality indicators; click-based features [25]; PageRank [18], and other web graph measures. There are **136 features** in total[4]. The dataset is pre-partitioned into five folds of equal size for consistent cross validation. Each fold contains **2,000 queries**. On average there are **120 documents** per query.

3.4.2 Method

To answer our research questions, we do the simulations as follows. We extend MQ2007 and MSLR-WEB10K datasets by assigning a random timestamp to each document to model the sorting by the attribute value. Scikit-learn[5] implementation of the Gradient Boosted Regression Trees (*GBRT*) [45] is used to predict the relevance scores. The optimal parameters for the final *GBRT* model are picked using cross validation for each dataset. We use the 5-fold cross validation partitioning from LETOR [102].

We consider three popular baselines, which are typically used to perform the filtering of the search results when an attribute-based sorting is applied:

- **Baseline 1 (B1)**: sort by the attribute value and don't do any filtering;
- **Baseline 2 (B2) based on score thresholding**: sort by the attribute value and keep only the results with the predicted relevance scores above the threshold (we normalized the scores to [0,1] and set the threshold=0.5);
- **Baseline 3 (B3) based on rank thresholding**: sort results by the predicted relevance score, take the top-k (where k is the cut-off point for the metric value calculation), and re-sort by the attribute value the results that are left after the filtering.

The evaluation procedure works as follows. First, we train the *GBRT* on the training folds. Second, we predict the relevance scores using the trained *GBRT* model for the documents in the testing fold. Third, we apply a baseline filtering algorithm to the documents in the testing fold by working with the relevance scores from the step two and the randomly generated timestamps. Fourth, we apply our filtering algorithm to the tuples that passed the baseline filtering. Finally, knowing the true relevance labels, we calculate the

[4]http://research.microsoft.com/en-us/projects/mslr/feature.aspx
[5]`http://scikit-learn.org/stable/index.html`

NDCG	***@1***	***@5***	***@10***	***@20***	***@40***
B1 only	0.226	0.245	0.273	0.336	0.496
A1 ∘ *B1*	**0.299**	**0.287**	**0.304**	**0.363**	**0.511**
B2 only	0.289	0.318	0.357	0.448	0.450
A1 ∘ *B2*	**0.315**	**0.326**	**0.364**	**0.453**	**0.454**
B3 only	0.433	0.417	0.418	0.451	0.498
A1 ∘ *B3*	0.433	0.417	**0.420**	**0.455**	**0.512**

NDCG	***@1***	***@5***	***@10***	***@20***	***@40***
B1 only	0.131	0.161	0.190	0.236	0.309
A1 ∘ *B1*	**0.173**	**0.183**	**0.208**	**0.250**	**0.321**
B2 only	0.170	0.208	0.244	0.300	0.379
A1 ∘ *B2*	**0.192**	**0.215**	**0.250**	**0.304**	**0.383**
B3 only	0.390	0.362	0.365	0.380	0.418
A1 ∘ *B3*	0.390	0.362	**0.366**	**0.382**	**0.421**

Table 3.2: The demonstration of effectiveness of the proposed approach for relevance-aware tuple filtering on MQ2007 (left) and MSLR-WEB10K (right) datasets.

$NDCG@k$ for the filtered result list sorted by the timestamp. To make sure that the conclusions are not due to randomness, we average the results from $1,000$ runs.

3.4.3 Results

The results of the experiment are presented in Table 3.2. We can see that the output (post-)filtered with our algorithm is consistently better than the baselines. We applied the binomial test and found that almost all differences in the $NDCG$ values are statistically significant (marked in bold), p-value is below 0.001. One average the increase in the metric value is around 2-4%. Moreover, since the datasets used have very different characteristics (e.g. the average number of documents per query length for MQ2007 is around 40 and for MSLR-WEB10K it is around 120), the experiment suggests that the algorithm achieves good performance for a broad range of ranking problems.

3.5 Experiments on Synthetic Datasets

In this section, we focus on the filtering only (both relevance labels and timestamps are generated) and study how the algorithm behavior changes for different input sizes and relevance label distributions.

3.5.1 Method

We consider the following four label distributions modeling the real situations: (a) uniform integer in the range $[0, 5]$; (b) uniform real in the range $[0, 5]$; (c) power law, the slope $\alpha = 2.0$; (d) $\frac{3x^2}{125}$ with the support in the range $[0, 5]$. We generate the input lists for the filtering algorithm by sampling from the corresponding distribution. Similarly to the previous experiment, we simulate each combination of conditions 1,000 times and average the runs. Only the Baseline 1 is used in this experiment for simplicity.

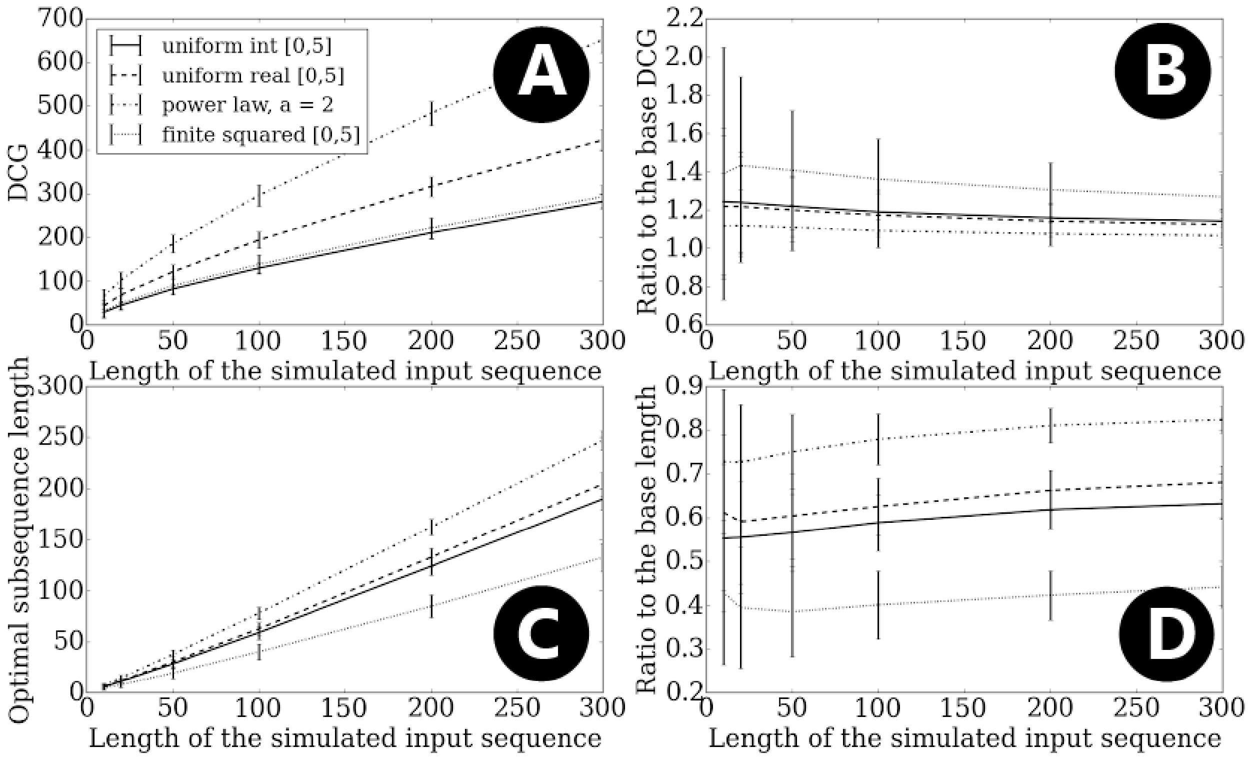


Figure 3.4: The behavior of the algorithm (A1) for different input sizes and relevance label distributions.

3.5.2 Results

The data from the simulation is presented in Figure 3.4. There are several observations that could be made with the help of this figure. First, the output size is linearly proportional to the input size (Figure 3.4, C). *DCG* also grows linearly with the growth of the input size (Figure 3.4, A). Second, the proposed algorithm always outperforms the Baseline 1 (Figure 3.4, B), which is expected because we do the filtering directly optimizing a given search quality metric. Third, both the graph for the ratio of the *DCG* values and the graph for the ratio of the output sequence length for the proposed algorithm to the output sequence length for the baseline monotonically converge for longer input lists (Figure 3.4, B and D). This means that our algorithm works better when the original hit list is shorter. At the same time, the filtering is not effective when the hit list is too short — the probability for a non-relevant result from the bottom of the list sorted by relevance to jump to the top of the list sorted by the attribute value is low. Therefore, we conclude that our algorithm is the most effective for the result sets made of 20-100 elements. Fourth, higher gains in *DCG* over the baseline are characteristic for the relevance label distributions where relevant results are more probable (Figure 3.4, B). This finding has a sound theoretical rationale — combinatorially counting the permutations of results, one can show that for our approach the DCG as a function of relevance peaks

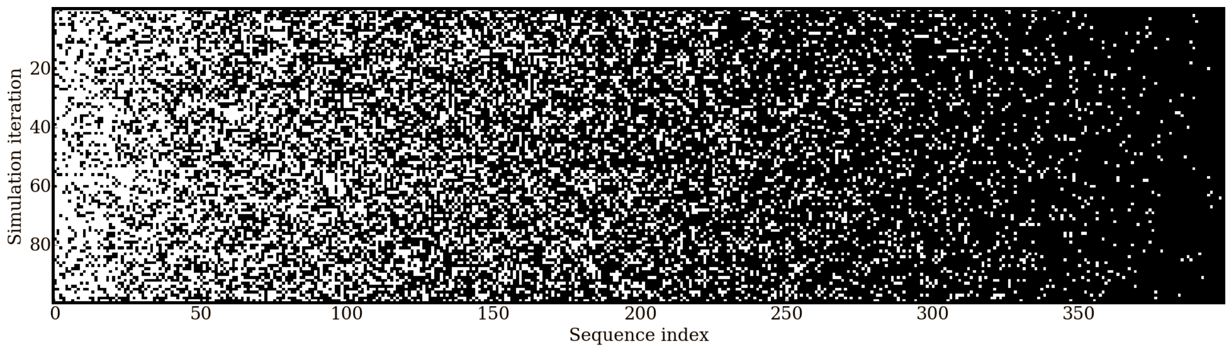


Figure 3.5: The indices of the elements included (black cells) by our approach in the optimal subsequence.

when relevance labels are skewed towards higher relevance. It is important to note that the observations above are valid for non-degenerate cases, e.g. not all labels are the same or sorted in a special order.

To better understand why the proposed approach works, we visualized the positions of elements that passed the filtering. Specifically, we first generated relevance labels by sampling 400 uniform real numbers[6] in the range $[0, 5]$ (the synthetic dataset (b) described above), then, we generated the random timestamps for each of the numbers, and, finally, applied our filtering algorithm on the sequence of numbers sorted by timestamps. This procedure was repeated 100 times and the indicator vectors were vertically concatenated into a 100 x 400 indicator matrix. The matrix is presented in Figure 3.5. As we can see, the algorithm is very selective at the beginning of the search result sets but it stops filtering the results as it moves deeper into the search results sets. We think that the main reason behind this behavior is the discounting factor in (N)DCG. The discounted gains for the elements at the bottom of the search result set become very small and the condition $M(i-1, j-1) + gain > M(i-1, j)$ from line six of our algorithm 3.3 rarely holds true.

3.6 Limitations and Future Work

In the previous sections, we demonstrated that our approach achieves superior performance by evaluating it on real and synthetic datasets and providing the argument that this algorithm is theoretically optimal. In this section, we describe the limitations of the algorithm and ideas for future work.

First, one should note that the increase in the ranking quality comes with the extra computational cost because the complexity of our algorithm is $O(\frac{l}{\log l})$ times larger than for the baselines. Therefore, to satisfy strict search user experience service level agreements (e.g. the results must be served under 300 milliseconds [108]), we recommend working with the result sets made of up to 1,000 elements (according to our latency benchmarks in Figure 3.6, we can process 1,000 results under 100 milliseconds using our C++

[6] 400 is not special and was picked solely for the purposes of illustration.

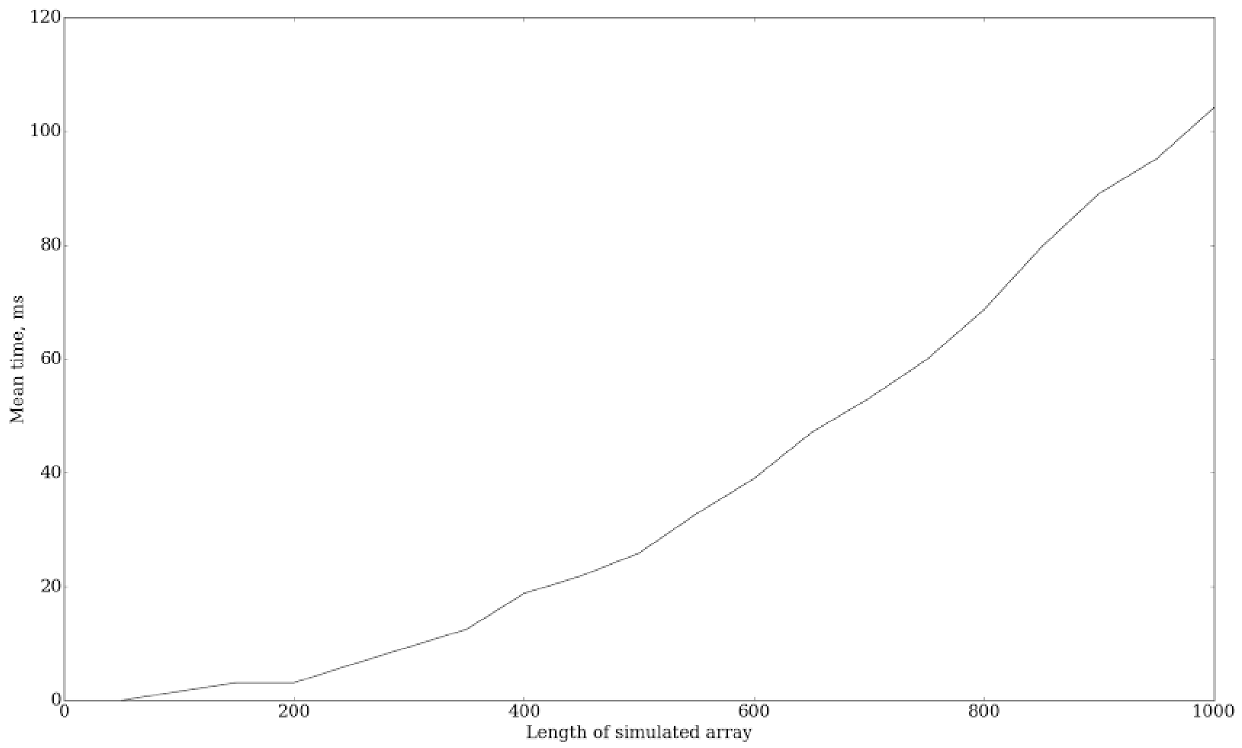


Figure 3.6: Scalability of the proposed approach.

implementation on a workstation with 4GB RAM and two 2.5GHz CPU cores).

Second, the algorithm is based on the idea of filtering and, hence, it removes some results from the original search result set. Therefore, the users might get confused, especially, if they first view the results sorted by relevance and then resort them by some other attribute. To avoid this confusion, we recommend keeping in the output the results already seen by the user within the session. That said, based on our experience and the evaluation described at the beginning of this Chapter 3, nowadays the results shown at the top of the SERP resorted by an attribute value are hardly relevant, which suggests that the users won't be upset not seeing them. An interactive user study is necessary to definitively answer this question.

Third, we evaluated the algorithm offline on the real and synthetic datasets. However, we didn't test this algorithm with real users. Therefore, it will be interesting to conduct an online A/B experiment to learn how users behave when some results are filtered out of the search output (re-)sorted by an attribute value. The control version will use the existing algorithm, which merely resorts the results by an attribute value. The treatment version will use the proposed algorithm. We will track direct and indirect metrics of search quality such as the number of job views per query, the number of job applications conditioned on SERP click, the distribution of clicks. The hypothesis will be that with the proposed algorithm users are more effective at search since they don't see irrelevant results.

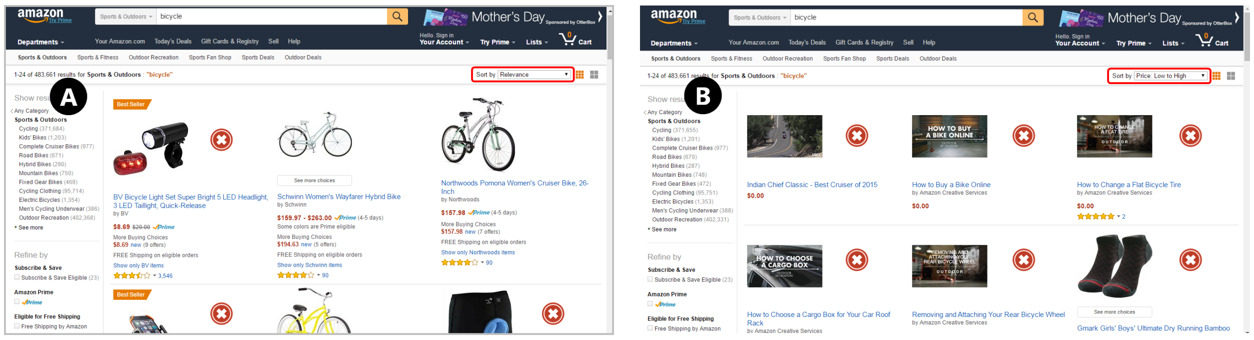


Figure 3.7: *(A)* Amazon's product search results for the query *"bicycle"* sorted by ***"relevance"***. *(B)* Amazon's product search results for the query *"bicycle"* sorted by ***"price (descending)"***. Sorting by relevance yields at least three relevant results. The results sorted by the attribute value are hardly relevant.

We also consider a different solution to the problem that irrelevant results jump to the top of the search results sorted by an attribute value. Rather than doing the filtering, we can still keep all results but additionally show a data visualization that signifies relevance for each result and allows to navigate quickly to a relevant result by clicking on its thumbnail. This idea is not novel as in the past there were several attempts to use data visualizations in search [3, 137, 144]. However, most of them were unsuccessful (see conclusions section at `http://searchuserinterfaces.com/book/sui_ch10_visualization.html`[7]. Two main barriers were: (1) that users didn't understand the concept of topical relevance, which is at the core of information retrieval and full-text search; (2) that *"the vertical position on the page served as a strong and effective relative signal of the result relevance"* [54]. Luckily, none of this is true in the case of structured search — the units of retrieval are structured entities which have typed attributes with clear semantics (e.g. price, date, size, amount, number of publications, age, etc.) and are sorted by an attribute value rather than by relevance. Therefore, an interactive user study of a new experimental system showing the data visualization briefly described above will be very fascinating. Plus, it might be useful to investigate the differences between precision-oriented search and recall-oriented search in the context of attribute-based sorting since the ranking order of search results doesn't serve as a proxy to relevance in this case any more.

3.7 Conclusions

In this Chapter 3, we addressed the significant problem in search, that is, how can we increase the relevance of the search results sorted by an attribute value. Our solution is based on the idea to perform relevance-aware search results filtering by directly optimizing a given search quality metric. We developed a simple,

[7]we are aware of seminal visual search user interfaces based on direct manipulation [2, 110] but consider a more traditional for information retrieval scenario, where the SERP with ten blue is used for the presentation of results.

yet effective algorithm based on the dynamic programming, which consistently outperforms typically used heuristic approaches and is guaranteed to deliver the optimal solution.

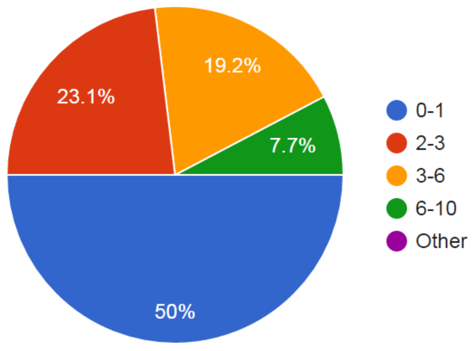
While this algorithm is extremely relevant for PSNs that contain a lot of structured data, such as user profiles, job postings, articles, and many others, it has a much wider area of application. Any structured search engine that deals with the typed data having numeric attributes might benefit from using this algorithm as it is guaranteed to increase the relevance of results when users perform sorting by an attribute value. For example, we foresee significant utility of this approach in e-commerce search when users do sorting by price, rating, delivery date, popularity, release date, weight, size (e.g. Figure 3.6), academic search when researchers do sorting by the number of publications, the number of citations, year, local search when users do sorting by distance or rating, and even web search as more and more data get transformed into the structured networked representation (e.g. Google Knowledge Graph [20, 93]).

Chapter 4

Extended Informative Structured Snippets for Job Search

In this Chapter 4, we describe our efforts towards the design of more informative snippets for job search and answer our **RQ6-RQ8**. First, in answering **RQ6(a)** (*"What information is necessary for users to decide whether they want to apply for a job?"*) and **RQ6(b)** (*"What information is necessary for users to decide which results to click on a SERP?"*), we conducted a mixed-method user study and found that in addition to the attributes commonly shown on the SERP by job search engines today (job title, company, location, date posted, and salary), users also consider responsibilities and requirements as important. Very often the information presented in the responsibilities and requirements sections of a job posting is used by users to decide whether they want to continue exploring the job posting. Therefore, we introduced the concept of *extended informative structured snippet* for job search that shows this information right on the SERP to save users' efforts. However, to be able to display this information, it first needs to be extracted. Second, in answering **RQ7(a)** (*"How can we automatically extract job responsibilities and requirements from an unstructured job posting with minimal supervision?"*), we developed a novel weakly-supervised approach for information extraction from job postings, that turns any unstructured job posting into the structured representation and in particular can extract the responsibilities and requirements sections. The approach leverages big data redundancy to create a training set with minimal supervision and achieves very high extraction quality. In answering **RQ7(b)** (*"How can we generate an extended informative structured snippet for a job posting having a list of job responsibilities and requirements in a structured form?"*), we adapted the Maximum Marginal Relevance Principle to our problem. Third, in answering **RQ8(a)** (*"Do extended informative structured snippets improve search user experience for job search?"*) and **RQ8(b)** (*"How do users behave when such structured snippets are used?"*), we conducted an online A/B test with the real users of HH.ru to evaluate the utility of extended informative structured snippets. In this Chapter 4, we will learn why and what information is crucial for users conducting job search online, how to extract this information from job postings, and how the presentation of this information on the SERP improves search UX.

