UCLA
UCLA Electronic Theses and Dissertations

Title
Recommendation Strategies Based on User-Generated Data

Permalink
https://escholarship.org/uc/item/5bf2d406

Author
Hsieh, Chu-Cheng

Publication Date
2013

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA
Los Angeles

Recommendation Strategies Based on User-Generated Data

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Chu-Cheng Hsieh

2013

© Copyright by
Chu-Cheng Hsieh
2013

ABSTRACT OF THE DISSERTATION

Recommendation Strategies Based on User-Generated Data

by

Chu-Cheng Hsieh

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2013

Professor Junghoo Cho, Chair

The challenge of discovering useful information from data has drawn much attention from researchers and scientists. In this work, we further explore the challenge by studying potential strategies and difficulties of harnessing human-generated data (content and activity). Facing the challenge of big data, our goal is to help web users satisfy their needs by providing recommendations and to guide them through the overwhelming amount of information on the internet.

Three aspects are targeted: crowd-sourcing, detection of trending topics, and the win-win principle. Correspondingly, three main applications are proposed: recommending similar items, popular items, and profitable items. For crowd-sourcing, we demonstrate the use of social tag data to find similar items. For detection of trending topics, we study bias sampling strategies to track trending news from accredited experts on Twitter. For the win-win principle, we investigate how to design query suggestions that maximize value to all interested parties. When a user is searching for solutions in a unfamiliar domain and is having a difficult time in forming effective queries, our work harnesses user-generated data to offer guidance, leading them to explore the sea of information more efficiently.

The dissertation of Chu-Cheng Hsieh is approved.

Wesley W. Chu

Carlo Zaniolo

Ying Nian Wu

Junghoo Cho, Committee Chair

University of California, Los Angeles

2013

To my parents, my wife, and my son . . .

who gave me their consistently support and understanding

during my journey toward being a PhD.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank Christopher Moghbel, Keith Chen, and Andrew Tai for their voluntary help in improving the editorial quality of this work as well as related publications, and my committee members (Professor Wesley W. Chu, Professor Carlo Zaniolo, and Professor Ying Nian Wu) who provided valuable comments in all stages of my research. In particular, I would like to express my sincere gratitude to my advisor, Professor Junghoo Cho, who advises, guides, and motivates me during my years in UCLA.

VITA

1999	Diploma (Electronic Engineering), National Taipei University of Technology.
2001	B.S. (Electronic Engineering), National Taiwan University of Science and Technology.
2003-2005	Lieutenant, Army, Taiwan
2005	M.S. (Electronic Engineering), National Taiwan University of Science and Technology.
2005–2007	Assistant Researcher, Chunghwa Telecom Laboratories, Taiwan.
2011	M.S. (Computer Science), UCLA.
2007–2012	Research Assistant, Computer Science Department, UCLA.
2008–2013	Teaching Fellow, Computer Science Department, UCLA.

PUBLICATIONS

Tzu-Yen Wang, Chin-Hsiung Wu, and Chu-Cheng Hsieh*. "A virus prevention model based on static analysis and data mining methods" *In Computer and Information Technology Workshops, 2008. CIT Workshops 2008. IEEE 8th International Conference on, pp. 288-293, IEEE, 2008.*

Tzu-Yen Wang, Chin-Hsiung Wu, and Chu-Cheng Hsieh*. "Detecting unknown malicious executables using portable executable headers." *In INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on, pp. 278-284, IEEE, 2009.*

Chu-Cheng Hsieh* and Junghoo Cho. "Finding similar items by leveraging social tag clouds." *In Proceedings of the 27th Annual ACM Symposium on Applied Computing,(SAC) pp. 644-651, ACM, 2012.*

Youngchul Cha, Bin Bi, Chu-Cheng Hsieh*, and Junghoo Cho. "Incorporating Popularity in Topic Models for Social Network Analysis." *In Proceedings of the 36th Annual ACM Special Interest Group on Information Retrieval (SIGIR), ACM, 2013.*

CHAPTER 1

Introduction

The emergence of search engines have been changing the way people acquire information. Most search engines, e.g. Google[1], ask users to describe their search intents by keywords and then perform keyword matching together with some ranking algorithm(s) to find relevant information. Essentially, the success of finding useful information often depends on the skills of transforming an intent into an effective query (consisting of useful keywords).

Sometimes a user just wants to satisfy a need quickly, and for most people, coming up with effective queries to satisfy the need is a time-consuming and tedious job. For example, to "plan a honeymoon", a person possibly starts with the query "*honeymoon locations*", digests pages, identifies candidate locations, and issues more queries to explore and to toil on digesting information, until some solution(s) are discovered. That is to say, a long process is expected because a search interface is designed to provide information, but not to provide recommendations for satisfying a need.

In the real world, a person might look for help from a professional to satisfy the person's need, and our research is motivated by observing how such a professional, e.g. a honeymoon consultant, can help satisfy the need (honeymoon). We investigate three aspects in this dissertation to help a user by making recommendations.

First, we study how to help a user find similar choices that share common characteristics by having the user provide some examples as inputs. Intuitively, we assume that a person who enjoys *swimming* likely enjoys *snorkeling*, and a customer who enjoys

[1] `www.google.com`

visiting a city with a cultural heritage would likely enjoy another such city. Therefore, we discuss strategies that can be used to help a user learn new keywords that can help them explore more information.

Second, we study how to identify trending / popular solutions to satisfy a need. In our honeymoon scenario, it is natural for a customer to ask "what are the top 10 honeymoon locations/activities?" To come up with an answer for this type of questions (i.e. "top N"), we have to first decide from whom we poll. Should we poll from some expert group or from everyone? If the answer is the former, what is the best strategy to form an expert group? As a result, we discuss strategies to form expert groups and compare whether polling from an expert group can be a better idea than from everyone.

Lastly, when a user starts his/her journey of information search, we study how to provide query suggestions, e.g. query expansion and related keywords, from the perspective of a service provider. In the real world, a honeymoon consultant would never suggest a tour he/she cannot offer, or the consultant may prefer to recommend some tours with higher commission. As the search service is widely provided in e-commerce websites, we further discuss algorithms that put into consideration the perspective of a service provider.

1.1 Challenges in Providing Recommendation

Here, we list the challenges involved in providing help that satisfies a need:

1. **What are the alternate choices?** One of the most intuitive ways to describe a vague need is to provide examples. In the above honeymoon scenario, suppose that a user is interested in visiting cities in Europe. The user may easily come up with examples like Athens or Rome. Since naming a city the user doesnt already know is infeasible, it would be helpful to provide users choices that shares common properties.

2. **What are the popular choices?** Another way to aid users is to provide popular choices. Often the most popular choices are, if not perfect, at least safe choices. Identifying trending choices from the entire world could consume lots of resources; therefore, it is important to explore possible sampling strategies and explore their limitations.

3. **What are the win-win choices?** Today providing a search interface is a popular practice all over the internet. When "search" is implemented as part of a website, especially an e-commerce one, it is essential to put the benefit of a service provider into consideration. For example, it is unreasonable to guide a customer to a query that leads to an unsatisfactory experience, say, a product in short supply. Therefore, we have to study how to guide customers to issue win-win queries.

1.2 Organization of Dissertation

In this dissertation, we deal with the aforementioned challenges by developing, implementing, and evaluating a variety of extensions. Along with our work, we create a query by example search interface, survey popular news detection strategies, and propose a visualized query suggestion application. The rest of this dissertation is organized as follows.

Chapter 2: We start by discussing how to find similar items as alternative choices (Challenge 1) based on social tag information. Social collaboration projects such as Wikipedia and Flickr have been gaining popularity, and more and more social tag information is being accumulated.

In this chapter, we demonstrate how to effectively use social tags created by humans to find similar items. We create a query-by-example interface for finding similar items through offering examples as a query. Our work aims to measure the similarity

between a query, expressed as a group of items, and another item by utilizing the tag information. We show that using human-generated tags to find similar items has at least two major challenges: popularity bias and the missing tag effect.

Chapter 3: To provide popular choices (Challenge 2), we need to answer whether we should pull opinions from everyone or we should pull opinions from selecting elites. In this chapter, we start with investigating the famous Efficient Market Hypothesis(EMH) [Fam70], which concludes that crowd wisdom is superior to any expert wisdom in selecting financial stocks.

Then we examine a similar hypothesis in the domain of news recommendation by conducting experiments on Twitter. We first identify a group of experts on Twitter who have been consistently recommending interesting news in the past and evaluate whether the news recommended by this group are more likely to be "interesting" than the news recommended by the overall "crowd."