In the next sections, we first cover the user study for needs elicitation by describing the research method in Section 4.1.1 and the results in Section 4.1.2. Then, we present a weakly-supervised approach for information



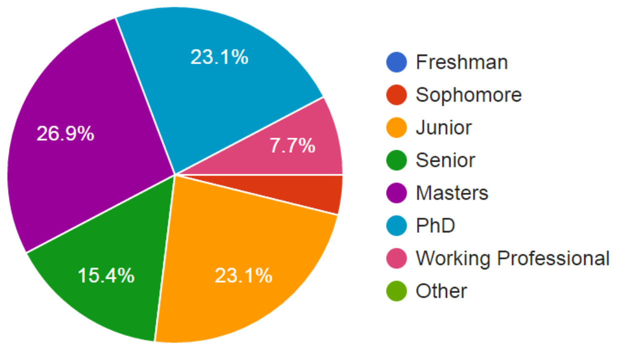


Figure 4.1: (Left) How many years of work experience do you have? (Right) What is your current student status (put “working professional” if you aren’t a student)?

extraction from unstructured job postings in Section 4.2 and describe the experiment that we conducted to evaluate the extraction quality in Section 4.2.1. We share our experience adapting the proposed approach for new languages in Section 4.3.1 and describe the online A/B experiment in Section 4.3.3.

4.1 User Study for Requirements Elicitation

We first conducted the user study to understand what information is necessary for users to decide whether they want to apply for a job (**RQ6(a)**) and what information users want to see on the SERP to make more informed job click decisions (**RQ6(b)**).

4.1.1 Method

Participants

Participants were recruited via personal contacts, print ads at UIUC campus, mailing lists, and word-of-mouth. All participants were required to have at least three months of work experience, be above 18 years old, and have experience using an online job search engine, e.g. Indeed.com, Monster.com, LinkedIn.com, Glassdoor.com. 26 people replied and were invited for the study.

Half of the participants were female (13 people). Most of the participants were UIUC students (24 people) and two were working professionals (a Marketing Manager and a Personal Trainer). The participants had diverse work experience (0-8 years, the average is 2.3 years) and education (2-9 years in the university, the average is 4.8 years). See Figure 4.1 for details. The participants represented many professional fields (sorted by popularity): Software Engineer (8 people), Data Scientist (3 people), Healthcare Consultant (2 people), Research Scientist (2 people), Personal Trainer (1 person), Genetics Counsellor (1 person), Product Manager (1 person), Translator (1 person), Occupational Therapist (1 person), Marketing Manager (1 person), Business Analyst (1 person), Foreign Policy Representative (1 person), Consultant (1 person), Biomedical Product Developer (1 person), Pharmacist (1 person). By interviewing a diverse set of professionals, we were able to capture the multiplicity of job search strategies and needs.

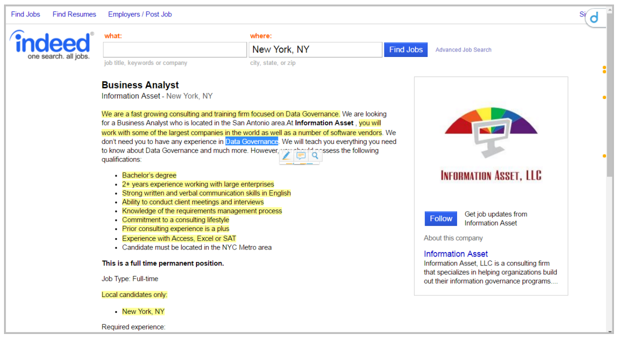
Procedure

We started the study session from the orientation (five minutes) where we explained to a participant the purpose of the study and the experimental procedure. The main part of the study (35 minutes) consisted of three stages. First, the participant was required to search for a job relevant to her/him using Indeed.com (ten minutes). The goal was to find and open five relevant job postings. We asked the participant to think aloud as s/he conducted her/his search to understand the rationale behind the search actions and behavior on the SERP. Second, once the participant discovered five relevant job postings, we asked her/him to look at each job posting and highlight information that s/he used to decide whether s/he wants to apply for this job. A web annotation tool was provided to assist the participant with this task[1] (Figure 4.2). Again, the participant had to explain why s/he highlighted some information to help us understand the rationale behind the thinking process. We allocated around three minutes per job posting. Third, we asked the participant to complete two surveys (five minutes per survey). In the surveys, we asked the participant to score each of 28 job attributes on a scale from 1-5 (5-point Likert scale [100]). We generated a set of 28 job attributes by analyzing search user interfaces of popular job search engines (LinkedIn, Monster, Glassdoor, Indeed, HeadHunter, SuperJobs) and a sample of job postings. One survey was directed towards the apply decision, i.e. we asked the participant to *"Rate the usefulness of each piece of information about a job to help you decide where to apply?"*, and the other was directed towards the search process and SERP click decisions, i.e. *"Imagine you are looking for a new job using an online search engine (e.g. Indeed, LinkedIn, Glassdoor, Monster,...). What information would you like to see on the search results page?"*. We intentionally put the surveys at the end of the user study session to make sure that the participant had the right context after s/he had engaged in job search/highlighting first. We rewarded the participant with the $5 gratuity.

Measures and Analysis

From each participant, we collected the background survey (also used as an application for the study), "think-aloud" comments about the job search process, annotated job postings, and survey responses. It allowed us to answer **RQ6(a)** and **RQ6(b)** from multiple perspectives and triangulate the findings.

[1]https://www.diigo.com/



Business Analyst
Information Asset - New York, NY

We are a fast growing consulting and training firm focused on Data Governance. We are looking for a Business Analyst who is located in the San Antonio area.At **Information Asset** , you will work with some of the largest companies in the world as well as a number of software vendors. We don't need you to have any experience in Data Governance. We will teach you everything you need to know about Data Governance and much more. However, [illegible]ssess the following qualifications:

- Bachelor's degree
- 2+ years experience working with large enterprises
- Strong written and verbal communication skills in English
- Ability to conduct client meetings and interviews
- Knowledge of the requirements management process
- Commitment to a consulting lifestyle
- Prior consulting experience is a plus
- Experience with Access, Excel or SAT
- Candidate must be located in the NYC Metro area

This is a full time permanent position.

Job Type: Full-time

Local candidates only:

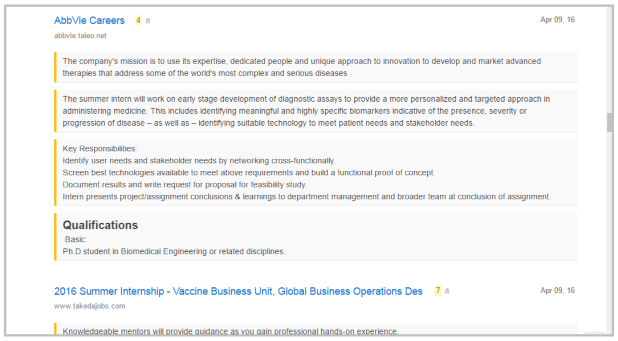
- New York, NY

Required experience:

Get job updates from Information Asset

About this company

Information Asset
Information Asset, LLC is a consulting firm that specializes in helping organizations build out their information governance programs....



AbbVie Careers 4 — Apr 09, 16
abbvie.taleo.net

The company's mission is to use its expertise, dedicated people and unique approach to innovation to develop and market advanced therapies that address some of the world's most complex and serious diseases

The summer intern will work on early stage development of diagnostic assays to provide a more personalized and targeted approach in administering medicine. This includes identifying meaningful and highly specific biomarkers indicative of the presence, severity or progression of disease – as well as – identifying suitable technology to meet patient needs and stakeholder needs.

Key Responsibilities:
Identify user needs and stakeholder needs by networking cross-functionally.
Screen best technologies available to meet above requirements and build a functional proof of concept.
Document results and write request for proposal for feasibility study.
Intern presents project/assignment conclusions & learnings to department management and broader team at conclusion of assignment.

Qualifications
Basic:
Ph.D student in Biomedical Engineering or related disciplines.

2016 Summer Internship - Vaccine Business Unit, Global Business Operations Des 7 — Apr 09, 16
www.takedajobs.com

Knowledgeable mentors will provide guidance as you gain professional hands-on experience.

Figure 4.2: *(Left)* Diigo.com web annotation user interface. It allows to highlight text on a page and save it. *(Right)* Diigo.com personal cabinet that allows to view all annotations in one place.

4.1.2 Results

Findings from the think-aloud comments

We coded the comments collected from the participants and counted the number of times they referenced different attributes. **Company** (36), **skills** (34), and **job title** (29) were the top three most frequently mentioned attributes[2]. The participants looked for the companies they know, the prestigious companies with the strong brand names in the respective industries, and the companies they had some relationship in the past (e.g. worked there in the past as an intern). For example, one participant remarked that *"Intel is a big name, good as the first job since it will help for future job search "*.

Skills were the second most important attribute helping the participants to decide whether they want to apply for a job. Most importantly, the skills (as well as other attributes related to the requirements and qualifications, which we will discuss below) were considered as a necessary condition. When the skills didn't match the participant background, the participant didn't click on it (while on the SERP) or stopped looking at it (when on the detailed job description page). On the other hand, when the skills matched the participant background, the participant felt very positive and was more willing to learn more about the job. To illustrate these general observations, we provide several anecdotal comments shared by the participants:

> *"I stopped once I saw SQL and other coding technologies. I am a different kind of analyst."* **[P2]**

> *"I look for the skills and try to evaluate how strong I am so that I don't get immediately rejected. Basically, I try to count the number of required skills I covered. If a lot of the skills don't match my background, I go for another job."* **[P14]**

[2]we report the absolute counts at the attribute level and not at the participant level. While it is less accurate, it is sufficient to understand the overall attribute preference.

"When I search I try to follow the following strategy: if I am sure that I fit, I will open the job posting [form the SERP], if it is 50/50, I will still open it since I am exploring more options, if am sure that I don't qualify, I will skip. Basically, I look for must have criteria and if they aren't satisfied, I skip. For me, these are title, skill, major, and degree." **[P22]**

The participants paid attention to the job title because it is one of the most salient points of the SERP and the job description. They liked the jobs that had job titles exactly matching the ones they were interested in, plus the ones that had more specific details in it. For example, the participants liked the job titles that specified the area of responsibilities and required skills (e.g. *"Software Engineer, Back-end Node.js"*), job type and dates of employment (e.g. *"Social Media Marketing Intern — Summer 2016"*), location (e.g. *"Healthcare Consultant, Greater New York Area"*). We learned that background knowledge about the industry was crucial to make more informed click decisions as one participant remarked that she knows *"...that legal translators are paid well, so I will check this job..."*.

Among other attributes, the participants noted (presented in the order of popularity): **responsibilities** (25), **years of work experience** (22), **degree** (15), **location** (14), about us information describing culture and mission of the organization (9), visa sponsorship and work authorization (8, especially relevant for international students), salary (7), job type (7), crowdsourced average company rating (7) and the number of votes (6), posting date (4) and deadline to apply (2), perks (1), and the amount of travel required (1).

The participants also noted that: (a) on the SERP they paid more attention to the information in the title since *"it is in bold, it stands out and I always look at it first "*; (b) when they *"...see something second time from the same company..."*, they *"would skip it since the information is redundant"*; (c) they preferred job postings with the structured layout and more visual content (images and may be videos with the employees); (d) they wanted to be able to specify the attributes the search engine shows to them.

Findings from the job annotations comments

We counted the number of times each specific attribute was highlighted by the participants. The counts were aggregated at the job posting level to account for the situations when the disproportionate amount of space is dedicated to one aspect of a job. The ranked list of attributes turned out to be as follows. The most frequent group of attributes was about **qualifications** (83 annotations in total). It, in turn, included **required skills** (72), **degree** (59), **years of work experience** (53), and very rarely GPA (2) and age (1). The next most frequent group was **responsibilities** (71). After that, we observed a drastic drop — location (24), job type (24), job title (16), work authorization / eligibility (13). People also considered useful the information about companies (14) and specifically about the company culture and mission (11), salary (11), how to apply (7), industry (6), application deadline (4) and posting date (3), the amount of travel required (3), perks (3). Five people highlighted company names and one person highlighted a logo.

Findings from the survey on job attribute importance

At the end of the study, each participant completed two multiple choice surveys. One survey asked the participants to *"Imagine you are looking for a new job. Rate the usefulness of each piece of information about a job to help you decide where to apply?"* and the other to *"Imagine you are looking for a new job using an online search engine (e.g. Indeed, LinkedIn, Glassdoor, Monster, etc.). What information would you like to see on the search results page?"*. The candidate attributes were synthesized by coding a sample of 100 job postings and by analyzing the SUIs from popular online job search engines before the study. In total, we identified 28 different attributes. We present the results aggregated at a user level in Table 4.1. The values above four points on a 5-point Likert scale and the attributes with the cumulative score above eight points from both surveys are highlighted.

As we can see, the most popular attributes are **job type**, **company**, **job title**, **required skills**, **location** (country and city), **educational requirements** (degree and major), **responsibilities**, and **required years of experience**. By exploring the attributes with high variance, we also found that these are typically attributes with bimodal preferences (e.g. US citizens don't need visa sponsorship, while all international employees do). One can notice that the scores assigned to attributes in Survey 2 are usually lower, which implies that the participants were more selective and wanted to see only the most important information on the SERP because of the limited presentation space.

4.1.3 Summary of Results

To summarize, following the user-centric design process, we conducted a mixed-method user study ("think-aloud" protocol, data annotation by the participants, and surveys). Our study revealed that in addition to the attributes currently presented on the SERP (job title, company, location, job type), the participants paid attention to responsibilities and requirements (skills, degree, major, years of work experience, work authorization). In the next section, we will present the algorithm that can extract this additional information from job descriptions automatically and requires minimal supervision during the training stage.

Job Attribute \ Survey ID	**Rank**	**Survey 1**	**Survey 2**
Job Type (Full-time, Contract, Internship)	1	**4.69 ± 0.67**	**4.69 ± 0.54**
Company	2	**4.34 ± 0.84**	**4.65 ± 0.89**
Job Title	3	**4.15 ± 1.12**	**4.69 ± 0.47**
Required Skills	4	**4.50 ± 0.64**	**4.15 ± 1.04**
City	5	**4.03 ± 0.91**	**4.38 ± 0.69**
Educational Requirements (Degree & Major)	6	**4.26 ± 0.77**	**4.03 ± 1.18**
Responsibilities / Duties	7	**4.53 ± 0.85**	3.88 ± 1.30
Required Years of Experience / Seniority	8	**4.34 ± 0.89**	3.92 ± 1.19
Country	9	**4.38 ± 0.85**	3.76 ± 1.45
Salary Range	10	**4.11 ± 0.81**	3.38 ± 1.09
Industry (e.g. Banking, Internet, Telecom, FMCG)	11	3.96 ± 1.03	3.26 ± 1.53
State / Region (e.g. Midwest, West Coast, East Coast)	12	3.76 ± 1.06	3.00 ± 1.35
Perks and Benefits	13	3.61 ± 0.75	2.76 ± 1.21
Company Mission	14	3.61 ± 1.13	2.38 ± 1.47
Average (Crowdsourced) Rating / Company Review	15	3.46 ± 1.02	2.69 ± 1.28
Company Size (e.g. 1-10, 10-100, 100-1000)	16	3.38 ± 1.16	2.26 ± 1.21
Company Type (e.g. Private, Public, Startup)	17	3.34 ± 1.19	2.69 ± 1.34
People I Know Working in this Company	18	3.30 ± 1.22	2.57 ± 1.33
(short) Company History / About Us	19	3.23 ± 1.14	2.42 ± 1.39
Company Revenue	20	3.15 ± 1.12	2.23 ± 1.36
Link to Company Website	21	3.11 ± 1.60	3.34 ± 1.54
Ready to Sponsor Visa	22	2.84 ±1.82	2.69 ± 1.82
Date the Job was Posted	23	2.76 ± 1.33	3.30 ± 1.34
Distance From Me (X miles around my location)	24	2.69 ± 1.54	2.15 ± 1.22
Hiring Manager is an Alumnus from the Same School	25	2.57 ± 1.33	2.33 ± 1.03
Zip/Postal Code	26	2.23 ± 1.50	1.96 ± 1.31
Company Logo	27	1.88 ± 1.21	2.23 ± 1.24
Founded in Year	28	1.88 ± 1.07	1.53 ± 0.94

Table 4.1: Job attribute importance based on the surveys (26 participants participated in each survey).

4.2 Weakly-supervised Approach for Job Posting Segmentation Based on Big Data Redundancy

To address **RQ7(a)** (*"How can we automatically extract job responsibilities and requirements from an unstructured job posting with minimal supervision?"*), one can build an information extraction model following the ideas from [94, 147]. The problem is that despite aiming to minimize labeling efforts, this approach still requires some training to be conducted manually. The problem becomes more severe if we take into account the fact that major job search engines and PSNs operate internationally, and hence, training sets must be created for each language. To handle this complication, we propose a weakly-supervised approach for information extraction from job postings that requires a very minimal amount of manual data annotation (1-3 words per language) and could be used by a job search engine operating internationally.

The inspiration for our weakly-supervised approach comes from the paper [50], where the authors discuss how very large amounts of data make it possible to tackle complex problems, like machine translation, scene completion, and natural language disambiguation, with relatively intuitive algorithms. The wisdom of the paper is that *one should define a non-parametric model and statistically fit it with very large amounts of data. In this case, since "most" of the variability of the phenomenon is captured in the data, the model will have quite high predictive ability.* We first present a method for the training set construction from a collection of job postings, which is the key trick of this work, and then we describe a machine learning algorithm, which uses this training set to build a model for information extraction from job postings. Since we want to generate snippets which contain information from the responsibilities and requirements sections, our training set must contain positive examples from these sections (target classes) and also have some negative examples representing the irrelevant content.

Our approach to generate the training set with minimal cost is based on the observation that despite being free text documents, job postings mainly consist of the following sections *in the presented order*: short attributes (job title, company, location, etc.), responsibilities, requirements, and company description. Moreover, some job postings are so well-structured that just one declarative rule is enough to extract information from them. Specifically, we noticed that a reasonable percentage of pages (about 10% based on experiments) contain a section header[3] followed immediately by a set of sentences wrapped in a $< li >$ tag. In other words, our extraction rule works as follows. For a job posting HTML page containing $< h3 > Responsibilities :< /h3 >< ul >< li > X < /li >< /ul >$, X is a training instance for the "Responsibilities" class (Figure 4.3). We can extract such sentences and use them during the training stage as positive instances for the corresponding class. Negative instances ("Others" class) could be generated by taking sentences preceding the responsibilities and following the requirements sections from pages, where these sections were detected by the strict rule defined above. The more job postings are crawled and added to a search index, the more training instances could be generated. In the limit, we could potentially "remember" all things companies look for in candidates and achieve very high extraction quality. To summarize, we only have to specify one word per section per language to bootstrap a sufficiently large dataset.

Once the training set is available, the learning task can be formally stated as follows. Given a labeled set of sentences from the responsibilities, requirements, and "others" sections of job postings, create an algorithm that can identify important sentences from a new job posting and assign them to the corresponding sections. Each sentence is used independently as an instance for a machine learning algorithm. As a model, we use a 2-level ensemble similar to stacking. At a base level, we train 3 textual classifiers based on unigrams, bigrams, and trigrams. At a meta level, we mix binarized predictions of the base algorithms as well as

[3]Identifiable with just one seed word per section, like "Responsibilities:" and "Requirements:" used in our experiments.

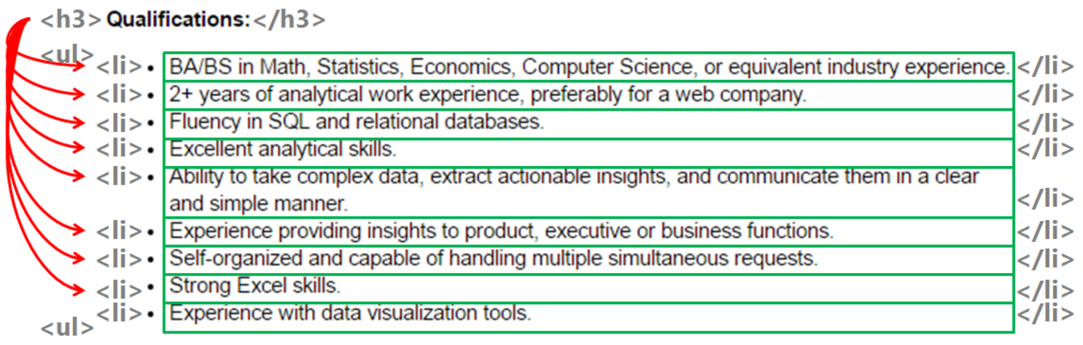

```
<h3> Qualifications:</h3>
<ul>
<li> • BA/BS in Math, Statistics, Economics, Computer Science, or equivalent industry experience. </li>
<li> • 2+ years of analytical work experience, preferably for a web company. </li>
<li> • Fluency in SQL and relational databases. </li>
<li> • Excellent analytical skills. </li>
<li> • Ability to take complex data, extract actionable insights, and communicate them in a clear and simple manner. </li>
<li> • Experience providing insights to product, executive or business functions. </li>
<li> • Self-organized and capable of handling multiple simultaneous requests. </li>
<li> • Strong Excel skills. </li>
<li> • Experience with data visualization tools. </li>
<ul>
```

Figure 4.3: A highly precise extraction pattern consists of a job posting section header word inside an HTML header tag and a list tag that contains the target sentence corresponding to this section and used for information extraction model training.

linguistic features, e.g. sentence length, capitalization, POS tags. As a learning algorithm, at both levels, we use SVM [135] with a linear kernel, which scales to very large datasets, and a Cutting-Plane Algorithm described in [65]. To further speed up computation, we use feature hashing as described in [78].

4.2.1 Experiment for Offline Evaluation of Information Extraction Quality

To evaluate the proposed approach, we crawled a corpus of 1,101,482 pages representing job postings. The set of URLs was generated by submitting top job titles (`http://www.indeed.com/find-jobs.jsp`) to Indeed.com and web scraping the corresponding SERPs. The destination pages for each URL were downloaded and stored locally. We then selected ten distinct job titles spanning multiple industries (16,054 pages total) and annotated 100 randomly sampled pages for each. The annotation speed with the custom-built annotation tool was about two minutes per page, which demonstrates the cost one must accommodate if s/he follows a classical supervised approach. We used ten-fold cross validation for the supervised approach evaluation. The proposed weakly-supervised approach was trained on the sentences automatically extracted from 16,054 pages excluding the labeled pages to make sure that we evaluated the performance fairly. Then, we applied the algorithm on the same fold as the supervised algorithm and repeated the procedure ten times. Since our ultimate goal is to use the extracted information to generate snippets, which is a precision-oriented task (due to the limited space available on a SERP), we tracked two metrics: *Precision* averaged across job titles and *Page Coverage* (the percentage of pages for which at least one relevant sentence was extracted).

4.2.2 Results

First, we found that the model trained on the dataset created using our weakly-supervised approach performed as good as the model trained on a corpus of 1,000 job postings labeled manually (Table 4.2), which

Job Section	Metric	Manual Annotation	Weakly-supervised Approach
Responsibilities	Precision	0.86 ± 0.09	0.84 ± 0.08
	Page Coverage	0.80 ± 0.12	**0.92 ± 0.07*****
Requirements	Precision	0.91 ± 0.07	0.88 ± 0.07
	Page Coverage	0.94 ± 0.05	**0.97 ± 0.04*****

Table 4.2: The evaluation of information extraction quality. The proposed weakly-supervised approach achieves similar values for *Precision* and has higher *Page Coverage* due to much bigger training set size.

implies that the proposed approach can be used to effectively and efficiently create training sets for information extraction algorithms from job postings. Second, one can notice that the proposed approach for the training set generation yields higher *Page Coverage* (Table 4.2) and the difference is statistically significant using a paired two-tailed t-test with $p = 0.05$. Generalizing the results from [6], who showed that simpler algorithms with more data outperform more complex algorithms, we hypothesize that we achieve better results also because of the larger training set. Third, we graphed the performance of the proposed weakly-supervised approach for different job titles (Figure 4.4) and found that it consistently extracts correct information from new job postings. It implies that the approach generalizes to the entire job domain.

To summarize, in this section we demonstrated that we can perform accurate information extraction from unstructured job postings and turn them into the structured representation with minimal supervision. This structured representation can be used to generate informative structured snippets by showing on the SERP not only job titles, companies, and locations but also requirements and responsibilities, which we found to be equally important as described in Section 4.1.

4.3 Extended Informative Structured Snippet Generation

In this section, we describe our efforts to evaluate the utility of extended informative structured snippets, which can be generated by having access to the structured representation generated with the approach presented in the previous Section 4.2. Our evaluation is based on the online A/B test with real users of HH.ru (the largest job search engine in Russia and CIS[4] region). Since HH.ru primarily serves job postings written in the Russian language, we had to adapt our approach from English to Russian. Therefore, we also report on how easily it generalizes to new languages. Our ultimate goal in this section is to answer our research questions **RQ8(a)** (*"Do extended informative structured snippets improve search UX for job search?"*) and **RQ8(b)** (*"How do users behave when such structured snippets are used?"*).

[4]Commonwealth of Independent States

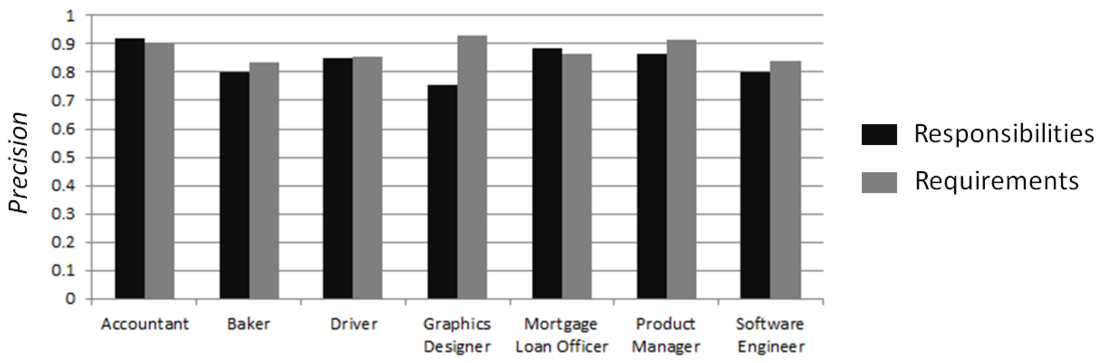


Figure 4.4: Information extraction Precision by the proposed weakly-supervised approach stably achieves high values across various job titles. Black bars represent Responsibilities, gray bars represent Requirements.

4.3.1 Adapting the Approach to New Languages

We were able to get access to the dataset of over 4,000,000 job postings provided by HH.ru. We first generated a training set for the information extraction model from Russian language job postings using our weakly-supervised approach by "seeding" it with just three most popular section words: the one, which is translated as *"Responsibilities"*, the one, which is translated as *"Requirements"*, and the one, which is translated as *"Conditions"*[5]. As a result, we were able to cover 16.3% of the data set (652,283 job postings)[6]. The training set contained 3,462,734 sentences for different sections of the job posting (requirements, responsibilities, conditions, other). We then trained the algorithm described in the previous section (an ensemble of linear SVMs on top of N-grams) and calculated the *Precision* for different sections. We got 91.38% for the responsibilities section, 90.15% for the requirements section, and 93.66% for the conditions section. These numbers imply that on average 7-10 out of 100 job postings were parsed incorrectly. This level of performance wasn't acceptable for the industrial user-facing service.

Therefore, we experimented with various ways to increase the performance of information extraction. First, we increased the size of the training set by adding more extraction patterns (section words). For example, in addition to using the translated version of *"Responsibilities"*, we also used the synonyms, like *"Duties"*, *"Key Accountabilities"*, *"Key Responsibilities"*, and etc. Eventually, we added around 50 patterns per section and covered section headers for almost the entire data set. It is worth mentioning that, as expected, we observed the effect of diminishing returns (Figure 4.5), i.e. with extra added patterns the amount of new covered pages that can be used for training didn't grow as significantly. We retrained the algorithm on 10,000,000 training instances and got the following *Precision* scores: 95.72% for the responsibilities section,

[5]due to the specifics of the local recruiting landscape, this section is known to be important and we were recommended to include it in the snippets by a HH.ru representative.

[6]precisely, we covered the section headers, but not job postings since not all job postings have a structured layout making the automated extraction with simple patterns infeasible.

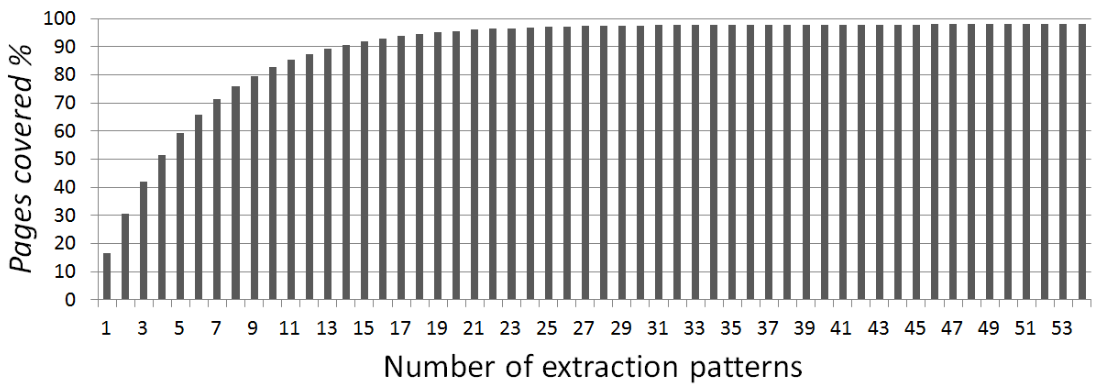


Figure 4.5: Section headers' coverage with the extraction patterns as a function of the number of patterns.

94.02% for the requirements section, and for 95.93% for the conditions section, which implies that on average 4-5 out of 100 job postings were parsed incorrectly. This level of performance was still unacceptable.

Second, we turned to the hybrid approach that combined highly precise extraction patterns created manually and machine learning as a "catch-all" back-off model. Specifically, for a new job posting, we first tried to parse it using the extraction patterns and only if none of the extraction patterns worked, we used the prediction from the machine learning model. This hybrid approach allowed to increase the *Precision* for information extraction to 96.64% for the requirements section, 96.17% for the responsibilities section, and 98.11% for the conditions section, which means that only 2-3 out of 100 job postings were parsed incorrectly. With this performance, we were able to move forward.

4.3.2 Assembling Informative Structured Snippets According to the Maximum Marginal Relevance Principle

In this section, we answer our **RQ7(b)** (*"How can we generate an extended informative structured snippet for a job posting having a list of job responsibilities and requirements in a structured form?"*). Having access to the structured (fielded) representations of job postings, we generated extended informative structured snippets in a query-biased manner by following the Maximum Marginal Relevance Principle [21]. Given a query submitted by the user, for each section (responsibilities, requirements, conditions) we picked the sentence, which had the highest similarity with the query, and, then, we assembled the rest of the snippet by taking the sentences maximizing the cumulative utility of the snippet according to the formula:

$$s_{k+1} = \arg \max_{s_i \in S \setminus S_k} \left\{ \lambda * sim_1(s_i, Q) - (1-\lambda) * \max_{s_j \in S_k} \{ sim_2(s_i, s_j) \} \right\},$$

where s_{k+1} is the next sentence to be selected; S are all sentences for a given section, S_k are the sentences selected so far, sim_1 and sim_2 are the corresponding similarity functions between the query and the sentences (the cosine similarity is used in our case), and λ is a linear mixing factor balancing between relevance to the query and redundancy of the sentences in the snippet (we set it to 0.5).

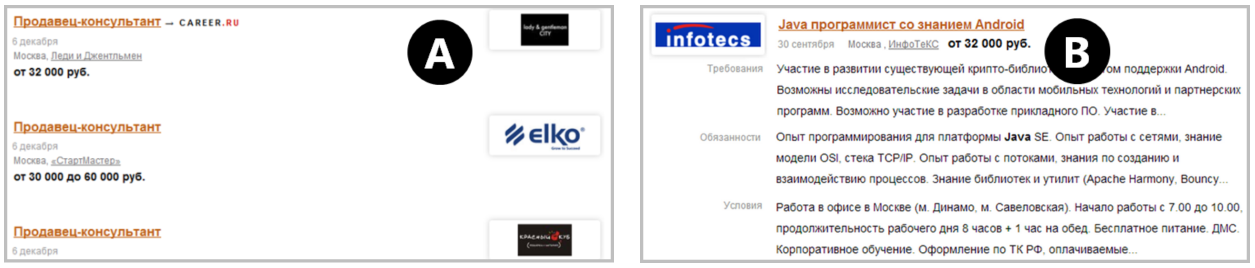
4.3.3 A/B Experiment for Online Evaluation of Informative Structured Snippet Utility

The online A/B experiment was conducted during 12 days in November 2013. Two different versions of the snippet were considered (Figure 4.6): (A) the one, with the traditional minimalistic snippets showing only job title, company, location, posting date, and salary; (B) the extended one, additionally showing requirements, responsibilities, and perks & conditions information. 1/64 of all unique users were sent to the experimental interface with the randomization at the IP level. All user actions were logged. The log records consisted of the following data fields: *user_id*, *snippet_version*, *action_id* (click, page view, search), *timestamp*, *meta_params* (service field containing context about a user, a query, a region, and other features relevant for search personalization). In total, 542,298 unique users participated in the study.

Implementation Details

We followed the service oriented architecture (SOA) and exposed the proposed algorithm as a private RESTful API. Every time a user visited HH.ru and submitted a search query, HH.ru sent a request to our API with the goal to get structured snippets for job postings retrieved from the HH.ru's index. Each request consisted of a query and 20 unstructured job postings. As output, we returned a list of 20 snippets, one for each search result. To avoid unnecessary computation, we used Redis[7] to build a simple cache layer. As a key, we used a query and a job posting ID. The value contained a query-biased snippet for the query and job posting pair. For machine learning, we used a scalable implementation of LibSVM developed specifically for short texts [154]. The API was hosted in Microsoft Azure Cloud on a machine with 16GB RAM and two 1.7GHz CPU cores. The API was proxied behind Nginx[8], and Gunicorn[9] was used as a web server. We performed load test and found that the described configuration was able to sustain the required portion of the traffic (around 100,000 requests per day) and serve the snippets for each request under 0.7 seconds[10]

[7]`http://redis.io/`
[8]`https://www.nginx.com/`
[9]`http://gunicorn.org/`
[10]HH.ru set 2 seconds as a service level agreement (SLA).



Продавец-консультант → CAREER.RU
6 декабря
Москва, Леди и Джентльмен
от 32 000 руб.

Продавец-консультант
6 декабря
Москва, «СтартМастер»
от 30 000 до 60 000 руб.

Продавец-консультант
6 декабря

Java программист со знанием Android
30 сентября Москва , ИнфоТеКС **от 32 000 руб.**

Требования	Участие в развитии существующей крипто-библиот ом поддержки Android. Возможны исследовательские задачи в области мобильных технологий и партнерских программ. Возможно участие в разработке прикладного ПО. Участие в...
Обязанности	Опыт программирования для платформы **Java** SE. Опыт работы с сетями, знание модели OSI, стека TCP/IP. Опыт работы с потоками, знания по созданию и взаимодействию процессов. Знание библиотек и утилит (Apache Harmony, Bouncy...
Условия	Работа в офисе в Москве (м. Динамо, м. Савеловская). Начало работы с 7.00 до 10.00, продолжительность рабочего дня 8 часов + 1 час на обед. Бесплатное питание. ДМС. Корпоративное обучение. Оформление по ТК РФ, оплачиваемые...

Figure 4.6: Two experimental snippet versions. *(Left)* Current snippets used at HH.ru showing job title, location, company, and posting date. *(Right)* Extended informative structured snippets additionally showing information about job responsibilities, requirements, and conditions.

4.3.4 Results

The summary of results from the A/B experiment is presented in Table 4.3.[11] We found that the version B with extended structured snippets led to better search users experience according to the multitude of tracked metrics. The number of queries per user decreased by 8% and the number of queries before the first apply action for a new unique user in the A/B experiment decreased by 1.4 times. These findings imply that with the extended informative structured snippets we increased the efficiency of search and the engagement with the SERP. In other words, users were able to accomplish their search tasks faster and made more informed decisions while interacting with the SERP. We also found that the number of short clicks (when a user clicks on a link on the SERP, visits the page, and then quickly closes it) decreased by 5.5% and that the number of wasted views (open a job posting page but don't make any actions) decreased by 1.25 times. At the same time, the number of applications overall increased by 1.6% (one of the key metrics for a job search engine related to revenue) and the application rate conditioned on click increased by 13%, which means that users were opening more relevant job postings from the SERP. By analyzing the SERP interaction logs, we discovered that click entropy decreased by 1.98 times. This finding implies that with structured snippets we denoise click data and could perform more effective relevance feedback calculation improving the search user experience within and across sessions. Finally, we found that the number of detailed page views decreased by 1.4 times. By combining this fact with the finding that overall the number of applications increased by 1.6%, we again conclude that the proposed snippets make users more efficient at search. Two most important graphs from the A/B experiment are presented in Figure 4.7.

[11]we don't report the absolute numbers because this information is confidential.

Metric	Relative Change
Average Number of Queries per User	−8%
Average Number of Detailed Page Views Per Query	−28.6%
Average Number of Short Clicks per Query	−5.5%
Average Number of Wasted Views per Query	−20%
Average Number of Queries before the First Apply Action	−27.5%
Average Application Rate Conditioned on SERP Click per Query	+13%
Number of Applications Overall	+1.6%
Average Click Entropy per Query	−49.5%

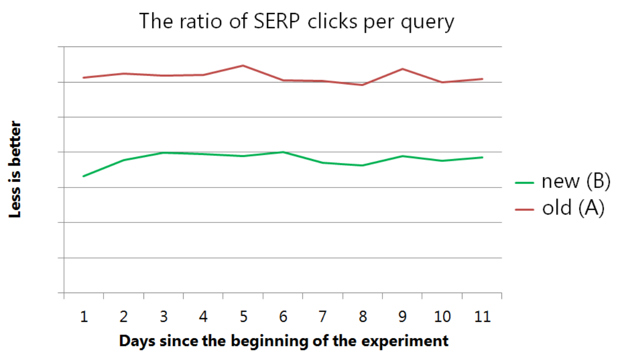
Table 4.3: Statistics for the A/B experiment conducted with HH.ru.

4.4 Discussion

According to the A/B test, the majority of analyzed metrics related to search user experience and satisfaction improved significantly demonstrating the usefulness of extended informative structured snippets for the users. However, the number of detailed page views decreased by 1.4 times. Therefore, the decision to use the proposed snippets in practice is not obvious and depends on the business model of a job search engine company and its long-term objectives. For a search engine, which uses advertising as a primary revenue source (implies that more page views are usually better), there are no reasons to integrate this innovation in the short-term as it will very likely to decrease revenue. We still recommend using the proposed snippets since otherwise the users might go away and switch to competitors, who focus on superior user experience rather than short-term revenue.

The extended informative structured snippets that we proposed are significantly longer than the snippets currently used by job search engines, and therefore, the users might get tired over time from viewing too much information on the SERP. We think that a longitudinal study or a longer A/B test might be necessary to investigate the way people react to the extended snippets over time. Plus, in addition to the query level analysis described in this section, we think that session level analysis might be necessary as it might reveal new insights regarding the way people engage with the snippets during a more prolonged period.

Looking more broadly at all projects related to snippets generation and usage, we warn search engine companies about possible copyright issues since they typically show the content from original websites without an official permission to do so (it is simply not feasible to do at scale). That said, we think that the value (measured as visibility, extra traffic, and "leads") provided by search engines to the original content owners significantly outweighs possible harm.



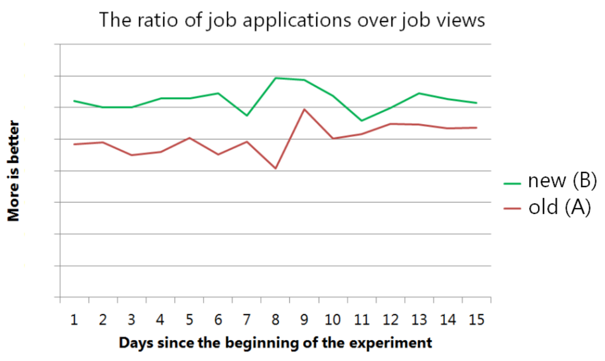


Figure 4.7: The key metrics from the A/B test in the graphical form over time. *(Left)* The ratio of SERP clicks per query for two experimental snippet versions. *(Right)* The ratio of job applications to job views (SERP clicks).

4.5 Limitations and Future Work

We do acknowledge several limitations of the study described in this Chapter 4. First, to answer our research questions **RQ6(a)** and **RQ6(b)**, we conducted the user study in the USA. However, the live A/B test to evaluate the utility of extended informative structured snippets was conducted in Russia due to the difficulty finding a research partner for the study in the USA. Therefore, the user needs and search behavior might be different because of the local job market specifics. For example, we were suggested to include the "Perks and Conditions" information to the snippets during the A/B test in Russia while this attribute didn't popup in the user study conducted in the USA (perhaps, it doesn't serve as a differentiator in that job market). We would like to setup an A/B test in the USA and evaluate the utility of the proposed snippet type for this user demographic to ensure consistency between the user study and the A/B test.

Second, it is important to note that with the limited budget and time in our A/B experiment we compared only two versions of the snippet (the shortest one with the minimal number of attributes per page and the longest one with three additional textual attributes including responsibilities, requirements, and conditions). It might have created invisible interactions between various elements of the snippet and led to confounding. Plus, the design space is much larger, and there are many other possible snippet versions. For example, the version extending the default snippet only with information from the requirements section is very promising because we found it to be the most important attribute among the ones not already shown by job search engines (Section 4.1.2). Plus, it is shorter than the experimental extended snippet version studied in Chapter 4. Therefore, it would be interesting to see a series of follow-up A/B experiments that systematically explore various snippet versions and test atomic snippet modifications. To accomplish this, we must decide: (1) how many attributes do we show in a snippet? (2) what attributes do we show? Given a set of candidate job attributes, we will first find the optimal snippet length and then find what combination of job attributes is the best by adding subsets of candidate job attributes to the existing snippet version. For example, assuming that the maximum allowed snippet length is two and given *responsibilities*, *requirements*, and *perks* attributes, we will test *"{responsibilities}"*, *"{requirements}"*, *"{perks}"*, *"{responsibilities, requirements}"*, *"{responsibilities, perks}"*, *"{requirements, perks}"* subsets. We also think that a task-oriented interactive user study, similar to the one described in Chapter 5, or an eye-tracking study, similar to [28], might reveal additional insights.