Chapter 4: In this chapter, we discuss the strategies for leading users to issue win-win query reformations (Challenge 3). Despite the vast research on query suggestions, theres been a lack of emphasis on how an e-commerce website may adjust their query suggestion algorithms to accommodate their best interests.

We investigate how to mine query logs to build query suggestion algorithms that allow a website owner to designate the owner's business goal, for example, maximizing the chances of closing a future deal. We propose algorithms based on constructing a transition graph that models user activities from logs, and its interpretation leads to a Markov model interpretation. At its core, our algorithm aims to maintain, for each query suggestion, a right balance between the click-through rate and its forecasted benefit (at a provider's judgement).

CHAPTER 2

Query By Examples

2.1 Introduction

The dominant interface of search engines today requires users to pinpoint their information needs with a few keywords. Unfortunately, users sometimes find it difficult to identify the keywords that best describe their needs. For example, a user who plans to apply for a graduate school in California may issue the query "outstanding universities in California". However, many outstanding schools, such as Stanford University, are missing in the top results of all major search engines, because the keywords outstanding and California are not presented in the web pages of those schools.

As a potential solution to query-by-example problem, we study how we can provide a "query-by-example" interface. In this interface, users provide a few representative examples of the ultimate information they seek. The system then returns search results most similar to the examples provided; for instance, to find outstanding graduate schools, a user may issue a query like "Caltech, UC Berkeley" and expects that the system will return similar outstanding schools in California such as UCLA and Stanford University.

The major challenge in building such a system is to identify similar items based on the user-provided set of examples. In this study, we leverage the tag clouds that are collaboratively created by web users in defining and measuring the similarity between multiple items. To verify the effectiveness of our solutions, we conduct experiments on one of the largest social collaboration projects, Wikipedia. In the Wikipedia dataset, we

consider a wiki page or entry as an entity and a category label of a page as a tag. We aim to identify and rank entities that are similar to the user-provided examples based on tag information.

As other researchers (Shirky 2005; Golder et al. 2006; Mathes 2009) have noted, the uncontrolled nature of user-generated metadata, such as free-form tagging in Wikipedia, often causes problems of imprecision and ambiguity when these tags are used as a foundation in other applications. In our study, we deal with two challenging problems associated with free-form uncontrolled tag clouds: *popularity bias* and *the missing tag effect*.

We propose several approaches to overcome the challenges and subsequently build a prototype website allowing users to issue a query by examples. Our results show that our techniques are able to return a sizable number of high-quality similar items even when the user provides only a few examples in the query. The proposed approaches are evaluated against a benchmark dataset that is built based on 600 valid questionnaire responses from 69 students. In terms of user satisfaction, the questionnaire responses show that our techniques outperform Google Sets.

In summary, we highlight our contributions as follows:

- We investigate how to extract a set of similar items through analyzing noisy social tags created by human beings, and show that the tag information is effective in identifying relevant similar items.
- We identify and solve two challenges in tag-based search frameworks: popularity bias and the missing tag effect.
- We propose and compare several models based on tag information. We build algorithms on top of these models, and study their advantages and disadvantages.
- We perform an extensive evaluation based on user surveys and show that in terms of user satisfaction, our tag-based approaches outperform Google Sets in most

testing cases.

2.2 Problem Overview

The essential problem can be phrased as the following: users want to retrieve relevant items sharing some characteristics, and their queries are composed of representative examples with desired characteristics. We consider the "query-by-example" task as a process of finding similar items in a dataset, where the query is composed of a number of items from the same dataset.

Assuming a user issues a query $X_Q : \{x_{q1}, x_{q2}, ...\}$, the input to the framework is the query itself, where X_Q should be composed of entities, like $Caltech, UC\ Berkeley$. Our goal is to create a function R to measure the similarity between every entity in the dataset and the query X_Q, where a higher score measured by $R(x_i, X_Q)$ implies a higher similarity between an entity x_i and the query X_Q.

2.2.1 Tag Generation

We continue our discussion by introducing a tag-entity model, explaining how tags are generated. For example, in Fig 2.1, after reading the content of a Wikipedia page about Washington D.C., someone labels the page with the tags $City$, $Capital$, $North\ America$, etc. A tag does not have to be the same word used in the content; for example, the concept of a metropolis is captured by the tag $City$. In most social collaboration projects, a user can select any phrase as his tag, i.e. free-form tagging. The phrase may be an existing tag, a modified phrasing of an existing tag, or even a newly introduced tag.

Although we illustrate our tag-entity model by considering "wiki pages" as entities and their titles as identifiers, the content of entities are not limited to plain texts. In this model, we utilize only tag information, ignoring the content of entity. This simplification allows our techniques to be broadly applicable to non-textual dataset as well.

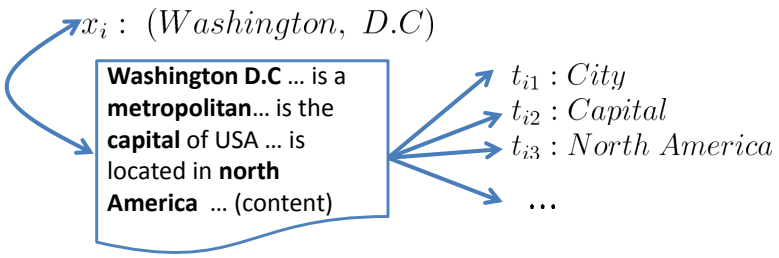

Figure 2.1: The data model

We use X_U to represent all entities in a dataset, that is, the universe of entities. We use $T_i = \{t_{i1}, t_{i2},\}$ to represent the tags associated with x_i. The universe of tags is denoted by T_U, referring to all tags in a dataset.

We illustrate our notations with the following examples:

Example 1. *Consider a dataset that contains four entities:*

$x_1 : Beijing \rightarrow T_1 : \{City, Capital, Asia, Summer\ Olympic\}$

$x_2 : Washington\ D.C. \rightarrow T_2 : \{City, Capital, North\ America\}$

$x_3 : London \rightarrow T_3 : \{City, Capital, Europe, Summer\ Olympic\}$

$x_4 : Los\ Angeles \rightarrow T_4 : \{City, North\ America\}$

In this example, the dataset contains a total of six tags:

$t_1 : City, t_2 : Capital, t_3 : Asia, t_4 : Summer\ Olympic,$

$t_5 : North\ America, t_6 : Europe$

□

We define the function $\mathcal{E}(t_n)$ as an operation for acquiring all entities associated with the tag t_n. For example, $\mathcal{E}(t_2 : Capital) = \{x_1 : Beijin, x_2 : Washington\ D.C.\ , x_3 :\ London\}$. In addition, we define $\mathcal{E}(t_m, t_n,)$ as $\mathcal{E}(t_m) \cup \mathcal{E}(t_n) \cup$ Namely, in Example 1, $\mathcal{E}(t_3 : Asia\ , t_6 : Europe) = \mathcal{E}(t_3) \cup \mathcal{E}(t_6) = \{x_1 : Beijin, x_3 : London\}$.

Without referring to the contents of an entity, we limit our similarity measurement to tag information. One might argue that tag-entity relations are unreliable and some-

times a reasonable relation is missing from an entity. We will discuss these concerns regarding the imperfect nature later.

2.3 The Intersection-Driven Approach

The core challenge in providing a query-by-example service is to figure out exactly what types of entities a user is looking for based on the input entities provided by the user. To examine how to approach this problem, we illustrate with a scenario as follows. In Example 1, when the user-provided input is the query $\{x_1 : Beijing, x_2 : Washington\ D.C.\}$, what will be a reasonable interpretation of the users intention? Both entities are associated with the tags $t_1 : City$ and $t_2 : Capital$ as we can see from $T_1 \cap T_2 = \{t_1 : City, t_2 : Capital\}$. That is, both entities are cities and capitals. Given this result, we may have two interpretations: the user is looking for cities that are also capitals, or the user is simply looking for cities, but the input examples happen to be capitals as well. Since the second interpretation is possible, not only $t_1 : City$ but also $t_2 : Capital$ should be weighed when we identify other similar entities.

Although the input entity Beijing is also associated with the tag $t_3 : Asia$, it is unlikely that the user is only looking for cities in Asia because this tag is not associated with $x_2 : Washington\ D.C.$. In the intersection-driven approach, if a tag is associated with only a subset of input examples, we claim that the user is unlikely to only look for entities associated with such a tag.

2.3.1 Similarity Measurement

Here, we define the degree of similarity between an entity x_i and the query X_Q as follows:

$$R(x_i, X_Q) = |T_i \cap T_Q^{\cap}| \tag{2.1}$$

The symbol $T_Q^{\cap}$ stands for tags associated with all entities in a query, i.e. $T_Q^{\cap} =$

$T_1 \cap T_2 \cap \cap T_n$, $\forall x_i \in X_Q$. In Example 1, if a query consists of entities $\{x_1 : Beijing$, $x_2 : Washington\ D.C.\}$, the set $T_Q^{\cap} = \{t_1 : City$, $t_2 : Capital\}$.