4.6 Conclusions

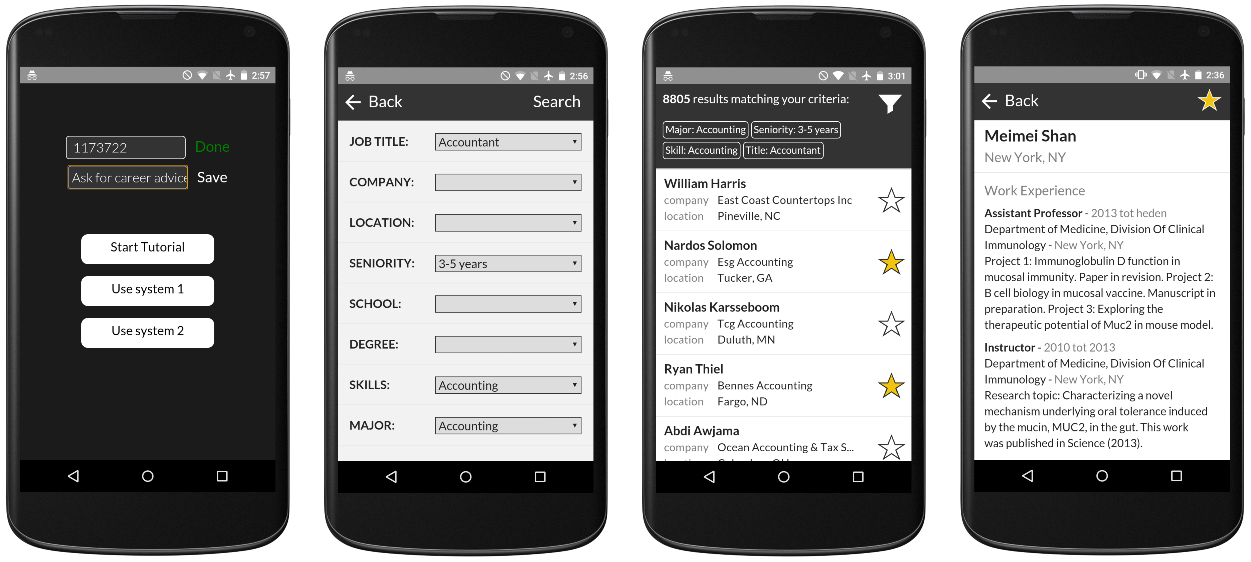
In this Chapter 4, we described the user need elicitation study, proposed the new approach for information extraction from job postings, introduced the new snippet format, and documented the results from the large scale online A/B experiment that we conducted to ultimately evaluate the value of the proposed ideas. The key takeaways from this Chapter 4 are four-fold. First, in addition to the attributes that are currently shown by job search engines on the SERP, users consider responsibilities and requirements as very important. Second, by leveraging big data redundancy, we can generate large scale annotated datasets made of job postings with minimum supervision. These datasets can then be used to train machine learning models that can automatically detect and extract information from the responsibilities and requirements sections of unstructured job postings. The experimental results show that we can achieve very high extraction quality. Third, the proposed weakly-supervised approach for information extraction can be easily adapted to new languages, and hence, it is useful for job search engines operating internationally. Fourth, the proposed extended informative structured snippets improve search user experience by showing the most important information right on the SERP and, hence, making users more effective and efficient at job search.

Chapter 5

Non-redundant Delta Snippets for Job and People Search

In the previous Chapter 4, we introduced *extended informative structured snippets* for job search, which according to our comprehensive experiments led to the improvements in search user experience and higher SERP utility by presenting the most critical information right on the SERP and eliminating unnecessary title-snippet redundancy. In this Chapter 5, we continue our investigation of non-redundant snippets and describe a study aimed to address more general questions **RQ9-RQ10**. First, in answering **RQ9** (*'What kind of snippets make users more productive and effective when performing structured search (e.g. job, people) on mobile devices?"*), we built an experimental people search application and conducted a task-oriented interactive user study with 39 participants. Four different versions of the snippet were compared by varying the snippet type (query-biased vs. non-redundant) and the snippet length (two vs. four attributes per results on the SERP). We adopted a within-subjects experiment design and made each participant do four realistic search tasks using different versions of the application. During the study sessions, we collected search logs, "think-aloud" comments, and post-task surveys. Each session was finalized with an interview. We analyzed search query log data, post-task subjective relevance judgments and surveys to compare the effectiveness and efficiency of various snippet versions. Second, in answering **RQ10** (*"Do users prefer non-redundant delta snippets or query-biased snippets based on their subjective feelings?"*), we further analyzed subjective post-tasks surveys, "think-aloud" comments, and interview data from the same group of participants. By the end of this Chapter 5, we will understand why and what snippets to show to the users performing structured search, especially on mobile devices.

In the next sections, we describe the experimental system (Section 5.1) and our research method (Section 5.2). Then, we present the results from the pilot user study in Section 5.3. Starting from Section 5.4, we share multiple insights from the formal user study. Specifically, we describe the results from the interviews in Section 5.4.1, System Effectiveness in Section 5.4.2, System Efficiency in Section 5.4.3. Finally, we present relevant search strategies used by the study participants in Section 5.4.4. We discuss the results, study limitations, and present our SUI design recommendations and ideas for future work in Section 5.5.


1173722 Done

Ask for career advice Save

JOB TITLE: Accountant

COMPANY:

LOCATION:

SENIORITY: 3-5 years

SCHOOL:

DEGREE:

SKILLS: Accounting

MAJOR: Accounting

8805 results matching your criteria:

Major: Accounting | Seniority: 3-5 years | Skill: Accounting | Title: Accountant

William Harris
company East Coast Countertops Inc
location Pineville, NC

Nardos Solomon
company Esg Accounting
location Tucker, GA

Nikolas Karsseboom
company Tcg Accounting
location Duluth, MN

Ryan Thiel
company Bennes Accounting
location Fargo, ND

Abdi Awjama
company Ocean Accounting & Tax S...

Meimei Shan
New York, NY

Work Experience

Assistant Professor - 2013 tot heden
Department of Medicine, Division Of Clinical Immunology - New York, NY
Project 1: Immunoglobulin D function in mucosal immunity. Paper in revision. Project 2: B cell biology in mucosal vaccine. Manuscript in preparation. Project 3: Exploring the therapeutic potential of Muc2 in mouse model.

Instructor - 2010 tot 2013
Department of Medicine, Division Of Clinical Immunology - New York, NY
Research topic: Characterizing a novel mechanism underlying oral tolerance induced by the mucin, MUC2, in the gut. This work was published in Science (2013).

Figure 5.1: Four primary views of the experimental mobile people search application: (left) Welcome / Settings page; (2nd left) Filters / Query Formulation page; (2nd right) SERP showing the number of search results, current query constraints ("breadcrumbs"); (right) Resume / Detailed Profile page.

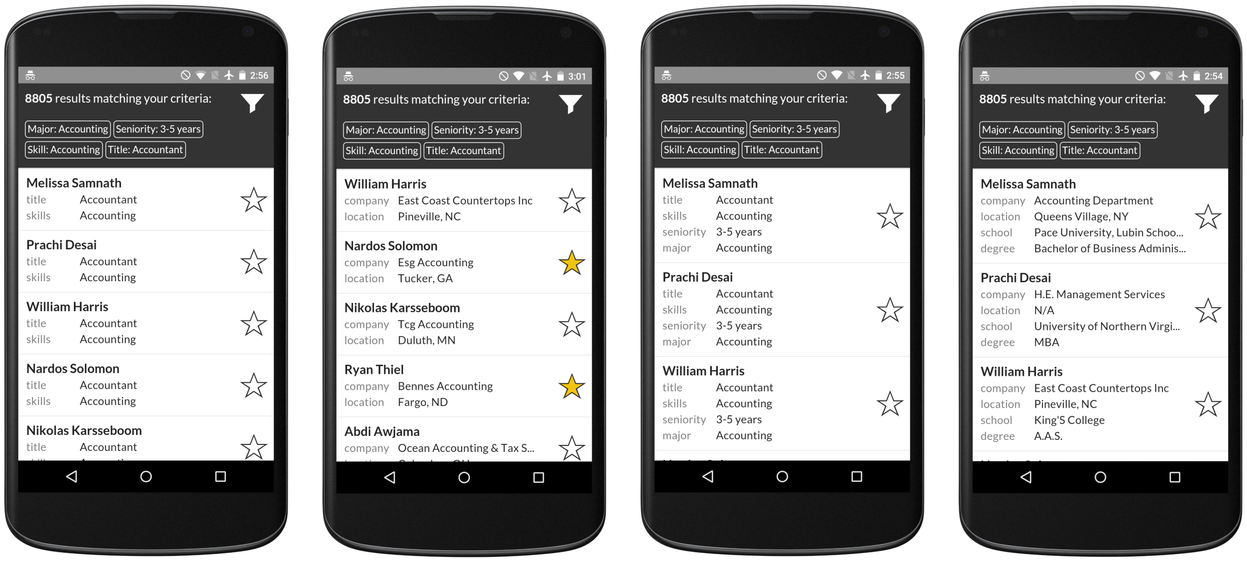
5.1 Experimental System

To answer our research questions, we built an experimental mobile structured search application. We decided to focus on *People* search as it is the most popular search vertical in social networks (Chapter 2), which contain a lot of structured data on people. Moreover, people search is by nature an exploratory task [52,83], and hence, it is interesting to understand whether users benefit from having extra non-redundant information in the snippets. We considered job search as a possible alternative vertical for the study but discarded it later since it was harder to come up with independent search tasks. We didn't want to use the repeated tasks to minimize the influence of learning effects during the study sessions.

In this section, we describe the specifics of the search user interface (most importantly, query formulation interface and SERP), the dataset, and the implementation details.

5.1.1 Search User Interfaces

Our app consists of four screens/pages (Figure 5.1): (1) *Settings*, where users can enter their anonymized userID, select the task type, and the version of the SERP; (2) *SERP*; (3) *Filters*, where users can specify search criteria; and (4) *Detailed Profile* with the detailed information about the particular search result. *Settings*, *Filters*, and *User Profile* screens are the same in all versions of the experimental system, and there are four different versions of the SERP (Figure 5.2):


8805 results matching your criteria:

Major: Accounting | Seniority: 3-5 years | Skill: Accounting | Title: Accountant

Melissa Samnath	title	Accountant
	skills	Accounting
Prachi Desai	title	Accountant
	skills	Accounting
William Harris	title	Accountant
	skills	Accounting
Nardos Solomon	title	Accountant
	skills	Accounting
Nikolas Karsseboom	title	Accountant

8805 results matching your criteria:

Major: Accounting | Seniority: 3-5 years | Skill: Accounting | Title: Accountant

William Harris	company	East Coast Countertops Inc
	location	Pineville, NC
Nardos Solomon	company	Esg Accounting
	location	Tucker, GA
Nikolas Karsseboom	company	Tcg Accounting
	location	Duluth, MN
Ryan Thiel	company	Bennes Accounting
	location	Fargo, ND
Abdi Awjama	company	Ocean Accounting & Tax S...

8805 results matching your criteria:

Major: Accounting | Seniority: 3-5 years | Skill: Accounting | Title: Accountant

Melissa Samnath	title	Accountant
	skills	Accounting
	seniority	3-5 years
	major	Accounting
Prachi Desai	title	Accountant
	skills	Accounting
	seniority	3-5 years
	major	Accounting
William Harris	title	Accountant
	skills	Accounting
	seniority	3-5 years
	major	Accounting

8805 results matching your criteria:

Major: Accounting | Seniority: 3-5 years | Skill: Accounting | Title: Accountant

Melissa Samnath	company	Accounting Department
	location	Queens Village, NY
	school	Pace University, Lubin Schoo...
	degree	Bachelor of Business Adminis...
Prachi Desai	company	H.E. Management Services
	location	N/A
	school	University of Northern Virgi...
	degree	MBA
William Harris	company	East Coast Countertops Inc
	location	Pineville, NC
	school	King'S College
	degree	A.A.S.

Figure 5.2: Four different versions of the SERP: (QB2) *query-biased* snippets with *two* attribute slots per result; (NR2) *non-redundant* snippets with *two* attribute slots per result; (QB4) *query-biased* snippets with *four* attribute slots per result; (NR4) *non-redundant* snippets with *four* attribute slots per result.

- one, which shows traditional **query-biased** snippets and has **two lines** for result attributes (QB2);
- one, which shows **non-redundant** snippets and has **two lines** for result attributes (NR2);
- one, which shows traditional **query-biased** snippets and has **four lines** for result attributes (QB4);
- one, which shows **non-redundant** snippets and has **four lines** for result attributes (NR4);

A user can interact with the SERP in several different ways: (a) to visit a detailed profile page, s/he can click on the search result on the SERP; (b) to scroll through the results, s/he can swipe up and down; (c) to request more results, s/he can click the *"Show more results"* button at the bottom of the SERP; (d) to save a search result for future reference, s/he can click on a *Star* shown next to the result snippet. We intentionally removed profile images because we wanted to focus user's attention on the snippets and eliminate unnecessary noise from the collected data. If we kept images on the SERP, the user would get distracted by the images lowering the signal-to-noise ratio.

To submit a new query, the user has to open the *Filters* screen and set constraints on the attribute values using drop-down menus. Then, s/he can either submit a new query or go back to the original SERP without changing the query. We don't support keyword search and only allow users to submit queries by selecting constraints via filters. Being limiting, such an approach to query formulation is common in popular mobile search apps over structured data (e.g. Zappos). The user can filter entities by using eight different attributes, namely *Job Title*, *Company*, *University*, *Degree*, *Location*, *Skills*, *Seniority / Years of Experience*, and *Major / Field of Study.* The SUI is designed in such a way that only one attribute value can be selected at a time. We decided to adopt this strategy to get maximum control over the collected data. Otherwise, if the user has selected several values per attribute, we could not have shown a non-redundant snippet for that attribute, because we must differentiate one result from the rest. Such queries must have been ignored, which would be costly, especially given our limited supply of participants and experimental budget. Finally, there is a special *Favorites* filter which overrides all other filters and returns only the "starred" profiles (added to favorites). The search criteria are always shown as the tags in the navigation bar. The navigation bar is fixed and always stays at the top of the SERP as users scroll down.

Query-biased snippets are generated as follows. First, we take all attributes specified by the user and sort them by priority (some global score for attribute importance that could be generated by computing the frequency of filter usage or based on a survey/interview). Then, we append all other attributes that aren't mentioned in the query also sorted by priority. Finally, we take as many attributes from the top of this concatenated attribute list as there are snippet lines in a specific version of the SERP. For non-redundant snippets, the algorithm is almost the same. The only difference is that we remove the attributes that are defined in the query.

5.1.2 Experimental Collection

By analyzing HTTP requests exchanged between the client and the server, we learned how to communicate with Indeed's resume search API [1]. It allowed us to send the search requests directly to production Indeed's servers and get the results which are identical to the results shown on the actual Indeed's SERP. In other words, we use the whole collection of over 21,000,000 Indeed's resumes in our experiment.

It is worth mentioning that Indeed is primarily a keyword search engine. However, it supports advanced query interface, which allows formulation of precise structured queries. We use this functionality while interacting with Indeed's servers to retrieve results exactly matching query constraints. For example, to retrieve only resumes from *"New York"*, we append `in-New-York-NY?co=US` parameter string to the URL.

5.1.3 Technology Stack

The front-end is a single page web application built with HTML5, CSS, JavaScript, jQuery [2], Bootstrap [3], and AngularJS [4]. Since in this study we focus on mobile search, the layout is optimized for mobile devices.

[1]http://www.indeed.com/resumes
[2]`https://jquery.com/`
[3]`http://getbootstrap.com/`
[4]`https://angularjs.org/`

We use GoNative [5] web-to-mobile SaaS to convert our web application into the mobile app. It helps get rid of the pesky browser address bar and provide superior user experience to the participants.

The back-end is built on Node.js[6] and uses MongoDB[7] as a database. MongoDB's document-oriented semi-structured model is especially useful for collecting logs as participants interact with the system. Since Indeed doesn't provide a public API for resume search, we interact with it by sending carefully tuned HTTP requests. Some responses are returned in HTML form. In such cases, we use CheerioJS [8] to do real-time HTML-to-JSON parsing. Despite the fact that we interact with the remote data provider and do real-time HTML parsing, the system can present new search results under one second. The system is deployed in DigitalOcean on an instance with Ubuntu (1GB RAM, 1 CPU core, 30GB SSD).

5.2 Method

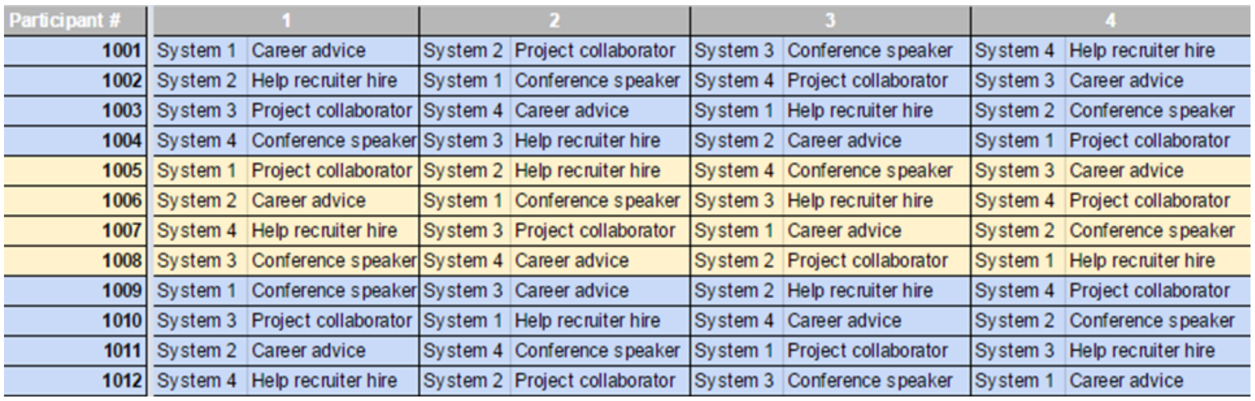
We used a simulated work task situation approach [17, 72] in our user study, i.e. we analyzed the behavior of participants seeking to accomplish real tasks by interacting with an experimental search application. The study had two stages.

In the first stage (a pilot study, 12 participants), we compared four different versions of the SERP by varying the snippet type and the snippet length[9]. Adopting a within-subjects study design, we made each participant do four different search tasks using four different versions of the application. Each participant did only one randomly assigned task using each version. The task/version order was randomized following the Greek-Latin square experiment design [124] to account for fatigue, task difficulty, and learning effects. We present the corresponding complete randomization table in Figure 5.3. The goal of the first stage was to eliminate the versions, which are clearly less favored by the participants.

In the second stage (a formal study, 24 participants), we compared only the best two versions from the first stage. This time, each participant did two out of four tasks using each version. It allowed us to increase the number of measurements per participant-version combination to two and make the analysis more solid. The goal of the second stage was to actually answer our research questions and collect rigorous data to support the conclusions. Likewise, we used the Greek-Latin square design as a randomization protocol (Figure 5.4).

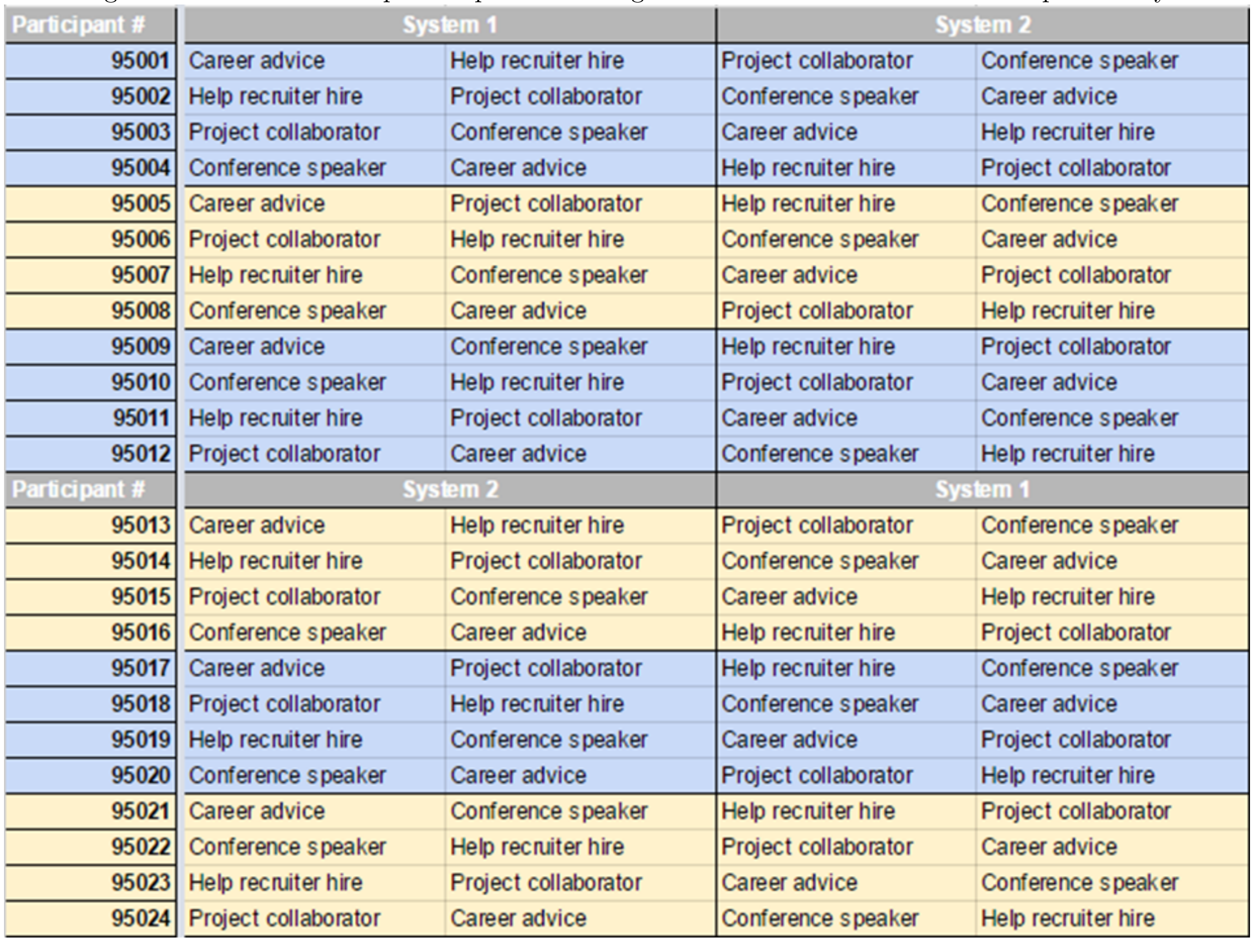
In the next section, we describe the elements of the experimental methodology which are shared across both stages. In case of differences, we make necessary clarifications. The methodology is significantly inspired by [52, 69, 145].

[5]`https://gonative.io/`
[6]`https://nodejs.org/en/`
[7]`https://www.mongodb.com/`
[8]`https://github.com/cheeriojs/cheerio`
[9]we decided to vary the snippet length and the snippet type in case there is some interaction between the two.



Participant #	1		2		3		4	
1001	System 1	Career advice	System 2	Project collaborator	System 3	Conference speaker	System 4	Help recruiter hire
1002	System 2	Help recruiter hire	System 1	Conference speaker	System 4	Project collaborator	System 3	Career advice
1003	System 3	Project collaborator	System 4	Career advice	System 1	Help recruiter hire	System 2	Conference speaker
1004	System 4	Conference speaker	System 3	Help recruiter hire	System 2	Career advice	System 1	Project collaborator
1005	System 1	Project collaborator	System 2	Help recruiter hire	System 4	Conference speaker	System 3	Career advice
1006	System 2	Career advice	System 1	Conference speaker	System 3	Help recruiter hire	System 4	Project collaborator
1007	System 4	Help recruiter hire	System 3	Project collaborator	System 1	Career advice	System 2	Conference speaker
1008	System 3	Conference speaker	System 4	Career advice	System 2	Project collaborator	System 1	Help recruiter hire
1009	System 1	Conference speaker	System 3	Career advice	System 2	Help recruiter hire	System 4	Project collaborator
1010	System 3	Project collaborator	System 1	Help recruiter hire	System 4	Career advice	System 2	Conference speaker
1011	System 2	Career advice	System 4	Conference speaker	System 1	Project collaborator	System 3	Help recruiter hire
1012	System 4	Help recruiter hire	System 2	Project collaborator	System 3	Conference speaker	System 1	Career advice

Figure 5.3: Greek-Latin square experiment design to randomize conditions for the pilot study.



Participant #	System 1		System 2	
95001	Career advice	Help recruiter hire	Project collaborator	Conference speaker
95002	Help recruiter hire	Project collaborator	Conference speaker	Career advice
95003	Project collaborator	Conference speaker	Career advice	Help recruiter hire
95004	Conference speaker	Career advice	Help recruiter hire	Project collaborator
95005	Career advice	Project collaborator	Help recruiter hire	Conference speaker
95006	Project collaborator	Help recruiter hire	Conference speaker	Career advice
95007	Help recruiter hire	Conference speaker	Career advice	Project collaborator
95008	Conference speaker	Career advice	Project collaborator	Help recruiter hire
95009	Career advice	Conference speaker	Help recruiter hire	Project collaborator
95010	Conference speaker	Help recruiter hire	Project collaborator	Career advice
95011	Help recruiter hire	Project collaborator	Career advice	Conference speaker
95012	Project collaborator	Career advice	Conference speaker	Help recruiter hire
Participant #	System 2		System 1	
95013	Career advice	Help recruiter hire	Project collaborator	Conference speaker
95014	Help recruiter hire	Project collaborator	Conference speaker	Career advice
95015	Project collaborator	Conference speaker	Career advice	Help recruiter hire
95016	Conference speaker	Career advice	Help recruiter hire	Project collaborator
95017	Career advice	Project collaborator	Help recruiter hire	Conference speaker
95018	Project collaborator	Help recruiter hire	Conference speaker	Career advice
95019	Help recruiter hire	Conference speaker	Career advice	Project collaborator
95020	Conference speaker	Career advice	Project collaborator	Help recruiter hire
95021	Career advice	Conference speaker	Help recruiter hire	Project collaborator
95022	Conference speaker	Help recruiter hire	Project collaborator	Career advice
95023	Help recruiter hire	Project collaborator	Career advice	Conference speaker
95024	Project collaborator	Career advice	Conference speaker	Help recruiter hire

Figure 5.4: Greek-Latin square experiment design to randomize conditions for the pilot study.

5.2.1 Participants

We recruited the participants by promoting the study via print ads, mailing lists, and word-of-mouth. All participants were required to be at least 18 years old, use a social networking site at least once a week, search

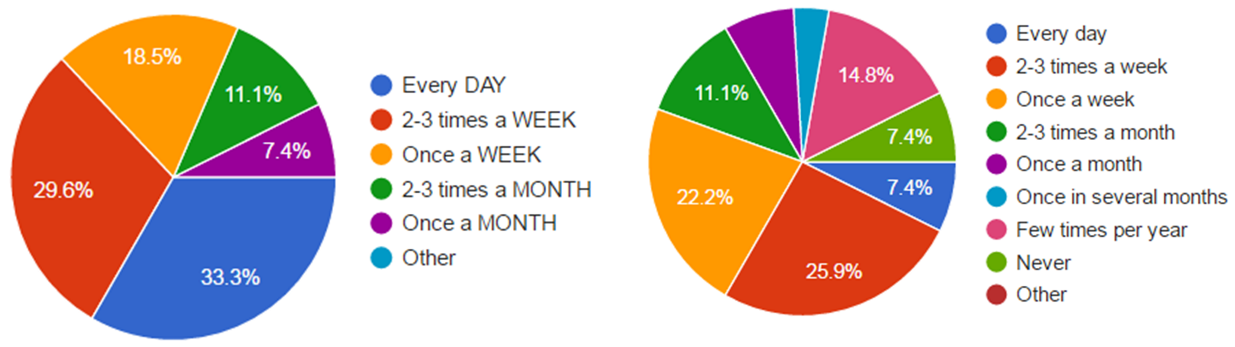


Figure 5.5: (Left) How often do you search for people online? (Right) How often do you use LinkedIn?

for people online at least once a month and have a mobile device. In total, 39 people participated in the study[10]. From the pre-study survey, we learned that the recruited participants were primarily students of University of Illinois at Urbana-Champaign (35 people). Plus, four participants were working professionals. Twelve students majored in Computer Science, three in Psychology, three in Biology, two in Mathematics, two in MBA, two in Nutrition, two in Agriculture, two in Mechanical Engineering, and we also had participants with the background in Civil Engineering, Political Science, Kinesiology, European Union Studies, Chemistry, Supply Chain Management, Accounting, Linguistics, Marketing, Medicine, and Education. 21 participants were female, and 18 were male. The age range varied from 22 to 34 years old, with the average age of 26.3 years old. There were 24 people pursuing or holding the Bachelor degree as the highest degree, ten people — the Masters degree, and five people — the PhD degree. More than 80% of participants searched for people online at least once a week, and more than 30% did it every day. All people were active Facebook (log in at least a few times per week) and LinkedIn (60% log in at least once a week) users (Figure 5.5).

5.2.2 Experimental Tasks

We picked four different people search tasks simulating a real need. The tasks are presented in Table 5.1. The participants needed to fill the gaps in the task templates based on their background and interests. We did it to further increase the relevance of the tasks to the participants and make them as realistic as possible. We asked the participants to choose different topics for each task to minimize the influence of the learning effects, i.e. so that the participants could not select the same set of candidates as they did for some previous task. According to our subjective post-task surveys (Table 5.3, Q8), the participants found tasks as realistic (4.06 on a 5-point Likert scale).

[10]while we only needed 12 + 24 = 36 participants, we had to recruit three more people to redo the user study sessions when we experienced technical difficulties with the data logging.

Task 1: Find a person to ask for career advice
Suppose you are looking for a new job as a ____________________. Before applying, you want to talk to people from different companies to learn about the culture, typical work-day/week, and existing projects. However, none of your friends work in the companies of your choice. Luckily for you, there is a new service that allows you to connect with the employees of a company and ask them all these questions. Select five people you would like to talk to and ask for career advice. You are also required to rank and evaluate the selected candidates after you finish the task.
Task 2: Find a keynote speaker for a conference
Suppose you are a part of the organizing committee for a conference on ____________________ and your task is to find a keynote speaker. You understand that people are busy and not everyone will agree/have time to accept your invitation. Your strategy is to find five candidates, message them and hope that one of them will accept the invitation. A search tool is provided to assist you in this task. You are also required to rank and evaluate the selected candidates after you finish the task.
Task 3: Help a recruiter find candidates to hire
Imagine that you are tasked to help a recruiter select a shortlist of ____________________ candidates for the interview. The recruiter doesn't know much about the space and needs your help. Please, find five candidates that should be invited for the interview by looking at their resumes. If a candidate is hired, s/he will be working in the same team as you. So, you should take into account your personal preferences while selecting the candidates. A search tool is provided to assist you in this task. You are also required to rank and evaluate the selected candidates after you finish the task.
Task 4: Find a collaborator for a project
Imagine that you are working on a ____________________ project and need help with ____________________. Luckily, there is a searchable resume database that you can use to find a person with the necessary expertise. You only need one person but want to talk to a few before making the final decision. Please, select five potential collaborators that you might invite to join your project by looking at their resumes. You are also required to evaluate the selected candidates after you finish the task.

Table 5.1: Four experimental tasks used in the study.

5.2.3 Experimental Procedure

At the very beginning of the study session, a participant was asked to sign the consent form and fill out a pre-study survey about their background. Then, the participant was given a tutorial about the experimental application and offered to perform a training task for ten minutes to get familiar with the user interface. Then, the participant was asked to work on four tasks using different versions of the experimental search system. As we mentioned above, in the first stage, we compared four versions of the SERP and, in the second stage, we compared the best two versions based on the results from the first stage. We allocated ten minutes per task, which is typically enough for the users to carefully explore the results [52, 145]. We asked the participant to "think-aloud" as s/he worked on the tasks. The experimenter took detailed notes for the subsequent analysis.

After each task, the participant was asked to complete a short post-task survey to capture the subjective satisfaction with the specific version of the application. The survey contained nine multiple choice questions (eight 5-point Likert scale [100] and one 5-point semantic differential). We also asked the partic-

Snippet Version	**Query-biased, 2 attribute lines**	**Non-redundant, 2 attribute lines**	**Query-biased, 4 attribute lines**	**Non-redundant, 4 attribute lines**
Rank	3.5 ± 0.7	2.8 ± 1.0	**2.2 ± 0.8**	**1.5 ± 0.8**

Table 5.2: The average rank given to each snippet version during the pilot study (lower is better).

ipant to evaluate the quality of selected search results on a 5-point Likert scale. Each result was judged independently, and hence, more than one result could receive the same relevance score.

All tasks were completed using our application with no interventions by the experimenter. The study was conducted in the same room to decrease the influence of external factors on study results. Since we interacted with the live search index, we grouped all user study sessions as close as possible timewise (12 days starting March 24th, 2016) to make sure that Indeed's index changes minimally affected the results.

At the end of each study, we conducted a 15-20 minute semi-structured interview by asking open-ended questions motivating the participant to express her/his thoughts about four (two during the formal study) versions of the SERP. The participant could see all four (two) versions and provide suggestions on how to improve the application. In total, one study session lasted approximately 90 minutes, and the participant was rewarded with the $15 cash gratuity.

5.2.4 System Instrumentation

To understand the way users interact with the search system, we logged all possible participants' actions, e.g. page views, clicks on the search results, additions/removals from favorites, page scrolls. For each event we additionally logged the following metadata: a task type, snippet type, timestamp, current query (combination of constraints on user attributes), and an anonymized userID. It allowed us to measure the system effectiveness and efficiency.

5.3 Pilot Study Results

We first present the key results from the pilot study and then dig deeper into the analysis of the quantitative and qualitative data from the formal study.

The main thing that we learned from 12 pilot study sessions is that the participants mostly noticed the difference in the snippet length and not in the snippet type and favored longer snippets more (11 out of 12 participants). Table 5.2 shows the average rank given to each of the four systems during the post-study interviews. Eight out of 12 participants placed NR4 in either the first or the second place; three placed QR4 in either the first or the second place. When scored on a scale of 1 to 4 (lower is better), the versions with the shorter snippets obtained the mean scores of 3.5 (QB2) and 2.8 (NR2), whereas the versions with the longer snippets obtained the mean scores of 2.2 (QB4) and 1.5 (NR4).

Surprisingly, we found that extra scrolling cost caused by longer snippets didn't have a negative impact on the participants' system choices. Many participants said that they either didn't notice it at all or preferred to have more information on the SERP to make more informed click decisions, like in [49, 151].

It is also worth noting that from this pilot study it seems that the participants preferred non-redundant snippets more than query-biased snippets given the fixed snippet length. However, this finding is not statistically significant. We will discuss it more in the next section within the scope of the formal study.

5.4 Formal Study Results

Based on the results of the pilot study, we limited the number of the snippet versions to two and only kept the longer ones, i.e. in the formal study we compared query-biased snippets spanning four lines (QB4) and non-redundant snippets spanning four lines (NR4). We compared different versions of the SERP using objective and subjective criteria. The objective evaluation was based on search user interaction logs and the subjective evaluation was based on user post-task survey responses and post-study interviews.

5.4.1 Subjective System Preference

At the end of each session we asked the participants *"Which version do you like most?"*. The majority of participants (19 out of 27)[11] preferred NR4. An exact binomial two-tailed test showed that the difference is significant ($p = 0.0357$).

Next, we probed the participants further and asked to explain the reasons behind their decision (*"Why do you like this version more? What makes it better than the other version?"*). In responding to this open-ended questions, the participants provided many rich and diverse opinions.

The ones, who preferred non-redundant snippets, said that it: (a) shows new non-redundant information (16 out of 19 participants); (b) helps discriminate the results on the SERP (12 out of 19 participants); (c) shows more relevant attributes (6 out of 19 participants); (d) helps accomplish the task faster (3 out of 19 participants); (e) requires less scrolling (3 out of 19 participants); (f) returns more relevant results (1 out of 19 participants). To add more color to the aggregated statistics and help the readers better relate to the participants' reasoning, we also provide verbatim anecdotal comments made by some of the participants:

[11]we decided to include post-study interview responses from 27 (and not 24) participants, including three extra people, since only the log data was corrupted. All participants of the formal study were able to successfully finish the tasks and make their opinion about different versions of the system.

"System 2 reduces repetition of information displayed on the screen. It can show more results per screen and I have to do less scrolling." **[P7]**

"It [System 2] is better since there is no extra line of information. I know they are all from Chicago and it is good that I don't have to see it here [on SERP]." **[P17]**

"System 2 shows less information but not loosing any information since it also shows search criteria compactly at the top." **[P22]**

The participants who preferred query-biased snippets paid more attention to different aspects of the SERP. They liked the version with the query-biased snippets more because it: (a) has a more regular layout (7 out of 8 participants); (b) shows more relevant attributes (6 out of 8 participants); (c) more predictable and reassuring (6 out of 8 participants); (d) demands less cognitive load and effort (5 out of 8 participants); (e) good balance of selected and new information (2 out of 8 participants); (f) works faster (2 out of 8 participants); (g) returns more relevant results (1 out of 8 participants); (h) forces to check individual profiles (1 out of 8 participants). Likewise, we provide several insightful anecdotal comments made by some of the participants:

"To be honest, I get distracted easily. If you show so much novel information, I don't know what to focus on. System 1 [query-biased] is more moderate in that sense. It either shows all info as in the query or only a few attributes are new." **[P6]**

"If I searched for a person with a doctorate degree, it will obviously have all results matching this filter. From that point of view, this information is redundant. But I still feel better psychologically when I see what I searched for. That's why I prefer System 1 more." **[P13]**

To summarize, almost all participants who preferred NR4 noted the fact that *non-redundant snippets are non-redundant.* On the other hand, rather than focusing on the informativeness of the user interface, the participants who favored QB4 paid more attention to the layout of the SERP and the effort required to do the search.

This key finding is also supported by numerous other subjective data that we collected during the study (Table 5.3). For example, in the post-task surveys the participants gave slightly higher scores to NR4 (Q6), but felt that QB4 was easier to use (Q1) and that the search process was less stressful (Q9).

It is important to note that the differences in the post-task survey responses mentioned above and some of the findings in the next section aren't statistically significant. That said, we don't consider it as the flaw in the analysis since the goal of this study is primarily to compare the systems/snippets from the users' perspective, and qualitative data is as important as quantitative data. Moreover, the combination of all

Metric \ Snippet Version	Query-biased, 4	Non-redundant, 4
I: Post-task Subjective Survey Responses		
Q1: The system is easy to use	**4.06 ± 0.59**	3.91 ± 0.79
Q2: The system provides me relevant candidates	3.98 ± 0.98	3.98 ± 0.86
Q3: The system helps me decide who to contact	3.92 ± 0.71	**4.13 ± 0.64**
Q4: The system helps me find relevant candidates efficiently	3.65 ± 0.93	**3.69 ± 0.88**
Q5: The display of each profile on the SERP is useful	3.96 ± 0.92	**4.04 ± 0.77**
Q6: Overall, I am satisfied with the system in this task	3.71 ± 0.68	**3.73 ± 0.68**
Q7: Summaries/attributes presented for each result on the SERP are useful	3.94 ± 0.93	**4.08 ± 0.82**
Q8: I can see myself doing this task in the real life	4.06 ± 0.81	4.06 ± 0.78
Q9: The search process is (stressful / relaxing)	**3.40 ± 0.87**	3.35 ± 0.93
II: Post-task Subjective Relevance Judgments		
Selected result relevance	4.00 ± 0.96	**4.07 ± 0.92**
First selected result	4.23 ± 0.81	4.23 ± 0.93
Second selected result relevance	4.02 ± 0.93	**4.04 ± 0.82**
Third selected result relevance	3.81 ± 0.87	**3.88 ± 1.10**
Fourth selected result relevance	3.83 ± 1.17	**3.90 ± 0.99**
Fifth selected result relevance	4.10 ± 0.97	**4.31 ± 0.69**
III: Query Log Data		
Query length	3.03 ± 0.97	**3.14± 0.95**
% of new queries	0.16 ± 0.17	**0.26 ± 0.19**
% of generalization queries	0.11 ± 0.11	0.11 ± 0.12
% of specialization queries	0.15 ± 0.15	0.15 ± 0.16
% of reformulation queries	**0.58 ± 0.33**	0.48 ± 0.27
# of queries per task session	**5.91 ± 5.60**	4.96 ± 3.90
Time between consecutive queries within a task session (seconds)	**63.80 ± 39.59**	56.77 ± 34.88
Time to complete a task (seconds)	**414.77 ± 122.10**	389.04 ± 118.30
Time to the first SERP click after submitting a query (seconds)	11.36 ± 4.71	**13.17 ± 6.96**
Mean SERP click position per task session	**6.48 ± 3.97**	6.18 ± 4.77
Maximum SERP click position per task session	**28.13 ± 21.14**	24.39 ± 19.78
# of SERP clicks (profile views) per task session***	**18.51 ± 5.80**	16.00 ± 4.60
# of profiles *added* to favorites per task session (*SERP*)	1.00 ± 2.13	**1.44 ± 2.77**
# of profiles *removed* from favorites per task session (*SERP*)	0.43 ± 1.11	**0.65 ± 1.40**
# of profiles *added* to favorites per task session (*detailed profile page*)	**4.85 ± 2.11**	4.48 ± 2.03
# of profiles *removed* from favorites per task session (*detailed profile page*)	**0.45 ± 0.78**	0.39 ± 0.74
IV: Post-study Subjective Preference (Interview)		
Percentage of participants favoring a specific version (out of 27)***	29.6% (8/27)	**70.4% (19/27)**

Table 5.3: A comparative table with the metrics collected for two experimental versions during the formal study (query-biased snippets with four attribute lines per result and non-redundant snippets with four attribute lines per result). *** is used to mark stat. significant differences (a two-tailed test at $p < 0.05$).

findings supports the thesis that the version with non-redundant snippets is more effective and efficient since all of our findings are consistent with each other.

5.4.2 System Effectiveness

Rather than using predefined topics typical for information retrieval system evaluation [72], in this study we used templatized tasks to closer simulate real life situations and motivate participants to engage more in exploratory search, where the objective is usually subtle and might evolve during the search session [83]. Therefore, only the participants themselves could evaluate the quality of selected candidates. To this end, after each task we asked the participants to provide independent relevance judgments to the selected candidates on a scale from 1 to 5 (5 is the highest)[12]. Then, we computed the average relevance score over the five selected candidates. We also computed the average relevance score for candidates selected at a specific position in the session, i.e. if some candidate was the first discovered candidate among the five candidates selected during the session, then his/her relevance score will be aggregated under the *First* group. The results are presented in a global comparative Table 5.3.

It seems that NR4 helps find more relevant candidates than QB4 (Table 5.3, Part II). The data also suggests that the candidates discovered at a specific position in the session have the higher relevance in the case of NR4. Remarkably, the candidates selected the last in the session on average had higher ratings compared to the candidates selected in the middle of the session. From the "think-aloud" comments we reconstructed that the first few candidates were typically selected from the result sets for the first most accurate query and, hence, had high relevance scores; then, the relevance started to decay as the participants scrolled down the SERP and "starred" a few less relevant results; eventually, the participants reformulated their query several more times and picked a few relevant results from the last SERP being engaged in exploratory search.

The post-task survey responses also point towards the fact that NR4 helps find more relevant results: the participants rated NR4 as more helpful to decide who to contact (Q3); felt that the information on the SERP was more useful (Q5 and Q7); and, overall, NR4 was rated slightly higher (Q6).