According to our definition in Equation 2.1, the more tags in $T_Q^{\cap}$ an entity is associated with, the more similar (to the query X_Q) the entity is. For instance, if an entity x_j is a city but not a capital, such as the entity $x_4 : Los\ Angeles$, it is associated with $t_1 : City$ but not with $t_2 : Capital$. According to the similarity definition in Equation 2.1, the ranking score is $R(x_4 : Los\ Angeles , X_Q) = 1$, meaning that the entity $x_4 : Los\ Angeles$ has only one tag in common with $T_Q^{\cap}$; Likewise, the entity $x_3 : London$ has the ranking score of $R(x_3 : London, X_Q) = |T_3 \cap T_Q^{\cap}| = 2$, meaning that the entity $x_3 : London$ has two tags in common with $T_Q^{\cap}$.

In the above example, our approach ranks $x_3 : London$ higher than the entity $x_4 : Los\ Angeles$. This result seems intuitive because no matter whether the users intent is to find cities or to find cities that are also capitals, the entity $x_3 : London$ is always an appropriate answer. The entity $x_4 : LosAngeles$ is inferior to $x_3 : London$ because $x_4 : Los\ Angeles$ is an inappropriate answer if the users intent is to find cities that are also capitals. Since we cannot deny such a possibility, ranking $x_3 : London$ higher than $x_4 : Los\ Angeles$ is reasonable.

2.3.2 Challenges

We create Example 2 based on our observations on the Wikipedia dataset. The example illustrates and highlights the challenges we expect to encounter in a real dataset. In Example 2, an underlined notation signifies that a tag is very popular. Also, we strike-through a tag, representing that a tag-entity relation should exist but does not appear in a real dataset.

Example 2. *Consider a dataset that contains six entities:*

$x_1 : Beijing \rightarrow T_1 : \{City, Capital, Asia, Summer\ Olympic, China, \underline{\cancel{Object}}\}$

$x_2 : Washington\ D.C. \rightarrow T_2 : \{\underline{\cancel{City}}, Capital, North\ America, \underline{Object}\}$

$x_3 : London \rightarrow T_3 : \{City, \cancel{Capital}, Europe, Summer\ Olympic, \underline{Object}\}$

$x_4 : Los\ Angeles \rightarrow T_4 : \{City, North\ America, \underline{Object}\}$

$x_5 : Michael\ Phelps \rightarrow T_5 : \{Summer\ Olympic, \underline{Object}\}$

$x_6 : Lyon \rightarrow T_6 : \{City, Europe, \underline{Object}\}$

In this example, the dataset contains a total of six tags:

$t_1 : City, t_2 : Capital, t_3 : Asia, t_4 : Summer\ Olympic,$

$t_5 : North\ America, t_6 : Europe, t_7 : China, t_8 : Object$

□

2.3.2.1 Missing Tag Effect

Not only in Example 2, but also in practice a tag-entity relation can be missing. There are several possible reasons of why this may happen. In any social collaboration project, a newly created entity might not be well tagged until its editors finish revising all the content of the entity. At the same time, the community may not be aware of a newly created tag or may decide not to use the newly created tag for other entities.

Missing tag-entity relations could cause the system to misinterpret user intent. For example, suppose that a users query $X_Q = \{x_2 : Washington\ D.C, .x_3 : London\}$ has been issued against Example 2 dataset; since the tag $t_1 : City$ is missing in $x_2 : Washington\ D.C.$, and the tag $t_2 : Capital$ is missing in x_3:London, the intersectiondriven approach interprets the user intention as finding entities associated with the tag $\{t_8 : Object\}$. Given these two input entities, intuitively, we feel that such an interpretation Object is problematic because the interpretation is too general. The intersectiondriven approach considers the entity $x_1 : Beijing$ as an irrelevant one, and suggests that $x_4 : LosAngeles$, $x_5 : MichaelPhelps$, and $x_6 : Lyon$ are equally similar to the query X_Q.

The impact of the missing tag effect is more pronounced as more entities are included in a query. Suppose that a user is looking for a set of entities associated with a

tag t_k; ideally, the tag t_k is expected to be associated with all the entities in the query. If we use the notation α to represent the probability of missing the tag t_k, the probability of t_k being not associated with all input examples becomes

$$P(missing\ \ t_k\ \ in\ \ T_Q^{\cap}) = 1 - (1-\alpha)^{|X_Q|} \tag{2.2}$$

, where X_Q is the number of entities in the query. If the value of α is 20% and the number of input examples is 3, the probability of missing the tag t_k in $T_Q^{\cap}$ is close to a half (48.8%). When the number of input examples increases to 10, the chance of missing the desired tag increases to 89.26%.

2.3.2.2 Partial Weighting Generalization

To generalize the intersection-driven approach for addressing the missing tag effect, one solution is to assign scores in real number, instead of either zero or one, to tags that are associated with only some entities in a query. In the previous example ($X_Q = \{x_2 : Washington\ D.C.,\ x_3 : London\}$), we now assign 0.5 to these tags: $t_1 : City,\ t_2 : Capital,\ t_4 : Summer\ Olympic,\ t_5 : North$ America, and $t_6 : Europe$, because we have two entities in the query and each tag associates with only one entity. These tags, although not in $T_Q^{\cap}$ ($\{t_8 : Object\}$), are now assigned partial weights in real numbers, and thus contribute to the similarity function $R(x_i, X_Q)$. As a result, $R(x_4 : Los\ Angeles, X_Q) = 2$ because not only the tag $t_8 : Object$ contributes 1 point, but also $t_1 : City$ and $t5 : North\ America$ contribute 0.5 points separately. Similarly, $R(x_5 : Michael\ Phelps,\ X_Q) = 1.5$ and $R(x_6 : Lyon,\ X_Q = 2$. The system returns a better ranking result 1st position: $x_4 : Los\ Angeles$ and $x_6 : Lyon$, 2nd position: $x_1 : Beijing$ and $x_5 : Michael\ Phelps$.

The strategy weights tags that appear in $T_Q^{\cap}$ the most heavily, and thus ensures that we follow the same intuition: the more tags in $T_Q^{\cap}$ an entity is associated with, the more similar (to the query X_Q) the entity is. Moreover, in case the intent cannot be captured in $T_Q^{\cap}$, the strategy helps the system return some related results as long as some tags are

associated with one or more entities in the query.

Such a generalization brings more entities to results. For example, in Consider a dataset that contains six entities. For example, in Example 2, assigning partial weight to the tag $t_4 : Summer\ Olympic$ brings $x_5 : Michael\ Phelps$ to the result. Nevertheless, introducing this suspicious result may not be a good idea. Therefore, if the system already returns satisfied results, we tend to not adopt generalization unless the user asks for more results.

2.3.2.3 Popularity Bias

Both results in the previous research [GH06] and results in our experiments show that the number of tags associated with an entity follows a power law distribution. That is, only a few entities are associated with a large number of tags, and most entities are associated with a small number of tags.

We define the popularity of an entity x_i based on the number of tags associated with x_i, denoted by T_i. Then, an entity x_i is more popular than another entity x_j if $|T_i| > |Tj|$. Similarly, we say that a tag t_k is more popular than another tag t'_k if $|\mathcal{E}(t_k)| > |\mathcal{E}(t'_k)|$, where $|\mathcal{E}(t_k)|$ represents the set of all entities associated with t_k.

We define the *popularity entity-bias* as follows: if we measure the similarity between an entity and a query based on tag information in the query, we tend to favor entities that are more popular. For example, a query X_Q consists of $\{x_1 : Beijing,\ x6 : Lyon\}$. To process the query, we firstly identify all cities. Then, to further rank all cities, we assign full weight (1.0) to the tag City, and partial weight (0.5) to these tags: Capital, Asia, Summer Olympic, China, Europe and Object. Since most of the tags originate from $x_1 : Beijing$, entities similar to $x_1 : Beijing$ are more likely to be returned in the result and ranked in higher positions.

The popularity bias happens in tags as well, and we name this kind of bias as *popularity tag-bias*. We argue that although a popular tag like $t_8 : Object$ in Example 2

is associated with some input example(s), this popular tag is probably not the concept the user intends to search for. That is, the tag $t_8 : Object$ exists in $T_Q^{\cap}$ probably only because it is popular, i.e. $|\mathcal{E}(t_8 : Object)|$ is a large number. In Example 2, if we randomly select two entities to create a query, $t_8 : Object$ is the most likely tag shown in $T_Q^{\cap}$.

2.4 Balanced Voting Model

We now further refine the intersection-driven approach to include the following properties: (1) a popular entity in a query should not unfairly influence the results such that the results are similar only to the popular entity, but not to others, and (2) even a few tag-entity relations are missing in input examples, the system should be able to identify relevant entities based on tags associated with a subset of input examples.

$$R(t_n, X_Q) = \begin{cases} \sum_{x_i \in X_Q} \frac{1}{|T_i|} & , \ if\ t_n \in T_i \\ 0 & , \ otherwise \end{cases} \tag{2.3}$$

$$R(x_i, X_Q) = \sum_{t_n \in T_i \cap T_Q^{cup}} R(t_n, X_Q) \tag{2.4}$$

To achieve the above desired properties, in this model, we define the similarity between an entity x_i and the query X_Q as follows:

Equation 2.3 can mitigate both the missing tag effect and the popularity entity-bias. In a nutshell, we consider every entity in the query as having a total score of one in Equation 2.3. Then, each tag associated with the entity is assigned an equal portion of the entitys score. The non-zero assignment alleviates the missing tag effect in that we now value significance of tags that are missing in some input examples. Furthermore, the assigned score of a tag is inversely proportional to $|T_i|$, i.e., the number of tags associated with x_i. As a result, a tag originating from a popular entity will get a lower

score than a tag originating from a less popular entity, alleviating the popularity entity bias.

In Equation 2.4, we introduce the symbol $T_Q^{\cup}$ to represent tags associated with one or more entities in a query X_Q, where the symbol $\cup$ stands for the notion union. We calculate the relevance score for every entity in the dataset with Equation 2.4 by summing up the relevance scores of all tags associated with the entity. Here, $R(t_n, X_Q)$ represents the relevance score of the tag tn to the query X_Q. Note that in this equation we assign a non-zero score to each tag in $T_Q^{\cup}$; therefore, a tag associated with only a subset of the input entities will still get a non-zero score. The following example shows how the scores are computed under this definition.

In Example 2, each tag associated with the entity $x_1 : Beijing$ gains 0.2 points because five tags are associated with it. Similarly, each tag associated with the entity $x_6 : Lyon$ gets 0.33 points. Namely, if a query X_Q consists of $x_1 : Beijing,\ x_6 : Lyon$, the $R(t1 : City,\ X_Q) = 0.2 + 0.33 = 0.53$. Although $x_1 : Beijing$ is associated with more tags than $x_6 : Lyon$, each tag in $x_6 : Lyon$ contributes a higher ranking score than a tag in $x_1 : Beijing$.

We calculate the relevance score for every entity in the dataset with Equation 2.4. For instance, the entity $x_3 : London$ gets a relevance score of 1.39 points by summing up the score of the tags associated with $x_3 : London$, including $t1 : City(0.53),\ t4 : Summer\ Olympic(0.2),\ t6 : Europe(0.33)$, and $t_8 : Object(0.33)$. Tags belonging to the set $T_Q^{\cup}$, such as $t_1 : City$, still contribute the highest ranking scores. Therefore, the balanced voting approach clings to our belief that tags shared by all entities in a query likely capture a users intention; meanwhile, it compensates biases caused by a popular entity in a query. Compared to our intersection-driven approach, each tag in a popular entity like $x_1 : Beijing$ contributes less influence in terms of ranking scores now. This difference makes our balanced voting approach less sensitive to the popularity entity-bias.

2.5 One-class Probabilistic Model

2.5.1 Measuring Similarity Using Probability

So far, we view the similarity measurement as determining a good weighting scheme for tags associated with input examples. Alternatively, we can consider the problem as computing the probability that an entity is the target a user is looking for. Given some entities as possible results, we can then rank the entities based on the calculated probabilities.

We start by thinking about how people create a query for finding similar items. We think that a user firstly has a desired property in mind, and then the user tries to recall some entities with the desired property based on his/her knowledge. If the intent can be captured by a tag (or tags) in a dataset, entities associated with the tag(s) in the dataset are considered as similar entities.

We propose the one-class probabilistic model based on the following assumptions. At first, we assume that a users intent corresponds to one tag t_k in a dataset. Since the intent is unpredictable, we claim that the desired tag t_k is randomly selected from all tags. Then, once the desired tag is selected, the user randomly selects $|X_Q|$ entities from $\mathcal{E}(t_k)$ to synthesize the query, where $\mathcal{E}(t_k)$ are all entities that are associated with t_k.

2.5.2 Measuring Similarity Using Probability

In order to better understand how our one-class probabilistic model works, in this section we discuss the model under the premise that there is no missing tags in the dataset. We will reconsider the problem of missing tags later in the next section. When a query X_Q is given, based on our one-class probabilistic model, the probability that x_i is what a user is looking for can be computed as

$$P(x_i|X_Q) \triangleq \sum_{t_k} P(x_i|t_k) * P(t_k|X_Q) \tag{2.5}$$

In Equation 2.5, the $P(t_k|X_Q)$ stands for the probability that a tag t_k is the desired tag when the system is given the user query X_Q; $P(x_i|t_k)$ stands for the probability that the system should return an entity x_i if the desired tag is t_k. Here, we assume that all tags are independent from each other, so we can sum up all the probability scores when more than one tag is considered. Using Bayes theorem, we can find an equivalent form to Equation 2.5, as shown in

Proposition 1. *Ranking entities based on $P(x_i|X_Q)$ is equivalent to ranking entities through the formula $\sum_{t_k \in T_i} P(X_Q|t_k)$. That is,*

$$P(x_i|X_Q) \propto \sum_{t_k \in T_i} P(X_Q|t_k) \tag{2.6}$$

(Proof) The idea of our model is captured through the following equation:

$$P(x_i|X_Q) = \sum_{t_k} P(x_i|t_k) * P(t_k|X_Q) \tag{2.7}$$

In one-class probabilistic model, we claim that once a tag t_k is the desired tag(representing a user intent), all entities associated with the tag t_k are appropriate results. Furthermore, if one of the tags associated with an entity x_i is t_k, we believe that the entity x_i is an appropriate result. That is,

$$P(x_i \in \mathcal{E}(t_k)|t_k) = \begin{cases} 1, & if\ t_k \in T_i \\ 0, & if\ t_k \notin T_i \end{cases}$$

By substituting Equation 2.8 into Equation 2.7, we get

$$\sum_{t_k} P(x_i|t_k) * P(t_k|X_Q) = \sum_{t_k \in T_i} P(t_k|X_Q) \tag{2.8}$$

The probability $P(t_k|X_Q)$ represents the probability that a tag t_k is the tag representing a user intent while seeing a query X_Q. To compute this probability, we apply Bayes' theorem:

$$P(t_k|X_Q) = P(X_Q|t_k) * \frac{P(t_k)}{P(X_Q)} \tag{2.9}$$

Since we have no information about how a user chooses the desired tag t_k, we assume that the desired tag is randomly selected from all tags, and thus the probability $P(t_k)$ is a constant. In other words, we argue that every tag has the same possibility to be selected to form a query. $P(X_Q)$ is also a constant because for each ranking task, we deal with the same query X_Q, Thus,

$$C = \frac{P(t_k)}{P(X_Q)}, where\ C\ is\ a\ constant \tag{2.10}$$

We substitute Equation 2.9 and 2.10 to Equation 2.8, we now show that

$$P(x_i|X_Q) = \sum_{t_k \in T_i} P(t_k|X_Q) \propto \sum_{t_k \in T_i} P(X_Q|t_k) \tag{2.11}$$

□

Given a query X_Q, $|X_Q|$ is the number of input examples. In this model, if the tag t_k is the desired tag, representing a users intention, our premise says that the query is created by randomly selecting $|X_Q|$ entities from the set $\mathcal{E}(t_k)$. $|\mathcal{E}(t_k)|$ represents the number of entities associated with a tag t_k, so we have $\binom{|\mathcal{E}(t_k)|}{|X_Q|}$ distinct choices to create a query because the user can randomly select $|X_Q|$ entities from $\mathcal{E}(t_k)$, where $\binom{|\mathcal{E}(t_k)|}{|X_Q|}$ stands for the number of combinations in a set with $|X_Q|$ distinct elements. Since the query X_Q is one of the $\binom{|\mathcal{E}(t_k)|}{|X_Q|}$ outcomes, the probability of having X_Q being the query and t_k being the desired tag is

$$P(t_k|X_Q) = \frac{1}{\binom{|\mathcal{E}(t_k)|}{|X_Q|}} \tag{2.12}$$

In this model, entities in the query are selected from entities associated with the desired tag t_k, representing the desired property in the users mind. If all tag-entity relations in a dataset are established, the desired tag must be one of the tags that are shared by all entities in the query, i.e., $t_k \in T_Q^{\cap}$. In addition, we know an entity x_i can be a similar entity to the query only if the desired tag t_k is associated with the entity, i.e., $t_k \in T_i$. Thus, for any entity x_i, only tags in $(T_i \cap T_Q^{\cap})$ are considered.