5.4.3 System Efficiency

We used query log data and subjective post-task responses to compare the efficiency of different systems. The primary measure to assess the efficiency of different versions is time. We computed the average time

[12]This method of using participants' generated labels to simulate ground-truth data has also been adopted in several other related works [52, 109, 151].

required for each participant to finish a task and found that NR4 led to faster task completion times in seconds ($\mu = 389.04$; $\sigma = 118.30$) than QB4 ($\mu = 414.77$; $\sigma = 122.10$).

Then, we dug deeper to understand why the participants were able to complete the tasks faster with NR4. We discovered that the average time between consecutive queries within a session was smaller for NR4 ($\mu = 56.77$; $\sigma = 34.88$) compared to QB4 ($\mu = 63.80$; $\sigma = 39.59$) and that the average number of queries per session was also smaller for NR4 ($\mu = 4.96$; $\sigma = 3.90$) compared to QB4 ($\mu = 5.91$; $\sigma = 5.60$). We went one level deeper to understand why the time between queries was shorter and why did people submit fewer queries when using NR4.

We found that the participants made less clicks on the SERP (visited fewer number of detailed profiles) in the case of NR4 ($\mu = 16.00$; $\sigma = 4.60$) compared to QB4 ($\mu = 18.51$; $\sigma = 5.80$). This difference was statistically significant according to non-parametric two-tailed Mann-Whitney test ($p = 0.0368$)[13]. On the contrary, time to the first SERP click was longer for NR4 ($\mu = 13.17$; $\sigma = 6.96$) compared to QB4 ($\mu = 11.36$; $\sigma = 4.71$), which suggests that the participants engaged with the SERP more and did more informed decisions by looking at the snippets. Another possible explanation could be that they felt overwhelmed with the amount of novel information presented on the SERP. However, this is an isolated hypothesis. There are several other findings supporting the hypothesis that users engaged with the SERP more in the case of NR4. While not statistically significant, participants added ($\mu = 1.44$; $\sigma = 2.77$) and removed ($\mu = 0.65$; $\sigma = 1.40$) from the Favorites on the SERP more often when using NR4 than they added ($\mu = 1.00$; $\sigma = 2.13$) and removed ($\mu = 0.43$; $\sigma = 1.11$) when using QB4. In the post-task surveys the utility of NR4 was also rated higher (Q5, Q7).

Interestingly, we found that people specify more constraints ($\mu = 3.14$; $\sigma = 0.95$) using NR4 compared to QB4 ($\mu = 3.03$; $\sigma = 0.97$). A possible reason came from "think-aloud" comments made by two participants – both of them noted that they tried to limit the number of constraints to three when using QB4 to make sure that there was at least one attribute to differentiate the results on the SERP. Such problem doesn't exist for NR4. The verbatim "think-aloud" comment and the follow-up explanation during the interview are presented below:

"...since all results would be the same there [on the SERP], I would go with three conditions..."

[P9, think-aloud]

"System 1 shows what I selected, and there is no way I could differentiate one candidate from another if I specify more than four conditions. Because of that I always submitted less than four.

[13]we use the non-parametric Mann-Whitney test because of the small sample size and uncertainty that the data is normally distributed, which make parametric tests not valid.

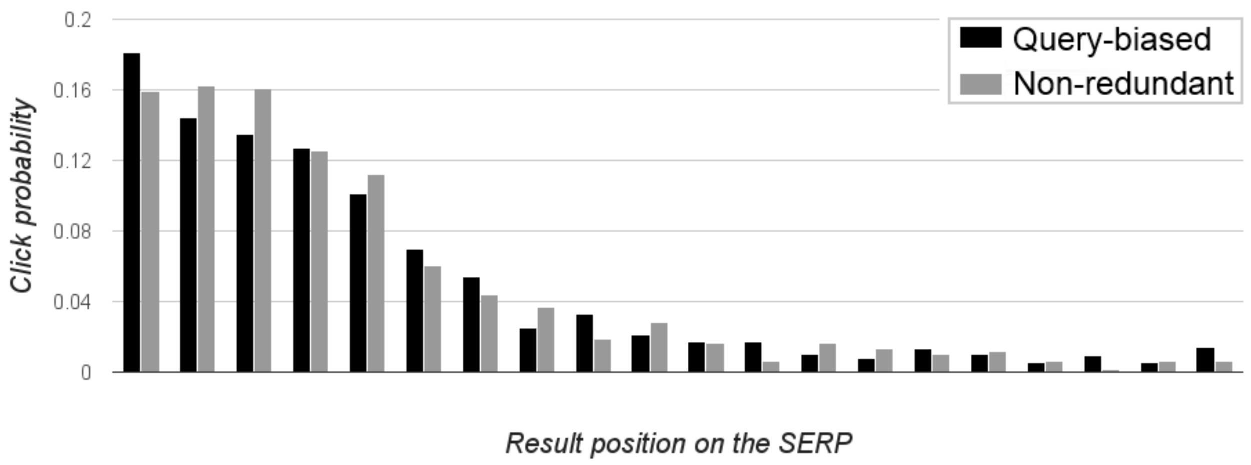


Figure 5.6: The percentage of clicks on search results broken down by the result position on the page for two different versions of the snippets.

System 2 is better..." [**P9, interview**]

Similarly to [52], we analyzed four query reformulation patterns: new, generalization specialization, and reconstruction. The corresponding definitions are provided below:

- **New:** Q_i is the first query or does not share any common terms with Q_{i-1}.
- **Generalization:** Q_i shares common terms with Q_{i-1} but Q_i contains fewer terms.
- **Specialization:** Q_i share common terms with Q_{i-1}, but Q_i contains more terms.
- **Reformulation:** Q_i shares common terms with Q_{i-1}, but Q_i has at least one different term with Q_{i-1}.

For each pattern, we computed the percentage of this pattern out of all used patterns. We found that the participants submitted more new queries when using NR4 ($\mu = 0.26$; $\sigma = 0.19$) compared to QB4 ($\mu = 0.16$; $\sigma = 0.17$), but reformulated their queries less with NR4 ($\mu = 0.48$; $\sigma = 0.27$) compared to QB4 ($\mu = 0.58$; $\sigma = 0.33$). It might be the case since with NR4 the participants explored the neighbourhood of the initial query faster by looking at the snippets, while with QB4 they had to change the query to see novel information.

Finally, we calculated the mean average and maximum click position on the SERP and found that both metrics were lower in the case of non-redundant snippets. We further explored this by drawing the histogram for click positions, which is shown in Figure 5.6.

As we can see, for both snippet types, the participants primarily examined the SERP from top-to-bottom, which is known as a *position bias* effect [66]. However, in the case of query-biased snippets the graph monotonically decays while for non-redundant snippets the first three positions (one screen) have the same percentage of clicks. This suggests that in the case of query-biased snippets the participants relied more on the ranking algorithm while for the non-redundant interface the snippets played a major role, which is also consistent with the finding that time to the first SERP click is larger for NR4.

To conclude, the combination of all findings provides compelling evidence that the version with non-redundant snippets makes users more effective and efficient.

5.4.4 Search User Interaction Strategies

In this section, we present query formulation, result examination, and result selection strategies revealed during the post-study interviews. The strategies provide additional context and help reinterpret some of the findings presented in the previous sections. For each strategy, we show the count denoting the number of participants using this strategy.

Query Formulation Strategies

The participants followed four different query formulation strategies. The most popular strategy was to submit a query that strictly describes an ideal candidate by using many query constraints, then reformulate or generalize it if is too specific (18 participants). To evaluate the specificity of the query, the participants paid attention to the number of search results. They considered the queries with less than 100 results as manageable and added more constraints in the case of 100+ results. On the contrary, some participants preferred to submit a query only with minimal requirements, then try several more specific queries by modifying optional attributes (6 participants). Likewise, some participants submitted a very board query as the first query in the session, then added many results to favorites, and finally picked the top-5 from the Favorites page (4 participants). In this case, the participants used the SERP rather than the detailed user profile page to add people to favorites. Finally, the last but not the least was a strategy to create a search "design space", then methodically check all possible combinations (6 participants).

The participants tended to change the query if they saw that the quality of results was decreasing with the rank on the SERP. Plus, half of the participants wanted to select several values per attribute (e.g. *"Lyft or Uber"*) to cover a broader set of options with one query.

Result Examination Strategies

When browsing the SERP, several participants explicitly mentioned that they looked at the names and provided two reasons for doing so. First, the participants remarked that the names were in bold, and hence, they attracted more attention as being more visually salient. Second, some participants purposefully looked for candidates with the names characteristic of a specific demographic group arguing that such candidates might be more approachable and willing to help them as being members of the same community. This serves as the demonstration of homophily in social interactions [88].

From the “think-aloud” comments, we also learned that the participants looked at the first and the last lines of the structured snippet, i.e. it seems that these areas of the snippet attracted more users’ attention. Although an eye-tracking study is necessary to investigate it more accurately. The participants aligned one result to the next one to see the difference in attribute values shown and decide what candidate profile to click. Interestingly, one participant invented a very unusual pattern for adding candidates to favorites. She first checked the detailed profile page, then went back to the SERP, and finally made the decision to favorite there arguing that she *“could compare it with the candidates above and below in on the SERP”* and that it *“gives me [her] one more second to think whether the candidate is good”*.

The participants remembered the query if it had 1-2 constraints, but found the “breadcrumbs” area [54] as useful for cases with 3+ query constraints.

As noted above, all participants explored the SERP from top-to-bottom confirming the *position bias* effect [66]. When a participant deviated from this strategy, s/he said that *“The order of examination is actually random, not top-to-bottom. Sometimes I just scroll and see if some word catches my eye.”*.

Result Selection / Bookmarking Strategies

People form social relationships, and therefore, ranking people is qualitatively more complex than ranking textual documents. As a result, [129] proposed the concept of social matching to emphasize the social dimension. The participants demonstrated the importance of the social dimension in our study. They tried to assess the social similarity by looking for distinctive names, alumni from the same university, people from the same age group and with the same level of qualifications. The argument was that *“such people will be more willing to help because they have a lot in common with me”* or that *“the interaction with them would be more effective”*. All things equal, the participants tended to select candidates, which had some familiar attribute values in their profiles (e.g. big companies, well-known universities).

Almost all participants added their best choices to favorites from the detailed page explaining that they *“don’t have enough information on the SERP to decide whether the candidate is good”*. Some participants also referred to fairness by saying that *“she wanted to give equal chances to all candidates rather than superficially picking the ones from famous places”*. When people added to favorites from the SERP, they did it: (a) because there were too many results and they wanted to shrink the search space; (b) to compare with the adjacent results on the SERP as we described in the previous section (can do it with non-redundant snippets); (c) intentionally wanted to select only the candidates that *"have something remarkable to sell themselves even from the SERP"*. The participants skipped the candidates with "N/A", "Self-employed", "Intern" in the job title.

The participants also mentioned that they wanted to be able to specify the attributes shown on the SERP and even change the resume layout depending on the task by reordering the sections. For example, for the recruiting task one might show the education at the top while for a conference speaker show the work experience at the top.

5.5 Design Implications

In this section, we discuss what our findings suggest for the design of PSN SUIs with the particular focus on mobile devices.

5.5.1 Eliminate Search User Interface Redundancy

Our key finding is that users do notice redundancy in the search user interface and prefer non-redundant snippets more. At the same time, non-redundant snippets cannot be used when users specify multiple constraints per attribute (e.g. *"Uber or Lyft or YellowCabNYCTaxi"*) or when an attribute has an array type (e.g. skills *"Java, Scala, Hadoop, Spark"*) because there will be variability in results' attribute values. Therefore, we suggest using non-redundant snippets when the query formulation interface doesn't allow to specify multiple constraints per attribute (e.g. Zappos mobile app) or when only one value per attribute/facet is selected. Query-biased snippets should be used when the query implies that the attribute values might vary from one result to another (in this case, query-biased snippets are at the same time non-redundant snippets since for different results they theoretically might be different).

That said, we warn the reader that the design recommendation to eliminate redundancy is somewhat opposite to the established SUI design guidelines [54, 112]. We speculate that it is due to the fact that in our study we focused on structured data, which might be associated with the specific information seeking strategies. More user studies and A/B tests are necessary to confirm that non-redundant snippets are more useful. Only by combining and triangulating the findings from many studies, we could definitively find that SUI redundancy must be eliminated in PSNs this way. In that sense, the study described in this Chapter 5 serves as the first step towards more usable and effective snippets for structured PSN search on mobile devices. We also note that in our study the participants were more effective and efficient with the non-redundant interface, which is opposite to the known fact that redundancy is beneficial for multi-modal interfaces [75].

We think that the finding is different in our study because of the different context. Redundancy in the user interface is useful for critical real-time domains (e.g. airplane cockpit, medical telemetry) when there is a lot of incoming information and the cost or error is very high. However, redundancy might not be required in less stressful domains, like search. An in-situ study should reveal additional insights on the way people search when under pressure.

Furthermore, we recommend experimenting with various other ways to eliminate clutter from the SERP. For example, we learned that users look at the query constraints at the top of the screen to confirm that the query is correct mostly when they revisit the SERP the first time after submitting a new query. Therefore, the SUI can be simplified with the proper interaction techniques by hiding the "breadcrumbs" area at the top of the screen as users scroll down or showing it only when the users pull the top area of the SERP down. Based on the fact that users remember the query when they specify less than three constraints, the "breadcrumbs" area might be eliminated completely. The system could show the "breadcrumbs" only for "hard" queries when users use more than three constraints. In other words, redundancy in the snippets is not necessarily required; query redundancy might be enough to provide informative feedback [112].

We also discovered that users mostly add people to favorites from the detailed profile page, and hence, the stars that occupy a significant part of the SERP space can be eliminated, too. We suggest allowing users to (un)bookmark candidates by swiping to the left/right rather than by clicking on stars. Since the swipe is purely an interaction pattern, no space on SERP is required to provide this functionality.

The freed space might be used to show some additional information to help users make more informed click decisions. From the interviews, we learned that in the case of people search the searchers not only try to assess candidates based on their relevance to the query/task but also use other factors, such as social similarity, status, approachability, and seniority, demonstrating the presence of homophily [88]. Therefore, to simplify this process for users, search engines could explicitly show on the SERP the similarity score between a searcher and a candidate result or a simple data visualization with the scores along several dimensions [52]. Furthermore, these scores can be made more interpretable by showing explanatory comments next to each number, e.g. *"87% (both you and John Smith went to MIT)"* or *"95% (Alice Smith is in top 5% among all people using Python)"*. While we share this idea in the design implications section, in the current form it should be considered more like a hypothesis. This seemingly simple idea to add scores to the snippets is actually very complex and raises many questions: What if the scores will bias searchers' click patterns? Will the presentation of scores cause discrimination? Will searchers pay attention to the scores at all? One can evaluate this idea by adding the scores and explanations to the SUI and running an interactive user study or an online A/B test.

From the pilot study, we learned that users don't notice the scrolling cost and prefer having more information about each result on the SERP. Therefore, we suggest using non-redundant long snippets. At the same time, we warn the readers that there must be at least several results per page to enable comparison as several of our participants explicitly mentioned that they use the results above and below the current result to decide whether to add her/him to favorites. Additional user studies are necessary to find the optimal snippet length on mobile devices. Interestingly, some popular applications do show one result per page, e.g. LinkedIn Studnets[14] and Tinder[15].

5.5.2 Provide More Control

Many participants mentioned that they would love to have more control over the search process, e.g. the ability to specify what attributes to show on the SERP or set attributes' importance weights to manipulate results ranking, like in [52]. Plus, the participants mentioned that despite having a good ranking algorithm, they would love to have more transparency in ranking and be able to resort the attributes based on well-defined criteria, e.g. years of work experience, age. Therefore, we suggest exposing the seams and provide more control, which might be especially beneficial for advanced users. Interestingly, some of the participants even wanted to reorder the sections of the resume depending on the task. The algorithmic transparency might lead to many positive outcomes for the users [42] and establish a higher level of trust between the users and the search engines.

From the study, we also learned that some users prefer having input confirmation while others seek more information per pixel. We think that a balanced solution might be designed. One idea is to mix non-redundant and query-biased snippets by always showing one query-biased attribute and filling in the rest of the slots with non-redundant information. Alternatively, this decision might be offset to the users such that they could select the snippet type in the search settings. Finally, one can predict the user type algorithmically based on search interaction data and show the snippets, which are more appropriate for them, e.g. elderly users might benefit from a less dynamic interface with query-biased snippets, while the younger users actively using digital products might see a more information-rich interface.

5.5.3 Direct Users to Use Better Search Strategies

We also noticed that in our study the participants on average specified more than three query constraints, while in Chapter 2 we found that structured grammar queries submitted to Facebook Graph Search typically have less than two constraints [117]. This might be due to our query formulation interface and, hence, we

[14]`https://students.linkedin.com/`
[15]`https://www.gotinder.com/`

recommend experimenting with various ways to steer users to write longer queries since it is known that longer queries tend to provide better results. For example, in [11] a method is proposed to motivate users to write longer keyword queries by providing a longer query bar.

Similarly, we discovered that the last selected candidate in the session, which is returned as part of the result set for the last query, has a higher rank than the candidates selected in the middle of the session, which are typically elements of the result set for the first query. Therefore, it might be beneficial to explore new mechanisms that encourage users to reformulate their queries. Structured query suggestions innovated by Facebook Graph Search [117] represent one such example. We envision many new techniques based on the unique features of mobile devices such as gyro, touch screen, or even eye-tracking.

5.6 Limitations and Future Work

Despite our efforts to design a rigorous study, it still has several limitations. In this section, we discuss the known limitations and ideas for future work. First, we focused on *People* search, and that's why our findings might be limited only to this search vertical. In the future, we will examine other structured search verticals within PSNs (e.g. *Jobs*) and beyond PSNs (e.g. *Travel* or *E-Commerce*) to investigate whether there are some vertical-specific differences. Yet, we argue that our study in its present form is useful since the SUI is quite the same across many search verticals. Second, the majority of our participants were students. Therefore, they might search differently than working professionals and be less familiar with the professional people search. We plan to do the follow-up study and investigate whether there is some interaction between the snippet type and the level of professional experience. Third, because the study was conducted in a quiet lab, the participants didn't have any interruptions. In the real world, users multi-task and are overloaded with information. This might affect the way they search. Therefore, we think that an ethnographic study or an online A/B test might be useful to further understand how external context influences the way people interact with the SUI. Fourth, we used subjective judgments to evaluate system effectiveness since there were no ground-truth labels. Alternatively, we could have designed known-item search tasks and have objective criteria for result relevance. However, it might have reduced the relevance of the tasks to the participants and limited exploration. An A/B test might be the most appropriate method to further understand which version of the snippet leads to better search outcomes.

We also acknowledge the fact that it might be hard to adopt non-redundant delta snippets in practice since the objects within PSNs need to be standardized first. At the moment, search ranking and matching are probabilistic and not exact because different objects have slight variations in attribute values. For example, it is not clear whether both people working as “Software Engineer” and “Senior Software Engineer” should appear as search results for the query “Software Engineer”. Plus, the adoption of non-redundant delta snippets might be inhibited because users have learned search habits and could object this change. Finally, it is worth noting that query-biased snippets present to the users information that they expect to see, while non-redundant delta snippets show novel and unexpected information. And, it is known that mismatch of expectations and reality leads to anxiety and stress. Therefore, search engine designers must know their users and their expectations before adopting non-redundant delta snippets.

5.7 Conclusions

In this Chapter 5, we described the comparative study aimed to understand which version of the snippet is better suited for structured search on mobile devices. We found that the participants preferred non-redundant snippets more. The multitude of our qualitative and quantitative data also indicate that the system with non-redundant snippets was more effective and efficient. Non-redundant snippets led to faster task completion times, helped find more relevant results, and made the participants do more informed SERP click decisions. The participants engaged with the SERP more when using the system with non-redundant snippets. On the other hand, the participants who favored query-biased snippets paid more attention to the layout and visual aspects of the SUI and felt that the system with query-biased snippets was easier to use. We also learned that the participants favored longer snippets without respect to the snippet type. Our findings serve as the first real evidence discovered via an interactive user-centric study that structured search engines should eliminate redundancy from the SUI.

Chapter 6

Conclusion

In this Chapter 6, we review our key contributions, share SUI design guidelines, and propose ideas for future work coming from the experience working on the projects described in the previous chapters.

6.1 Thesis Summary

In this thesis, we rethought, redesigned, and optimized search user interfaces and interactions within professional social networks. We considered all aspects of the search user interface starting from the query formulation, to control, and to results presentation. By reasoning from basic principles and mining large scale user behavior and interaction data, we identified areas for improvement and designed novel domain-specific solutions. We rigorously evaluated each of the proposed new ideas offline and online by using synthetic and real datasets and by combining various user-centric research methods, such as surveys, interactive lab user studies, interviews, query log analysis, and large scale online A/B tests.

Following the searcher journey, this thesis started from the query formulation. With the Facebook Graph Search query log analysis study, we investigated people search patterns and showed how more cognitively demanding and highly interactive structured query language enables new search behavior within Facebook. Going forward, we considered the control aspect of the search user interface and proposed to do relevance-aware search results filtering when sorting by an attribute value rather than by relevance is requested. The significant number of queries to structured search engines, including the ones inside PSNs, fit this scenario. We proposed a novel theoretically optimal algorithm to perform such filtering and evaluated it with the series of experiments on real and synthetic data sets. Finally, we moved on to the presentation of the search results and focused on various new ways of snippet generation. We challenged the common belief that query-biased snippets, which are used almost everywhere these days, are the most effective and proposed domain-specific alternatives. Specifically, based on the results from the mixed-method user need elicitation study, we proposed the concept of the *extended informative structured snippet* for job search that shows the responsibilities and the requirements for a job right on the SERP. We designed the effective and efficient weakly-supervised algorithm that can extract relevant information from unstructured job postings to make the generation of such a snippet possible. We tested the utility of this new snippet type with the online A/B experiment involving over half a million real users. In another project, we compared the utility of query-biased and non-redundant delta snippets for structured search on mobile devices by conducting the comprehensive task-oriented user study. Based on the results of this study, we concluded that structured search engines should eliminate redundancy from the user interface and shared numerous other design recommendations relevant for the design of the future SUIs and interaction techniques within PSNs. The combination of the proposed ideas demonstrates how we can optimize the query-control-view interaction loop and make users more effective while searching for information within PSNs.