If an entity x_i is associated with two or more tags in $T_Q^{\cap}$, we sum up all probability values to get the probability of the entity xi being a similar entity. Then, we rank every entity in a dataset based on this probability.

We summarize the above discussions as follows:

Proposition 2. *If every tag is correctly associated with all entities to which it is related (no missing tag), we show that*

$$\sum_{t_k \in T_i} P(t_k|X_Q) = \sum_{t_k \in (T_i \cap T_Q^{\cap})} \frac{1}{\binom{|\mathcal{E}(t_k)|}{|X_Q|}} \tag{2.13}$$

(Proof) From Equation 2.12, for a desired tag t_k, we know the probability of having X_Q as query is $1/\binom{|\mathcal{E}(t_k)|}{|X_Q|}$.

Since X_Q is always randomly selected from $\mathcal{E}(t_k)$, if a tag is not associated with all input examples, it cannot be the t_k, In other word, we know

$$P(X_Q|t_k) = 0 \; if \; t_k \notin T_Q^{\cap} \tag{2.14}$$

Through adopting the notion we introduce in Equation 2.14 to the Equation 2.6, Proposition 2 is derived.

□

Equation 2.13 has an advantage of alleviating the popularity tag-bias because the system will assign a low value to $P(X_Q|t_k)$ for a popular tag. The equation conveys the

notion: when a rarely seen tag ($|\mathcal{E}(t_k)|$: a small number) is shared by all entities of the query, the probability that the rarely seen tag is what the user desires is high because it is unlikely that input examples are associated with the tag by chance. As a result, Equation 2.13 places more value on it than on a popular common tag ($|\mathcal{E}(t_k)|$: a large number).

In Example 2, suppose that we see a tag $t_8 : Object$ in $T_Q^{\cap}$; we could argue that $t_8 : Object$ exists because a user is looking for objects, or because $t_8 : Object$ is associated with almost every entity in the dataset. If $|\mathcal{E}(t_8 : Object)|$ is a large number, the chance of the latter is high, and thus we could reason the tag $t_8 : Object$ might not be the desired tag. In other words, the model alleviates popularity tag-bias through assigning a low value $P(X_Q|t_k)$ to a popular tag.

2.5.3 Refinement of the Approach

In this section, we deal with the missing tag effect. We start with introducing some new notations for extending our one-class probabilistic model.

The function $\mathcal{E}(t_k)^C$ returns entities that are relevant to the tag tk but the tag-entity relation is missing, and $|\mathcal{E}(t_k)^C|$ is the number of entities missing the tag t_k. For instance, in Example 2, $\mathcal{E}(t_8 : Object)^C = \{x_1 : Beijing\}$

The symbol m_k denotes the number of entities missing a tag t_k in a query. For example, in Example 2, a query $\{x_1 : Beijing,\ x_2 : WashingtonD.C.,\ x3 : London\}$ has values $|X_Q = 3|$ because the query contains three input examples, and for the tag $t_1 : City$, $m_1 = 1$ because only one entity ($x_2 : WashingtonD.C.$) is missing the tag $t_1 : City$.

The symbol u stands for the number of all entities in our dataset; the symbol $T_Q^{\cup}$ represent all tags associated with at least one input example. Then, we generalize Proposition 2 as follows:

Proposition 3. *When considering the missing tags effect , we show that* $\sum_{t_k \in T_i} P(t_k|X_Q)$

in Proposition 1 can be approximated by the following formula:

$$\sum_{t_k \in T_i} P(t_k|X_Q) = \sum_{t_k \in (T_i \cap T_Q^{\cup})} \left[(\frac{|\mathcal{E}^C(t_k)|}{u})^{m_k} * \frac{1}{\binom{|\mathcal{E}(t_k)| + |\mathcal{E}^C(t_k)|}{|X_Q|}} \right] \tag{2.15}$$

,where we claim that the dataset has the following properties: (1) $u \gg |\mathcal{E}(t_k)|$ *and* $u \gg |\mathcal{E}^C(t_k)|$ *and (2)*$(|\mathcal{E}(t_k)| + |\mathcal{E}^C(t_k)|) \gg |X_Q|$.

(Proof) If entities of a query X_Q are selected from entities associated with the desire tag t_k, but some entities in query X_Q are not associated with the tag t_k, the only explanation is that such entities are suffering from the missing tag effect. For instance, in Example 2, if the desired tag is the tag $t_1 : City$, a user creates a query $\{x_1 : Beijing,\ x_2 : Washington\ D.C.,\ x_3 : London\}$, but the intersection $T_Q^{\cap}$ is an empty set. Under the premise of the model, we conclude that $T_Q^{\cap} \in \emptyset$ is caused by the fact that the tag $t_1 : City$ is missing in $x_2 : Washington\ D.C$.

Note that we define $\overline{\mathcal{E}(t_k)}$, the complement of the set $\mathcal{E}(t_k)$, as $X_U - \mathcal{E}(t_k)$, and thus $\{X_Q \cap \overline{\mathcal{E}(t_k)}\}$ refers to all entities in X_Q that are not associated with t_k. The function $\mathcal{E}^C(t_k)$ returns entities that are relevant to the tag t_k but the tag-entity relation is missing. We use the probability $P(\{X_Q \cap \overline{\mathcal{E}(t_k)}\} \in \mathcal{E}^C(t_k)|t_k)$ to represent the chance that all entities in the query X_Q that are not associated with the desired tag t_k happen to be caused by the missing tag effect.

We summarize the above discussion and revise the Proposition 1 to include the possibility that a tag is not in $T_Q^{\cap}$ but is the desired tag t_k as follows:

$$\sum_{t_k \in T_i} P(t_k|X_Q) \propto \sum_{t_k \in T_i} P(\{X_Q \cap \overline{\mathcal{E}(t_k)}\} \in \mathcal{E}^C(t_k)|t_k) * P(X_Q|t_k) \tag{2.16}$$

The way we compute probability $P(X_Q|t_k)$ is identical to Equation 2.12, except that we have to consider the missing tag effect:

$$P(X_Q|t_k) = \frac{1}{\binom{|\mathcal{E}(t_k)| + |\mathcal{E}^C(t_k)|}{|X_Q|}} \tag{2.17}$$

The probability $P(\{X_Q \cap \overline{\mathcal{E}(t_k)}\} \in \mathcal{E}^C(t_k)|t_k)$ can be computed by

$$P(\{X_Q \cap \overline{\mathcal{E}(t_k)}\} \in \mathcal{E}^C(t_k)|t_k) = \frac{\binom{u - |\mathcal{E}(t_k)| - m_k}{|\mathcal{E}^C(t_k)| - m_k}}{\binom{u - |\mathcal{E}(t_k)|}{|\mathcal{E}^C(t_k)|}} \tag{2.18}$$

To calculate the number of combinations is very expensive. By applying some mathematic simplifications, an approximated value of Equation 2.20 is derived as follows:

$$\frac{\binom{u - |\mathcal{E}(t_k)| - m_k}{|\mathcal{E}^C(t_k)| - m_k}}{\binom{u - |\mathcal{E}(t_k)|}{|\mathcal{E}^C(t_k)|}} \simeq \left(\frac{|\mathcal{E}^C(t_k)|}{u - |\mathcal{E}(t_k)|}\right)^{m_k} \tag{2.19}$$

According to Equation 2.16, a tag that is never associated with any entities in the query X_Q still can be the desired tag. To compute this probability, we can simply set $m_k = n$. However, in such a case, the Equation 2.19 is close to zero, so we only take $t_k \in T_Q^{\cup}$ into account, where $T_Q^{\cup}$ represents all tags associated with *at least one* input example. By substituting Equation 2.17, 2.18, and 2.19 into equation 2.16, we get:

$$P(x_i|X_Q) \propto \sum_{t_k \in (T_i \cap T_Q^{\cup})} \left[(\frac{|\mathcal{E}^C(t_k)|}{u - |\mathcal{E}(t_k)|})^{m_k} \right] * \frac{1}{\binom{|\mathcal{E}(t_k)| + |\mathcal{E}^C(t_k)|}{|X_Q|}}$$

Through acknowledging two constraints: (1) $\gg |\mathcal{E}(t_k)|;\ u \gg |\mathcal{E}^C(t_k)|$, and (2) $(|\mathcal{E}(t_k)| + |\mathcal{E}^C(t_k)|) \gg |X_Q|$, we can simplify Equation 2.20 to Equation 2.15. Proposition 3 is derived.