It is important to note that the ideas presented in this thesis have broader applicability beyond the scope of PSNs. For example, the recommendations derived from the study of structured querying capabilities within Facebook might be applicable for many other search engines built on top of structured networked data. We believe that like on Facebook users will be able to more effectively explore any entity graph, which is meaningful for them, if they are provided with a powerful structured query language and highly interactive query suggestions. Graph databases (e.g. Neo4j[1]) democratize the development of search engines providing such affordances. Likewise, the algorithm for relevance aware search results filtering might be useful for any structured search engine, where tuples have numeric or even just ordinal attributes. When users request sorting by an attribute value, we can perform the filtering and increase the cumulative value delivered in the result list. Finally, despite the limitations described in Section 5.6, the comparative study of query-biased and non-redundant snippets provides evidence that a wide variety of structured search engines might increase search utility by eliminating SUI redundancy.

6.2 Search User Interface Design Guidelines for Professional Social Networks

In this section, we expand on the search user interface design guidelines proposed in previous work [54, 112] and share new design guidelines based on the studies described in this thesis.

- **Provide both named entity queries (NEQs) and structured queries (SQs).** From the FBGS query log analysis study (Chapter 2), we learned that both NEQs and SQs are important to facilitate navigation and exploration within the social network: users search for friends with NEQs and for non-friends and explore the graph using SQs. Therefore, an interactive Typeahead interface supporting

[1]`http://neo4j.com/`

both NEQs and SQs facilitates navigation and exploration and makes information stored within the online social network useful and easy to search. Having these two query types tailored to a specific class of information needs within one system is beneficial for users of an online social network as these queries don't cannibalize but complement each other.

- **Personalize search user experience and focus on structured query suggestions.** As part of the FBGS query log analysis study (Chapter 2), we noticed significant changes in search behavior and interaction data for users with different demographics. We found that the number of *Friend* queries grows as users gain more friends, while the number of *Non-friend* queries slightly declines, that celebrity users search for celebrities more than typical users, that females and users older than 60 are more interested in the first degree connections compared to the rest of the users in our sample, and several others. Therefore, we suggest to further innovate around personalized search query suggestions given our demographics' distinct people search patterns. Plus, since structured query usage behavior has a wider variation across different demographics, it makes sense to focus efforts on that query type.

- **Account for graph search distance between a searcher and an object to be searched.** Since entities within a PSN are interconnected and form an Entity Graph, it is important to take this information into account. As part of the FBGS query log analysis study (Chapter 2), we found that users repeatedly search for non-friends without adding them to their friend network. Therefore, we suggest including some interesting distant network vertices rather than limiting query suggestions to friends only. We also found that the distribution over graph distances varies from predicate to predicate. While some predicates are used primarily to explore information about friends, other predicates are used for non-friends. We propose ranking entities for a predicate using its graph distance distribution. We discovered that users write shorter queries when they search for friends and use more predicates to find non-friends. Therefore, we propose to generate interactive query suggestions by predicting an intended graph search distance. In the past, the interlinked structure of the web graph turned out to be very important [18]. We believe that it might be even more important in the context of OSNs.

- **Incorporate relevance into the ranking process when sorting by an attribute value is requested.** From the simulations described in Chapter 3, we learned that the quality of search results sorted by an attribute value could be improved using relevance-aware filtering by 2-4% with the help of the relevance-aware search results filtering algorithm. Plus, we noticed that higher gains are characteristic for the relevance label distributions, where relevant results are more probable, and for medium length result sets (20-100 tuples).

- **Present information about job responsibilities, requirements, and conditions right on the SERP.** From the user need elicitation study described in Chapter 4, we learned that users pay attention to the job responsibilities and the job requirements when deciding whether they want to apply/click on a job. Moreover, through the A/B test we learned that by showing this information right on the SERP we could minimize irrelevant clicks, increase search effectiveness and efficiency, and decrease SERP click entropy.

- **Eliminate query-snippet redundancy.** From the comparative study described in Chapter 5, we learned that users do notice redundancy in the SUI and prefer non-redundant snippets more. Therefore, we suggest using non-redundant snippets when the query formulation interface doesn't allow to specify multiple constraints per attribute (e.g. Zappos mobile app) or when only one value per attribute/facet is selected. Query-biased snippets should be used when: (a) the query implies that the attribute values might vary from one result to another (in this case, query-biased snippets are at the same time non-redundant snippets since for different results they theoretically might be different); (b) users specify multiple constraints per attribute (e.g. *"Uber or Lyft or YellowCabNYCTaxi"*); (c) an attribute has an array type (e.g. skills *"Java, Scala, Hadoop, Spark"*) because there will be variability in results' attribute values. That said, we warn the reader that it is necessary to run an A/B test and confirm that non-redundant snippets are more useful for the users of a specific search system prior to switching. Only by combining the findings from the interactive user study described in Chapter 5 and from the A/B test, we could definitively state that redundancy should be eliminated in PSNs this way.

- **Help users differentiate one result from another.** From the comparative study described in Chapter 5, we learned that one of the key reasons for users to prefer non-redundant snippets over query-biased snippets was that with non-redundant snippets they can differentiate results easily and hence be more effective at search. The approaches described in [29, 30, 79, 91] are especially useful to accomplish this.

- **Eliminate unnecessary input confirmation messages.** From the comparative study described in Chapter 5, we learned that users remember the query when they specify less than three constraints and look at the query constraints at the top of the screen to confirm that the query is correct mostly when they revisit the SERP the first time after submitting a new query. Therefore, the SUI can be simplified with the proper interaction techniques by hiding the "breadcrumbs" area at the top of the screen as users scroll down or showing it only when the users pull the top area of the SERP down. A less aggressive solution is to show the "breadcrumbs" only for "hard" queries when users use more than three constraints. In other words, redundancy in the snippets is not necessarily required; query redundancy might be enough to provide informative feedback [112].

- **Show "call-to-action" elements only where they are necessary.** From the comparative study described in Chapter 5, we discovered that users mostly add people to favorites from the detailed profile page and, hence, the "Connect" or "Favorite" buttons that occupy a significant part of the SERP space can be eliminated from SERP. However, we warn the reader that these findings are specific to the search scenario employed in our study and with the selected group of participants. A large scale A/B test is necessary to accurately answer this question and measure user engagement with the "call-to-action" elements in an unbiased way.

- **Provide more and novel information about each result on the SERP.** From the comparative study described in Chapter 5, we learned that users don't notice the scrolling cost and prefer having more information about each result on the SERP. Therefore, we suggest using non-redundant long snippets. At the same time, we warn the reader that there must be at least several results per page to enable comparison as several of our participants explicitly mentioned that they use the results above and below the current result to decide whether to add her/him to favorites.

- **Provide more control.** During the comparative study described in Chapter 5, many participants mentioned that they would love to have the ability to specify what attributes to show on the SERP or set attributes' importance weights to manipulate result ranking, like in [52]. Plus, the participants mentioned that despite having a good ranking algorithm, they would love to have more transparency in ranking and be able to resort the attributes based on well-defined criteria, e.g. years of work experience, age. Therefore, we suggest exposing the seams and provide more control, which might be especially beneficial for advanced users.

- **Direct users to use better search strategies.** During the comparative study described in Chapter 5, we discovered that the last selected candidate in the search session, which is returned as part of the result set for the last query, has a higher relevance score than the candidates selected in the middle of the session, which are typically elements of the result set for the first query. Therefore, it might be beneficial to explore new mechanisms that encourage users to reformulate their queries. Structured query suggestions innovated by Facebook Graph Search [117] represent one such example. We envision many new techniques based on the unique features of mobile devices such as gyro, touch screen, or even eye-tracking. We also suggest experimenting with various ways to steer users to write longer queries since it is known that longer queries tend to provide better results [11].

6.3 Scenario Revised: Search Within Professional Social Networks Tomorrow

In this section, we reconsider Alices scenario from Chapter 1 in lieu of the innovations proposed in this thesis. Alice starts her job search for a software engineering job in New York by going to the *"Jobs"* tab on LinkedIn. She types *"Software Engineer"* to the profession input box and *"New York"* to the location input box and hits *"Search"*. The job search engine returns a list of relevant jobs. Each job is represented by the job title, which is still the same for all retrieved jobs, and a series of attributes that help Alice differentiate one result from another. Since Alice already knows that all jobs will be in "New York" area, this information is not displayed on the SERP for each result but instead it is shown in the "breadcrumbs" area (Chapter 5). It saves the scarce SERP space and allows to show some other relevant information, e.g. instead of the location attribute on the SERP Alice could see a salary range as it was shown to be important to help users make job clicks/apply decisions. Moreover, Alice could also see the responsibilities and the requirements for each job right on the SERP (Chapter 4). It helps Alice make more informed click decisions and saves her time searching since she could quickly evaluate job relevance without visiting individual job posting pages.

Next, Alice decides to resort jobs by date. She changes the corresponding selector. The job search engine resorts the results by date and eliminates from the top of the resorted list the jobs with very low relevance scores by performing relevance-aware filtering (Chapter 3). It again saves Alice's effort and makes her search process more efficient since she doesn't have to scroll to see relevant results — the most relevant for her results are displayed right at the top of the SERP.

Next, Alice tries to find friends, who work as software engineers in the companies she selected. For that, she starts typing her structured query *"Software Engineers who... "* in a smart input box and the people search Typeahead algorithm shows several relevant query suggestions personalized for Alice's current search session, e.g. *"Software Engineers who live in New York"* (because she recently search for jobs in New York) or *"Friends of my friends who are Software Engineers"* (because it is known that the majority of structured queries related to professional search are about non-friends). Alice selects the most relevant typeahead query suggestion *"Friends of my friends who are Software Engineers"* and quickly hits *"Enter"* without switching to a mouse. Alice sees a list of relevant people, however, not all of them live in New York. Therefore, she quickly appends *"...and live in New York and work at Google"* to her query and hits *"Enter"* again. The people search engine returns a list of relevant people, who are *"Friends of my friends who are Software Engineers and live in New York and work at Google"*. In other words, Alice interactively and searches within the professional social network using a powerful query interface described in Chapter 2.

Satisfied with the query, Alice starts skimming the SERP. The SERP is composed in such a way that the shared and known to Alice attribute values (location — *"New York"*, job title — *"Software Engineer"*, degree — *"Friends of friends"*) are shown as "breadcrumbs" at the top of the SERP to avoid redundancy. Instead of these attribute values in the snippets, Alice could see skills, shared friends, and hobbies, which are unique for each search result. In other words, the snippets are non-redundant and show information complementary to the query (Chapter 5). Alice learns a lot about each person just by looking at the SERP, selects a few similar to her people, and contacts them. These people kindly reply to Alice knowing that they have strong shared connections.

Finally, based on advice she receives from these people, Alice successfully applies for several software engineering jobs in New York.

6.4 Future Research Agenda: Graph Search and Beyond

This thesis contributed multiple novel ideas on how to optimize search user interfaces and interactions within professional social networks to make users more effective and efficient. However, this is just the first step in this direction. Below we present many exciting directions for future work.

6.4.1 Search User Interface Personalization

Search personalization leads to significant gains in search relevance and improves search user experience [13, 86, 127]. Because of that, different users submitting the same query to the same search engine might get different results. However, to the best of our knowledge, none of the search engines adapt their SUIs for different users. They merely provide *"Advanced Search"* forms that allow formulating more precise queries. We see it is as a great opportunity for research.

For example, based on our user study presented in Chapter 5, we concluded that non-redundant snippets are better for structured search on mobile devices and put it as one of our design recommendations. At the same time, we do acknowledge the fact that not all users might feel comfortable seeing non-redundant snippets. Elderly users, who typically have less experience with digital products and are subject to age-related cognitive decline [67, 105], might still prefer to see query-biased snippets. With query-biased snippets rather than trying to recall the query, users only engage in the recognition process, which is known to be less cognitively demanding [41]. Therefore, similar to the way we personalize search ranking now, we can personalize the snippet type and the SUI in the future.

Likewise, it is reported that adding information to the contextual snippet significantly improves performance for informational tasks but degrades performance for navigational tasks [28]. Therefore, in addition to the user-based SUI personalization, we think that task-based SUI personalization might be valuable.

It is worth mentioning that some initial steps have already been made in this direction. For example, recently [161] proposed an Interface Card Model by framing the task of an interactive retrieval system as a game, where a search engine and a user cooperate to satisfy the information need of the user and minimize user's efforts. It was shown that this model was effective in automatically generating adaptive navigational interfaces. In the related work, [46] presented the Supple system, which can automatically generate interfaces adapted to a person's devices, tasks, preferences, and abilities. Finally, this idea is not completely new for the users since major search engines continuously test different versions of the SUI using online A/B tests.

6.4.2 Micro-vertical / Task-oriented Search

General web search engines (e.g. Google, Bing, Yandex, Baidu) provide a unified interface to search for information. However, depending on a search task and a search goal, users sometimes switch to vertical search engines (e.g. LinkedIn and Indeed for jobs and people search, Twitter for real-time news search, Kayak for flights and hotel search, Amazon for product search), which provide superior search user experience by modeling vertical specific aspects in a more principled way, like in this thesis. This illustrates a more general concept of unbundling [98], when a more general product is replaced by smaller niche products that deliver more value. The question is what is a vertical search engine and how niche it could be? A user searching for a person to ask for career advice on LinkedIn might be willing to explore several diverse candidates, while a recruiter hiring people for a specific position will likely to focus only on people with the strictly matching skill set. Depending on the task, users employ different search moves, tactics, stratagems, and strategies [7]. It implies that a vertical search engine can be unbundled further to better support well-defined search tasks. To realize this idea, two different and complementary directions of research might be pursued.

First, we can build powerful machine learning models for task [81, 138] and intent [53, 107] detection from search query logs and, then, adjust the results based on the best task/intent guess predicted by the algorithm. This is the most dominant approach used by major search engines. We see many advantages in this approach. It minimizes user efforts, it can generalize to new queries and search tasks quickly, to name just a few. Therefore, we think that further research is necessary in this direction. However, we also see one major limitation. Machine learning algorithms don't have access to the human brains (yet), and, hence, they can only adjust the results for a task/intent to the point by mining the traces generated by users online.

Second, we think that this limitation might be addressed by involving users deeper into the search process and asking them directly about the search tasks they want to accomplish. In this case, we could decrease the asymmetry of information between a user and a search system, and, hence, deliver more relevant results. In turn, this opens up many new questions related to the SUI and interaction design. When should the search system ask a user about a task? What is the best way to ask the user about the task? How often can the system ask the users about the task? How could predictive models be used to facilitate the user in communicating their task to the system? How can we generate task names in a way clear for ordinary search engine users? By answering these questions, we will develop more interactive and effective search systems.

6.4.3 Transparent User-guided Search

Extending the idea from the previous Section 6.4.2, we think that users could be involved more deeply in the search process not only during the task specification stage but during the entire search session. Specifically, from the user studies described in this thesis, we learned that users wanted to specify the attributes shown in the snippets on the SERP, the order of sections on the detailed profile page for each search result, and the weights for different attributes, like in [52]. By looking at this idea from a different perspective, one can say that search engines should open up and become more transparent by exposing more search controls[2] and helping users better understand its inner workings as it was shown to positively affect user experience [42].

Along these lines, search engines can also increase transparency by showing more technical and quantitative information in the snippets. For example, from the study presented in Chapter 5, we learned that in the case of people search the searchers not only try to assess candidates based on their relevance to the query/task but also use other factors, such as social similarity, status, approachability, and seniority. Therefore, to simplify this process for the users, search engines might explicitly show on the SERP the similarity score between a searcher and a candidate result or a simple data visualization with the scores along several dimensions [52]. Furthermore, these scores can be made more interpretable by showing explanatory comments next to each number, e.g. *"87% (both you and John Smith went to MIT)"* just like it is done in recommendation systems [89, 114]. In Section 6.4.1, we proposed an idea to personalize the SUI for each user. Embracing the idea presented in this Section, we can also let users decide which version of the SUI to show. Each internal parameter of the search system exposed in the search user interface or even on the search settings page must be carefully tested, which creates numerous opportunities for research.

6.4.4 Immersive People Search

While working on the *"Find a Person to Ask for Career Advice"* task (Section 5.2.2) during the user study described in Chapter 5, the majority of participants said that they looked for people, who had similar

[2]sliders to adjust feature weights, knobs to communicate search tasks/intents, checkboxes to select relevant attributes.

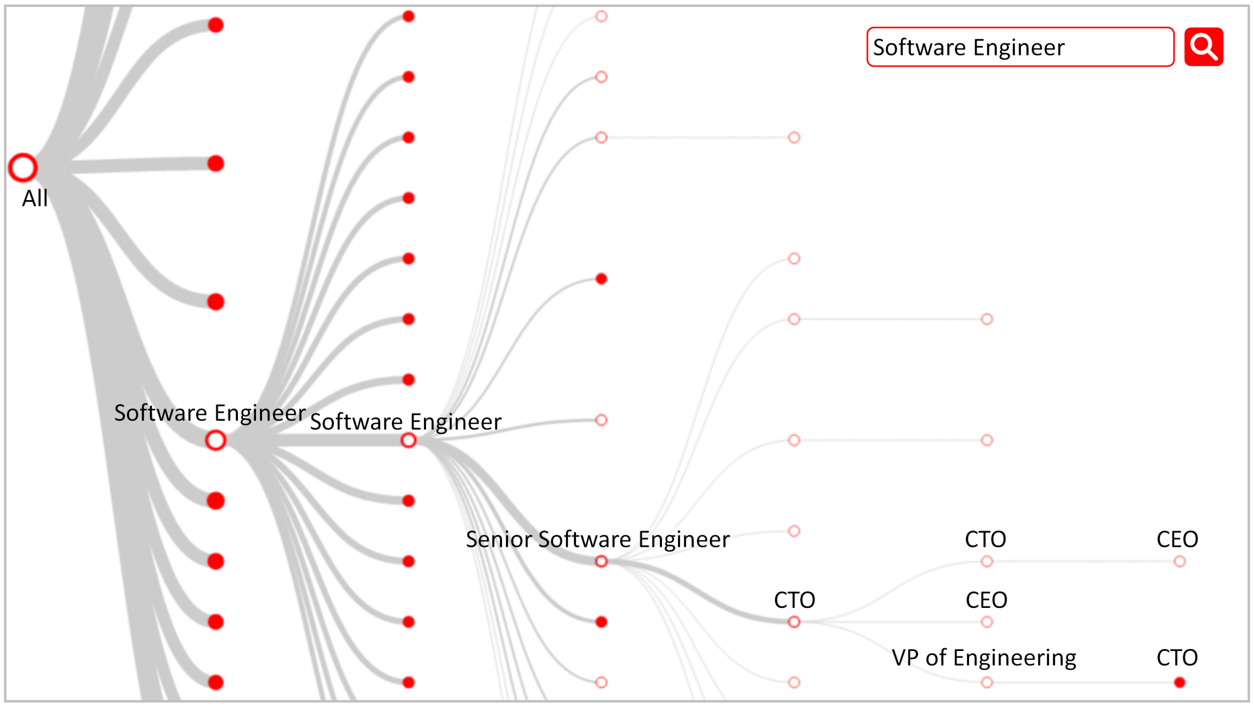


Figure 6.1: Interactive visualization of career trajectories providing more immersive way to do people search. It shows a tree of possible career trajectories for the Software Engineer and an exemplary path to the CTO.

to them background and who managed to grow into the position that they considered as desirable. The participants often remarked that they looked for matching educational background (major and degree) and work experience (job titles, companies, and years of experience). However, this information was somewhat hard to find since in the experimental application only one line per attribute per result was provided. Therefore, the participants had to check many detailed profiles before they could pick the best candidates. Plus, many participants tried to select a diverse set of candidates to be able to learn about various possible career paths. Yet, there was no way to look at the big picture and compare many candidates.

To facilitate users with this task, we envision a completely new immersive search user interface that employs unique visual information processing capabilities of humans [2] and allows to systematically explore the entire pool of candidates (Figure 6.1). As nodes, it has job titles, as edges extending to the right, it has career transitions over time, and on click, it shows candidates, which share the career trajectory. Similar to the faceted search interface [132, 152], it allows to dynamically filter nodes by submitting relevant keyword queries. With this interface, a searcher can see the entire tree of career trajectories based on millions of CVs, narrow down to the specific job title, pick on outliers with non-standard careers, and much more.

In turn, this interface creates numerous research questions that span data mining, data visualization, privacy, design, and social science. How can we visualize cases when one person works in two companies?

How can we identify insightful career trajectories? How can we protect the privacy of people presented in resumes yet provide value for the searchers? How can we normalize job titles (e.g. "Software Engineer" vs. "Software Developer")? How can we align people having a different number of jobs and duration in those jobs along the time axes? How can we index the tree to enable interactivity? We believe that similar applications could be created for other search tasks relevant for the users of PSNs (and beyond) and that they will inspire a new set of research questions.

6.4.5 Professional (Structured) Search on Mobile Devices

As mobile devices and search become more prevalent among users [34, 97], it is important to optimize and rethink the SUI and interactions for this emerging platform. We encourage SUI designers to reason from basic principles and create innovative native solutions taking into account the constraints and advantages of mobile devices instead of copying the techniques that work for the web. Natural language, voice, and visual input should significantly decrease users' efforts required to enter new queries and increase engagement. For example, imagine that we can search for a person we *"met yesterday at a conference"* by describing their appearance and other known attributes with a paragraph of natural language text dictated to a mobile microphone or by uploading their photo taken during the keynote presentation that they delivered.