□

The two properties in Proposition 3 are easily satisfied in practice. The first property, $u \gg |\mathcal{E}(t_k)|$ and $u \gg |\mathcal{E}^C(t_k)|$, is satisfied if the number of all entities in a dataset is larger than the number of entities associated with any individual tag. In most social collaboration projects, the number of entities is a very large number, for example, 3,459,565 entities in our experiment dataset (Wikipedia), so the property is satisfied. The second property, $(|\mathcal{E}(t_k)| + |\mathcal{E}^C(t_k)|) \gg |X_Q|$, is also satisfied because we believe that in practice a user provides a small portion of desired results as input examples and expects a sizable number of results [AK08].

The part of the equation $(\frac{|\mathcal{E}^C(t_k)|}{u})^{m_k}$ can be interpreted as an adjustment for missing a tag t_k in some input examples. When a tag is shared by all the input entities, the number m is zero and this part of the equation becomes one, meaning that no adjustment are required. As the number of input entities missing a tag t_k increases (m increases), the value decreases exponentially. Thus, this part adheres to the notion we learned in naive intersection model that tags in $T_Q^{\cap}$ are important.

When we consider the missing tag effect, the part $1/\binom{|\mathcal{E}(t_k)| + |\mathcal{E}^C(t_k)|}{|X_Q|}$ can be interpreted as a refinement for addressing the popularity of a tag. The concept is identical to the explanation of $1/\binom{|\mathcal{E}(t_k)|}{|X_Q|}$ in Proposition 2, where we tend to place more value on a rarely-seen tag in the query X_Q than a popular tag, so that the model alleviates the popularity tag-bias.

In our experiments, since the ratio of missing tags is unknown, we simply make the following assumption: 50% of tag-entity relations being missing. Thus, if half tag-entity relations are missing, we could conclude $|\mathcal{E}(t_k)| \simeq |\mathcal{E}^C(t_k)|$. In practical, to make a reasonable approximation of tag-missing ratio requires the understanding of target datasets and some heuristic trials.

2.6 Experiment

Evaluating effectiveness of different approaches is a challenging task because the quality of results is subjective and no standard corpus exists. Thus, we think the best way to evaluate a ranking algorithm is to build a search engine and see how well users perceive our new ranking results.

Some IR communities, such as TREC (Text retrieval conference; http://trec.nist.gov), provide datasets for evaluating keyword search. Unfortunately, those datasets are not collected for testing query-by-example search. As a result, we download the dataset of Wikipedia and create a search interface on top of the dataset for collecting user surveys [1].

In this section, we are going to provide the overview of our dataset, explain the design of the surveys, and then analyze the users' satisfaction scores for each proposed algorithm.

2.6.1 A Study of Dataset – Wikipedia

We implement our system based on a Wikipedia snapshot dumped on November 3rd, 2009. Wikipedia is the largest collaborative encyclopedia, and it contains more than 3 million articles in English. The collaborative nature means that all tags in Wikipedia are generated by human, and some tag-entity relations might be missing.

We consider every wiki page as an entity, so the name (page title) of the wiki page becomes a unique identifier of the entity. For tag information, every wiki page contains "category" entries, and we consider each category as a tag. When a category is labeled with an article, we consider the category (the tag) is associated with the article (the entity). The dataset contains 471,443 unique tags, 3,459,565 entities, and 13,196,971 associations.

[1] The user surveys can be downloaded from http://oak.cs.ucla.edu/~chucheng/publication/qbe_survey_results_public.zip

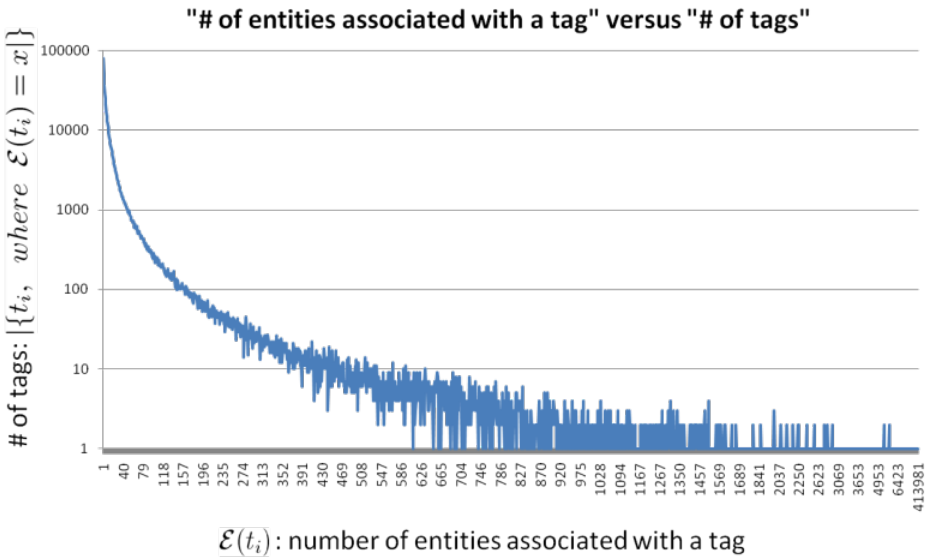

Figure 2.2: Size of category distribution

In Fig. 2.2, the x-axis refers to $|\mathcal{E}(t_i)|$, the number of entities associated with the tag t_i, and the y-axis refers to the number of tags with the size $|\mathcal{E}(t_i)|$. For example, the value of y-axis of "x=1" is 79,870, which means that 79,870 tags (about 16% of all tags) appear only once. These tags are possibly newly created, or they have not yet been accepted by other users. On the other hand, the right-most point of the line in Fig. 2 is the Living People, and its y value, one, means that only this tag is associated with 413,981 entities (value of x-axis).

The chart in Fig. 2.2 shows that some tags are extremely popular and many tags are unpopular. Our statistics show that about 16% of tags are singleton, 74% of tags are associated with 2 to 50 entities, and less than 10of tags are associated with more than fifty entities ($|\mathcal{E}(t_i)| > 50$). Golder et al. [GH06] reported a similar distribution on Del.icio.us data where a power law distribution was also observed.

2.6.2 Effectiveness Evaluation

Evaluating the similarity between an entity and a query is difficult because whether the entity is similar to another entity in the query is subjective. For example, someone might argue that the entity Nikon (a manufacturer of cameras) is not similar to the entity Toyota (a manufacturer of cars), but another person may feel they are similar because both of them are Japanese companies. Thus, we decide to create a benchmark through conducting user surveys.

2.6.2.1 Experiment Design

We collected 600 valid questionnaires from 69 students in UCLA to create a benchmark for evaluating user satisfaction. In each questionnaire, the system randomly selects one of 10 pre-set scenarios, where every scenario contains two to three entities to form a query. The 10 pre-set scenarios are:

- America pop singers (Query: Britney Spears and Michael Jackson)
- Famous basketball players (Query: Michael Jordan and Kobe Bryant)
- Coffee shops (any shops that offer coffee as a main product) (Query: Coffee Bean, Peet's Coffee & Tea and Seattle's Best Coffee)
- Scenic spots in Bali Island (Query: Jimbaran Beach and Kuta)
- Cities that hosted the Olympic games (Query: Beijing and Atlanta)
- Metropolitan cities (Query: Beijing, London and Taipei)
- Giant software companies (Query: Microsoft, IBM and Sun Microsystems)
- Ivy League institutions (Query: Harvard University and Cornell University)
- Car Manufacturers (Query: Toyota and Honda)

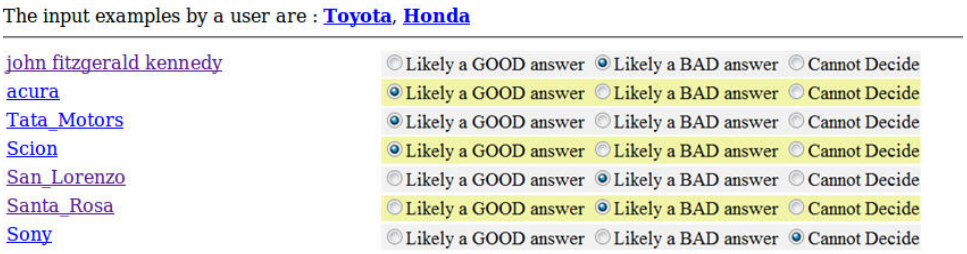

Figure 2.3: A screen clip of a questionnaire

- Fast-food chains (Query: Wendy's, McDonald's and Burger King)

Once the scenario is selected, the query is issued to our system and we collect the top 200 results from each algorithm. We mix all the top 200 results, and the questionnaire is then generated through randomly selecting entities from the mixed set. Note that we run a customized questionnaire generation process so that high-rank results are reviewed by more persons. Entities in the first few pages, high-ranked results, are often considered important because users tend to read results according to the order of entities. With limited resource (questionnaire takers), we are more interested in knowing how each approach performs in these high rank results. Therefore, in the process of questionnaire generation, 30 questions are drawn from high-rank entities, particularly the entity in the top 40 results of any model, and 10 questions are drawn from the other entities.

To avoid bias, questionnaire takers are unaware of how their questionnaires are generated. They are told that entities in a query belong to a group where all entities share some properties (the intent of the query). They are asked to examine whether an entity in a questionnaire belongs to the intent of the query. The intent of the query, such as Car Manufacturers, is not revealed to the questionnaire taker. They are asked to read the Wikipedia pages and make an evaluation of possible intentions to the query. Fig. 2.3 is a (partial) snapshot of a questionnaire.

Every questionnaire contains 50 entities; however, 10 of 50 entities are known to be clearly unrelated to the query for filtering out invalid surveys. If any pre-set unrelated

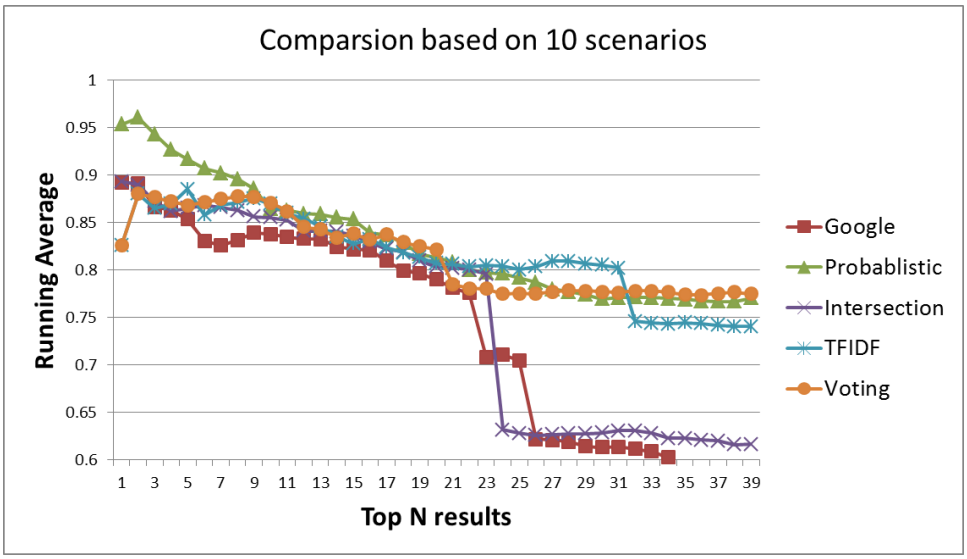

Figure 2.4: Comparison of different models (top 40)

entity is marked as a positive example, we rule out the entire questionnaire. Thus, only 600 of 714 questionnaires are considered valid after the screening process, and a total of 24,000 feedbacks are collected.

2.6.2.2 Benchmarks and Results

We use Equation 2.20 to calculate the satisfaction score of an entity to a query. Whenever a questionnaire taker reports a good answer, the corresponding entity earns the full credit of one. While he cannot make a clear judgment, we still assign 0.5 point to the entity. After averaging scores of an entity, a high satisfaction score implies that most users believe that the entity and entities in the query are similar. And a score close to 0.5 could imply a situation that users hold their opinions. We then consider the satisfaction scores as benchmarks for evaluating different algorithms.

$$S(x_i|X_Q) = \frac{\#\ of\ Good + 0.5\ *\ \#\ of\ Cannot\ Decide}{\#\ of\ total\ response} \tag{2.20}$$

Fig. 2.4 contains the comparison results of all models based on the ten scenarios we use in experiments. In addition to the four algorithms proposed in the paper, we add

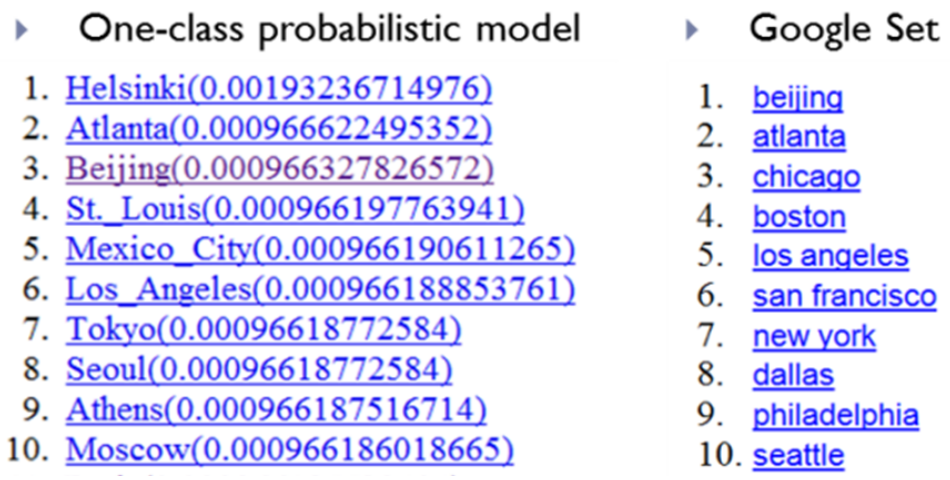

Figure 2.5: One class probabilistic model vs. Google Sets

Google Sets [2] and the term frequency–inverse document frequency (TFIDF) algorithm into the comparison. The x-axis represents for top N results, and the y-axis indicates the running average of satisfaction scores for top N results.

In Fig. 2.4, our one class probabilistic model outperforms other models in the top 10 results; our balanced voting model and our one class probabilistic model have high user satisfactions while comparing top 40 results. We notice that the intersection model and Google Sets return approximately only 25 entities on average. As a result, the average user satisfaction promptly drops in that a user cannot find any results. The probabilistic model reports a promising result: 5.26% more satisfaction than the Google Sets based on 2400 questions in 600 questionnaires.

2.6.2.3 Discussion

Although the algorithms of Google Sets are not disclosed, we compare our results against its result because Google is currently the leading search engine and Google Sets aims to solve the same problem: finding similar items through quires consisting of examples.

[2] http://labs.google.com/sets; Note that Google shutdown all lab projects in 2011, including Google Sets.

Fig. 2.5 demonstrates a query in which most survey responders feel the results provided by Google Sets are inferior or questionable. We create a query Beijing, Atlanta, where the intent behind the query is to find cities that hosted the Olympic Games. Google Sets seems to interpret our intent as finding cities in America. In contrast, our one-class probabilistic model identifies cities that hosted the Olympic Games.

Though which interpretation is better is controversial, during interviews after the survey, many responders said that their first impressions after seeing Beijing, Atlanta shown in the query were about the event that Beijing hosted the 2008 Summer Olympics, or about the notion that both of them are metropolises. Even for responders preferring the notion metropolises, they were still unhappy when seeing a result that is biased towards Atlanta and neglects properties contributed by Beijing.

2.7 Related Work

The study of social collaboration tagging system has been attracting attention from researchers. Mathes [Mat04] investigated the social tag data and pointed out that it was a fundamentally chaotic. Shirkly [3] also argued that using tag information is difficult because a user has the freedom of choosing any word he or she wants as a tag. Later, Gloder et al. [GH06] analyzed the structure of collaborative tagging systems on top of Del.icio.us data and concluded that tag data follow a power law distribution. Their studies back up our argument that some tags are extremely popular while others are rarely used.

Despite the challenges in using tag-entity information, many researchers continue to work in the field and have shown the potential of social tag clouds. Tso-Sutter et al. [TMS08] used relationships among tags, users, and entities to recommend possible interesting entities to users. Penev et al. [PW08] used WordNet to acquire terms relating to a tag and applied TFIDF [RJ76] similarity on both the tags and their related terms

[3] http://www.shirky.com/writings/ontology-overrated.html

for finding similar entities (pages in their research). They used WordNet to interpret the meaning of tags, trying to measure the similarity between entities based on keywords through expanding tags with WordNet. Although we aim to solve similar problems, our approaches focus on using only tag information, because we believe, as Strohmaier et al. [SKK10] suggested, that users use social tags for categorizing and describing resources.

We believe that our study can benefit many applications. For example, Givon et al. [GL09] showed that social tags can be used in dealing with recommendations in large-scale book datasets. Moreover, the keyword generation task can be considered as a similar problem, for example, Fuxman et al[FTA08] draw an analogy from a keyword to a url and an entity to a tag. Finding similar entities based on shared tags is similar to finding related keywords based on shared urls that users click on.