Focusing on structured search, in particular, we think that new innovative ways for query formulation are necessary and feasible. For example, several of our user study participants mentioned that they would like to be able to search for similar jobs (Chapter 4) and people (Chapter 5) using the results that they found so far. Moreover, we think that "search by example" might become the prevalent type of structured search on mobile devices because: (1) different from web search, where the goal is either to find a definitive document on the topic or find a set of non-redundant documents complementing each other with new information [19], in the case of professional structured search the objects (jobs and people) have the clear distance semantics; (2) typing structured queries or using dropdowns is slow and difficult because of the small screen size; (3) voice input is not accurate enough for structured data. More research is necessary to make this real.

6.4.6 From Ten Blue Links to Conversational Search

Since the early days of information retrieval, SUIs haven't experienced many changes. As an input interface, we still use the simplistic query bar that allows to formulate keyword and, more recently, structured queries. As an output, we still present ten blue links (TBL) with free text and, more recently, structured snippets. In this thesis, we also focused on this traditional version of the SUI. However, as we described in Section 6.4.2, more effective niche micro-vertical search engines targeted to the specific tasks could be built. For example, in the context of professional social networks, we imagine an innovative conversational UI that allows searching for jobs in a more intuitive and interactive way.

The UI represents itself a standard messaging application, like Facebook Messenger or Telegram. As input, it takes voice/textual natural language replies. As output, it provides relevant jobs by sending the links to the chat. The unique feature of this SUI and application is that it is more interactive. Plus, we could make it more engaging by anthropomorphizing it. Ultimately, we envision that such a conversational job SUI/AI could replace career consultants by providing more affordable[3] access to personalized career advising services to many people. To make this vision the reality, we must answer many new research questions. What questions should the AI ask? How can we extract meaningful phrases from the user answers? How can we develop rational and coherent discourse from user replies within and across sessions? How can we recommend relevant jobs? How can we generate human-friendly answers and avoid repetitions? How can we incorporate user feedback and ignore deviations from the main topic? How can we measure user progress and engagement? With all these challenges solved, we could completely redefine the way people develop their careers and eliminate societal costs associated with the inefficient talent allocation.

Conversational search is also quite appropriate for people search and generally search over networked and interconnected datasets. Existing input interfaces for Graph Search are based on the textual input, which requires a lot of efforts. It is hard and inconvenient to type and modify long relational queries such as *"Friends of my friends who are Software Engineers and live in New York"*. With conversational and voice search the query formulation cost will decrease dramatically, which will lead to the increased search usage, and following the reinforcement loop to even more innovative input interfaces. It is reported by Google that voice search accounts for over 20% of all mobile search queries[4]. This trend will continue to grow.

[3]at the moment, a career consultant might charge $200-300 per session.

[4]http://searchengineland.com/google-reveals-20-percent-queries-voice-queries-249917

References

[1] L. A. Adamic and E. Adar. How to search a social network. *Social Networks*, 27(3):187–203, 2005.

[2] C. Ahlberg and B. Shneiderman. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, 1994.

[3] K. Andrews, V. Sabol, W. Lackner, C. Gütl, and J. Moser. Search result visualisation with xfind. In *Proceedings of the Second International Workshop on User Interfaces to Data Intensive Systems (UIDIS'01)*, UIDIS '01, 2001.

[4] A. Archambault and J. Grudin. A longitudinal study of facebook, linkedin, &#38; twitter use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2741–2750, New York, NY, USA, 2012. ACM.

[5] R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. KDD '07.

[6] M. Banko and E. Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting on association for computational linguistics*, pages 26–33. Association for Computational Linguistics, 2001.

[7] M. J. Bates. Where should the person stop and the information search interface start? *Information Processing & Management*, 26(5):575–591, 1990.

[8] D. Beaver, S. Kumar, H. C. Li, J. Sobel, P. Vajgel, et al. Finding a needle in haystack: Facebook's photo storage. In *OSDI*, volume 10, pages 1–8, 2010.

[9] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Hourly analysis of a very large topically categorized web query log. In *Proceedings of SIGIR '04*.

[10] N. J. Belkin. Some(what) grand challenges for information retrieval. *SIGIR Forum*, 42(1):47–54, June 2008.

[11] N. J. Belkin, D. Kelly, G. Kim, J.-Y. Kim, H.-J. Lee, G. Muresan, M.-C. Tang, X.-J. Yuan, and C. Cool. Query length in interactive information retrieval. In *Proceedings of ACM SIGIR '03*.

[12] R. Bellman. *Dynamic Programming*. Dover Books on Computer Science, USA, 2003.

[13] P. N. Bennett, R. W. White, W. Chu, S. T. Dumais, P. Bailey, F. Borisyuk, and X. Cui. Modeling the impact of short- and long-term behavior on search personalization. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, 2012.

[14] B. Bi, M. Shokouhi, M. Kosinski, and T. Graepel. Inferring the demographics of search users: Social data meets search queries. In *Proceedings of WWW '13*.

[15] S. Bodell and A. Hook. Using facebook for professional networking: A modern-day essential. *The British Journal of Occupational Therapy*, 74(12):588–590, 2011.

[16] B. Bollobás. *Graph theory.* Elsevier, 1982.

[17] P. Borlund. Experimental components for the evaluation of interactive information retrieval systems. *Journal of documentation*, 56(1):71–90, 2000.

[18] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International Conference on World Wide Web 7*, WWW7, 1998.

[19] A. Broder. A taxonomy of web search. In *ACM Sigir forum*, volume 36, pages 3–10. ACM, 2002.

[20] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1(1):538–549, 2008.

[21] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM, 1998.

[22] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. CIKM'09.

[23] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10.

[24] N. A. Christakis and J. H. Fowler. *Connected: The surprising power of our social networks and how they shape our lives.* Little, Brown, 2009.

[25] A. Chuklin, I. Markov, and M. de Rijke. *Click Models for Web Search.* Morgan & Claypool, 2015.

[26] A. Chuklin, P. Serdyukov, and M. de Rijke. Click model-based information retrieval metrics. SIGIR'13.

[27] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *Proceedings of Web Search and Data Mining 2008.*

[28] E. Cutrell and Z. Guan. What are you looking for?: An eye-tracking study of information usage in web search. In *Proceedings of the ACM SIGCHI 2007.*

[29] G. Das, V. Hristidis, N. Kapoor, and S. Sudarshan. Ordering the attributes of query results. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06, 2006.

[30] M. Das, H. Rahman, G. Das, and V. Hristidis. Generating informative snippet to maximize item visibility. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, 2013.

[31] J. Davitz, J. Yu, S. Basu, D. Gutelius, and A. Harris. ilink: Search and routing in social networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07.

[32] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of OSDI '04.*

[33] J. M. DiMicco and D. R. Millen. Identity management: Multiple presentations of self in facebook. In *Proceedings of the 2007 International ACM Conference on Supporting Group Work*, GROUP '07, pages 383–386, New York, NY, USA, 2007. ACM.

[34] J. Dischler. Building for the next moment. *Google Official Inside AdWords Blog*, February, 2015.

[35] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01.

[36] S. Duarte Torres, D. Hiemstra, and P. Serdyukov. An analysis of queries intended to search information for children. In *Proceedings of IIiX '10*.

[37] S. Duarte Torres and I. Weber. What and how children search on the web. In *Proceedings of CIKM '11*.

[38] S. T. Dumais and J. Nielsen. Automating the assignment of submitted manuscripts to reviewers. SIGIR '92.

[39] R. I. M. Dunbar. Coevolution of neocortical size, group size and language in humans. *Behavioral and Brain Sciences*, 16(4):681–735, 1993.

[40] G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *Proceedings of ACM SIGIR 2008*.

[41] M. Eagle and E. Leiter. Recall and recognition in intentional and incidental learning. *Journal of experimental psychology*, 68(1):58, 1964.

[42] M. Eslami, A. Rickman, K. Vaccaro, A. Aleyasen, A. Vuong, K. Karahalios, K. Hamilton, and C. Sandvig. "i always assumed that i wasn't really that close to [her]": Reasoning about invisible algorithms in news feeds. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, 2015.

[43] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *Proceedings of PODS '01*.

[44] D. Fallows. Search engine users. *Pew Internet & American Life Project*, 2004.

[45] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5), 2001.

[46] K. Z. Gajos, D. S. Weld, and J. O. Wobbrock. Automatically generating personalized user interfaces with supple. *Artif. Intell.*, 174(12-13):910–950, Aug. 2010.

[47] M. S. Granovetter. The strength of weak ties. *American journal of sociology*, pages 1360–1380, 1973.

[48] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04.

[49] K. Haas, P. Mika, P. Tarjan, and R. Blanco. Enhanced results for web search. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, 2011.

[50] A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 2009.

[51] J. Han. Mining heterogeneous information networks: The next frontier. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12.

[52] S. Han, D. He, J. Jiang, and Z. Yue. Supporting exploratory people search: A study of factor transparency and user control. In *Proceedings of CIKM '13*.

[53] A. Hanjalic, C. Kofler, and M. Larson. Intent and its discontents: The user at the wheel of the online video search engine. In *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, 2012.

[54] M. A. Hearst. *Search User Interfaces.* Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[55] W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185. ACM, 1999.

[56] R. Hoffman, B. Casnocha, and C. Yeh. *The alliance: Managing talent in the networked age.* Harvard Business Press, 2014.

[57] D. Horowitz and S. D. Kamvar. The anatomy of a large-scale social search engine. In *Proceedings of ACM Conference on WWW '10.*

[58] S.-W. Huang, D. Tunkelang, and K. Karahalios. The role of network distance in linkedin people search. In *Proceedings of ACM SIGIR '14 Conference.*

[59] Y. Huang, Z. Liu, and Y. Chen. eXtract: A snippet generation system for xml search. *Proc. VLDB Endow.*

[60] Y. Huang, Z. Liu, and Y. Chen. Query biased snippet generation in xml search. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, 2008.

[61] L. A. Jackson, K. S. Ervin, P. D. Gardner, and N. Schmitt. Gender and the internet: Women communicating and men searching. *Sex roles*, 44(5-6):363–379, 2001.

[62] B. Jacobs. *Categorical logic and type theory*, volume 141. Elsevier, 1999.

[63] B. J. Jansen, A. Spink, and T. Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information processing & management*, 36(2):207–227, 2000.

[64] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4).

[65] T. Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, 2006.

[66] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. SIGIR'05.

[67] H. E. Jones and H. S. Conrad. The growth and decline of intelligence: a study of a homogeneous group between the ages of ten and sixty. *Genetic Psychology Monographs*, 1933.

[68] R. Jones and K. L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. pages 699–708. ACM, 2008.

[69] J. M. Jose, J. Furner, and D. J. Harper. Spatial querying for image retrieval: A user-oriented evaluation. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, 1998.

[70] J. Kallinikos. *The consequences of information: Institutional implications of technological change.* Edward Elgar Publishing, 2007.

[71] H. Kautz, B. Selman, and M. Shah. Referral web: combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997.

[72] D. Kelly. Methods for evaluating interactive information retrieval systems with users. *Foundations and Trends in Information Retrieval*, 3(1–2):1–224, 2009.

[73] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03.

[74] A. Large, J. Beheshti, and T. Rahman. Gender differences in collaborative web searching behavior: an elementary school study. *Information Processing & Management*, 38(3):427–443, 2002.

[75] J. D. Lee, A. Kirlik, and N. B. Sarter. Multimodal displays: Conceptual basis, design guidance, and research needs, 2009.

[76] R. Lempel and S. Moran. Predictive caching and prefetching of query results in search engines. WWW '03.

[77] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 1(1).

[78] P. Li, A. Shrivastava, J. L. Moore, and A. C. König. Hashing algorithms for large-scale learning. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2672–2680. Curran Associates, Inc., 2011.

[79] Z. Liu, P. Sun, and Y. Chen. Structured search result differentiation. *Proc. VLDB Endow.*

[80] L. Lorigo, B. Pan, H. Hembrooke, T. Joachims, L. Granka, and G. Gay. The influence of task and gender on search and evaluation behavior using google. *Information Processing & Management*, 42(4):1123–1131, 2006.

[81] C. Lucchese, S. Orlando, R. Perego, F. Silvestri, and G. Tolomei. Discovering tasks from search engine query logs. *ACM Trans. Inf. Syst.*, 31(3):14:1–14:43, Aug. 2013.

[82] R. C. Lyndon, P. E. Schupp, R. Lyndon, and P. Schupp. *Combinatorial group theory*. Springer-Verlag Berlin, 1977.

[83] G. Marchionini. Exploratory search: from finding to understanding. *Communications of the ACM*, 2006.

[84] G. Marchionini. Toward human-computer information retrieval. *American Society for Information Science and Technology*, 2006.

[85] D. Marshall. *Celebrity and power: Fame in contemporary culture*. U of Minnesota Press, 1997.

[86] N. Matthijs and F. Radlinski. Personalizing web search using long term browsing history. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, 2011.

[87] D. W. McDonald and M. S. Ackerman. Just talk to me: A field study of expertise location. In *Proceedings of ACM Conference on CSCW '98*.

[88] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444, 2001.

[89] D. McSherry. Explanation in recommender systems. *Artificial Intelligence Review*, 24(2):179–197, 2005.

[90] Q. Mei, D. Cai, D. Zhang, and C. Zhai. Topic modeling with network regularization. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, 2008.

[91] M. Miah, G. Das, V. Hristidis, and H. Mannila. Standing out in a crowd: Selecting attributes for maximum visibility. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 356–365. IEEE, 2008.

[92] G. Mishne and M. de Rijke. A study of blog search. In *Proceedings of ECIR '06*.

[93] K. Murphy. From big data to big knowledge.

[94] U. Y. Nahm and R. J. Mooney. A mutually beneficial integration of data mining and information extraction. In *Proceedings of AAAI 2000*.

[95] M. E. Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[96] T. O'Reilly. *Networks and the Nature of the Firm*. Medium, 2015.

[97] J. O'Toole. Mobile apps overtake pc internet usage in u.s. *CNN Money*, February, 2014.

[98] G. G. Parker and M. W. Van Alstyne. Two-sided network effects: A theory of information product design. *Management science*, 51(10):1494–1504, 2005.

[99] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, 1998.

[100] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, and T. Carey. *Human-Computer Interaction*. Addison-Wesley Longman Ltd., Essex, UK, UK, 1994.

[101] K. Purcell, J. Brenner, and L. Rainie. Search engine use in 2012. *Pew Internet & American Life Project*, 2012.

[102] T. Qin, T.-Y. Liu, J. Xu, and H. Li. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Inf. Retr.*, 13(4).

[103] S. Robertson, H. Zaragoza, and M. Taylor. Simple bm25 extension to multiple weighted fields. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, 2004.

[104] E. M. Rogers. *Diffusion of innovations*. Free Press, New York, NY [u.a.], 5th edition, 08 2003.

[105] T. A. Salthouse. Independence of age-related influences on cognitive abilities across the life span. *Developmental Psychology*, 34(5):851, 1998.

[106] M. Sanderson and S. Dumais. Examining repetition in user search behavior. In *Proceedings of ECIR '07*.

[107] R. L. Santos, C. Macdonald, and I. Ounis. Intent-aware search result diversification. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 595–604. ACM, 2011.

[108] E. Schurman and J. Brutlag. Performance related changes and their user impact. In *velocity web performance and operations conference*, 2009.

[109] C. Shah and R. González-Ibáñez. Evaluating the synergic effect of collaboration in information seeking. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, 2011.

[110] B. Shneiderman. Direct manipulation for comprehensible, predictable and controllable user interfaces. IUI '97.

[111] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1997.

[112] B. Shneiderman, D. Byrd, and W. B. Croft. Clarifying search: A user-interface framework for text searches. Technical report, 1997.

[113] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. In *Proceedings of SIGIR '99*.

[114] R. Sinha and K. Swearingen. The role of transparency in recommender systems. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '02, 2002.

[115] M. M. Skeels and J. Grudin. When social networks cross boundaries: A case study of workplace use of facebook and linkedin. In *Proceedings of the ACM 2009 International Conference on Supporting Group Work*, GROUP '09, pages 95–104, New York, NY, USA, 2009. ACM.

[116] A. Spink, D. Wolfram, M. B. Jansen, and T. Saracevic. Searching the web: The public and their queries. *Journal of the American society for information science and technology*, 52(3):226–234, 2001.

[117] N. V. Spirin, J. He, M. Develin, K. G. Karahalios, and M. Boucher. People search within an online social network: Large scale analysis of facebook graph search query logs. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, 2014.

[118] L. Streeter and K. Lochbaum. An expert/expert-locating system based on automatic representation of semantic structure. *IEEE Artificial Intelligence Applications.*

[119] R. Sumbaly, J. Kreps, and S. Shah. The big data ecosystem at linkedin. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13. ACM, 2013.

[120] A. Sun, M. Hu, and E.-P. Lim. Searching blogs and news: A study on popular queries. In *Proceedings of SIGIR '08.*

[121] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu. Rankclus: Integrating clustering with ranking for heterogeneous information network analysis. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09.

[122] Y. Sun, Y. Yu, and J. Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09.

[123] A. Sutcliffe and M. Ennis. Towards a cognitive theory of information retrieval. *Interacting with computers*, 10(3):321–351, 1998.

[124] J. Tague-Sutcliffe. The pragmatics of information retrieval experimentation, revisited. *Inf. Process. Manage.*, 28(4):467–490, Mar. 1992.

[125] M. Tan, T. Xia, L. Guo, and S. Wang. Direct optimization of ranking measures for learning to rank models. KDD'13.

[126] J. Teevan, E. Adar, R. Jones, and M. A. S. Potts. Information re-retrieval: Repeat queries in yahoo's logs. In *Proceedings of SIGIR '07.*

[127] J. Teevan, S. T. Dumais, and D. J. Liebling. To personalize or not to personalize: Modeling queries with variation in user intent. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, 2008.

[128] J. Teevan, D. Ramage, and M. R. Morris. #twittersearch: a comparison of microblog search and web search. In *Proceedings of WSDM '11.*

[129] L. Terveen and D. W. McDonald. Social matching: A framework and research agenda. *ACM Trans. Comput.-Hum. Interact.*

[130] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive: A warehousing solution over a map-reduce framework. *Proc. VLDB Endow.*, 2(2), Aug. 2009.

[131] A. Tombros and M. Sanderson. Advantages of query biased summaries in information retrieval. In *Proceedings of ACM SIGIR 1998.*

[132] D. Tunkelang. Faceted search. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 2009.

[133] G. Turner. *Understanding celebrity.* Sage, 2004.

[134] S. K. Tyler and J. Teevan. Large scale query log analysis of re-finding. In *Proceedings of WSDM '10.*

[135] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[136] R. Varadarajan and V. Hristidis. A system for query-specific document summarization. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, 2006.

[137] A. Veerasamy and N. J. Belkin. Evaluation of a tool for visualization of information retrieval results. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96, 1996.

[138] M. Verma and E. Yilmaz. Entity oriented task extraction from query logs. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 1975–1978, New York, NY, USA, 2014. ACM.

[139] C. Wang, F. Jing, L. Zhang, and H.-J. Zhang. Learning query-biased web page summarization. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, 2007.

[140] I. Weber and C. Castillo. The demographics of web search. In *Proceedings of SIGIR '10.*

[141] I. Weber and A. Jaimes. Demographic information flows. In *Proceedings of CIKM '10.*

[142] I. Weber and A. Jaimes. Who uses web search for what: And how. In *Proceedings of WSDM '11.*

[143] W. Weerkamp, R. Berendsen, B. Kovachev, E. Meij, K. Balog, and M. de Rijke. People searching for people: Analysis of a people search engine log. In *Proceedings of ACM SIGIR '11.*

[144] R. W. White, M. Bilenko, and S. Cucerzan. Studying the use of popular destinations to enhance web search interaction. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, 2007.

[145] R. W. White, J. M. Jose, and I. Ruthven. A task-oriented study on the influencing effects of query-biased summarisation in web searching. *Information Processing & Management*, 39(5):707–733, 2003.

[146] M. L. Wilson. *Search User Interface Design.* Morgan & Claypool Publishers, 2011.

[147] T.-L. Wong, W. Lam, and B. Chen. Mining employment market via text block detection and adaptive cross-domain information extraction. In *Proceedings of ACM SIGIR 2009.*

[148] F. Wu, B. A. Huberman, L. A. Adamic, and J. R. Tyler. Information flow in social groups. *Physica A: Statistical Mechanics and its Applications*, 337(1):327–335, 2004.

[149] L. Wu, S. Shah, S. Choi, M. Tiwari, and C. Posse. The browsemaps: Collaborative filtering at linkedin. In *Proceedings of the 6th Workshop on Recommender Systems and the Social Web*, RSWeb '14, 2014.

[150] J. Xu, T.-Y. Liu, M. Lu, H. Li, and W.-Y. Ma. Directly optimizing evaluation measures in L2R. SIGIR'08.

[151] S. Yarosh, T. Matthews, and M. Zhou. Asking the right person: Supporting expertise selection in the enterprise. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, 2012.

[152] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03.

[153] S. Yogev, H. Roitman, D. Carmel, and N. Zwerdling. Towards expressive exploratory search over entity-relationship data. In *Proceedings of WWW '11.*

[154] H. Yu, C. Ho, Y. Juan, and C.-J. Lin. Libshorttext: A library for short-text classification and analysis.

[155] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14.

[156] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, pages 452–473, 1977.

[157] G. Zenz, X. Zhou, E. Minack, W. Siberski, and W. Nejdl. From keywords to semantic queries-incremental query construction on the semantic web. *Web Semant.*, 7(3), 2009.

[158] C. Zhai. Statistical language models for information retrieval. *Synthesis Lectures on Human Language Technologies*, 1(1):1–141, 2008.

[159] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, 2001.

[160] J. Zhang, J. Tang, and J. Li. Expert finding in a social network. In *Advances in Databases: Concepts, Systems and Applications*, pages 1066–1069. Springer, 2007.

[161] Y. Zhang and C. Zhai. Information retrieval as card playing: A formal model for optimizing interactive retrieval interface. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, 2015.