Many researchers also work on tag ranking or tag aggregation. Recently, Wetzker et al. [WZB10] focused on creating a mapping between personal tags to aggregate tags with the same meaning, Heymann et al. [HPG10] used tag information to organize library data, Wu et al. [WYY09] explained how to avoid noise and compensate for semantic loss, Liu et al. [LHY09] studied how to rank tags based on importance, and Dattolo et al. [DEM11] studied how to identify similar tags through detecting relationships between them.

2.8 Summary

In this chapter, we investigated the problem of finding similar items with query-by-example interface on top of social tag clouds. We introduced three approaches, and built a search engine on top of them, creating a benchmark for evaluating the users' satisfaction through collecting 600 questionnaires. The experiment results suggest that social tag data, even though they are uncontrolled and noisy, are sufficient for finding similar items. Finally, we show that both the voting model and the one-class probabilis-

tic model reach high user satisfaction.

We explain two important challenges of utilizing tag information: popularity bias and the missing tag effect, and explain how to overcome these difficulties through partial weight strategies and probability utilization. We show that, in terms of users' satisfaction, our algorithms are superior or at least compatible to Google Sets and TFIDF model. Our approaches return hundreds of relevant entities without sacrificing the quality in the top results. Moreover, our models rely on only social tag information.

Our proposed framework not only provides an ability to find similar items, but also shows the application potential of social tag information. We demonstrate that the task can be accomplished through providing a query consisting of entities and using only tag information, even though the tag information is uncontrolled and noisy. Through this study, queries for finding similar items, such as "Honda or Toyota or similar", are handled properly. Our research also highlights the value of using social collaboration data, tag clouds, to refine existing search technologies.

CHAPTER 3

Query By Sampling – Experts vs The Crowd

3.1 Introduction

Life in human society relies on collective decisions. Often people believe two (or more) heads are better than one when making a decision, because individual judgments may suffer from bias or limited information. For example, researchers [PBW90] discovered that a pair of inexperienced physicians may collectively make better decisions than one experienced physician because they can take advantage of their complementary abilities and avoid extreme estimates. Political scientists [Sur05] believe that democratic forms of governance benefit from the wisdom of the populace.

In this study, we further investigate this problem of whether decisions are better made by acknowledged experts than by the crowd. If such an argument holds, one may take advantage of this fact, because polling a small group of experts can be accomplished much easier than polling the crowd (in order to aggregate the crowd's wisdom). For example, passing legislation through a parliamentary system is often faster and easier than having the populace vote directly [Bat86]. However, whether experts exist is itself a long debated question: the "Efficient Market Hypothesis(EMH)" [Fam70], a famous hypothesis proposed in the 1960s in the finance domain, concludes that no expert can consistently outperform the market with regards to making stock investments in an informationally efficient market.

We investigate whether a similar hypothesis holds in the *news domain* by conducting experiments on Twitter. Twitter is one of the largest social, micro-blogging web-

site, allowing users to publish text-based messages consisting of at most 140 characters, called *tweets*. According to Java et al. [JSF07], sharing URLs and spreading news are two main purposes of using Twitter, in addition to online chatting.

Towards this goal, we collect tweets from a large number of users on Twitter and examine their tweets to identify a group of "experts" who have consistently discovered "interesting" news early on — the exact definition of interesting news will be given later, but it essentially means that the news is widely circulated on Twitter over time — and have recommended them in their tweets. After identifying this expert group, we then collect two sets of fresh news: a set of news that appears in this "expert" group's tweets and another set of news that appears in the tweets of the "crowd". We then observe, for a few months, which news in the two sets are circulated more on Twitter and become more popular (thus, are likely to be more interesting).

Our results lead us to a conclusion similar to the EMH — Expert wisdom did not outperform crowd wisdom (of the entire population) with regard to predicting popular news in the future. That is, in our repeated experiments with various parameter settings, we could not find an expert group on Twitter, whose recommended news set contained more popular news than the set recommended by the crowd.

We then proceed to investigate (1) whether there exist certain circumstances when an expert group might have an advantage, and (2) possible strategies to further improve crowd wisdom. Firstly, we limit our news recommendations to a specific topic domain, e.g., international-politics, to see whether an expert group can perform better in a specific domain. We observed a similar conclusion, despite spotting some exceptions. Secondly, we identified two strategies that could be used to improve crowd wisdom: (a) augmenting crowd recommendation with a small number of news that all experts agree to recommend and (b) reducing noise in crowd wisdom by removing users who "talk too much". Lastly, we discuss how to extend our work for news recommendation, as a sampling strategy designed to maximize resource usage.

3.2 Twitter Data

Comparing expert wisdom against crowd wisdom in the news domain requires a dataset on *who* recommends *what* news articles *when*, and *how interesting* the recommended articles are. Forming large groups to conduct such experiments has traditionally been infeasible. Now with the help of social websites where people organically share interesting news, we are able to simulate such an experiment.

Researchers stated in their studies [JSF07, KLP10, PMS09], reporting and sharing news stories is one of the main activities on Twitter. In this study, we use Twitter as our data source, from which each user's news recommendation behavior is collected and analyzed. We now describe how we collected our dataset from Twitter in detail and present a few relevant statistics from the collected dataset.

3.2.1 Filtering News Tweets

Ideally, we would like to collect all tweets from Twitter, identify tweets that contain a link to a news article, and use them for our experiments. But several practical limitations prevent us from taking this approach. First, the number of tweets published on Twitter is significantly larger than what our infrastructure can handle. Second, to avoid abuse, Twitter puts a daily cap on the number of tweets downloadable via their streaming API, so even if we did have sufficient infrastructure, we are unable to download all such tweets through the official Twitter API. Third, the exact definition of a link to a "news article" is quite fuzzy. For these reasons, we decided to download only the tweets that contain a link to *The New York Times* Web site via the Twitter streaming API. Initially, we also downloaded tweets with a link to a number of other well-known news outlets, such as CNN and Los Angeles Times, but our preliminary study showed that, in terms of the number of tweets circulating on Twitter, *The New York Times* outnumbers others by at least a scale of ten, so focusing on the most popular news Web site seems to be sufficient for our purposes.

Downloading all tweets containing a link to *The New York Times* Web site is relatively straightforward. The Twitter API provides a mechanism to specify a set of string filters, such that only those tweets that contain the strings are streamed through the API. Since the full URL `http://www.nytimes.com/` or the shortened URL `http://nyti.ms/`[1] both contain the strings "**http nyti**", we provide them as the string filter for our tweet stream.[2] In summary, we define the term "news tweet" as follows in this study:

News tweet: If a tweet contains a link that points to the news website `http://www.nytimes.com`, we say the tweet is a *news tweet*. Here we have a broad definition of news tweets, including tweets pointing to reader letters, opinions, and other news-related articles.

To trace news-article circulation on Twitter, we need a unique identifier for each article. The URL embedded in every news tweet is not an ideal identifier because one article might be linked to by multiple URLs; for example, a news article can be accessed through a full URL `http://www.nytimes.com/` and a shortened URL `http://nyti.ms/`, or a URL may include a few extra parameters at the end for tracking purposes, even though they all point to the same news article. For this reason, we decide to use the *title* of the news article page as the identifier of a news article as follows:

News-tweet thread: Assume multiple tweets contain different URLs of New York Times Web pages. When the pages are downloaded, however, if all pages share the same *title* (i.e., the text enclosed in the `<TITLE>` tag are the same), we consider them as the same news article (from the perspective of content). All tweets referring to the

[1] By default, a click of the Twitter button on pages of *The New York Times* will be converted to a short URLs prefixed with `http://ntyi.ms/`.

[2] In the past, people often used a URL shortening service when they embed a link to a Web page, but after Twitter started providing their own URL shortening service, most tweets now contain a direct link.

same news article are then categorized into the same *news-tweet thread*.

3.2.2 Data Collection and Cleaning

Following the aforementioned definitions, we collected news tweets for six months, starting from August 1st, 2011 to January 31th, 2012, giving a total of 4,234,899 raw tweets.

Tweeting activities for some news stories are only partially observed, and we aim to avoid those stories. For example, if we observed the first tweet in a news-tweet thread on August 1st, 2011, we do not know whether this is indeed the first time that the news article was mentioned on Twitter or if the article was mentioned earlier on Twitter, but we do not have the earlier tweet due to our limited data collection period.

To exclude partially observed news stories, we *censor* our data for the *first* and the *last month*. Censoring is a popular strategy in statistics for dealing with missing data problems. More precisely, we keep a news-tweet thread in our dataset if and only if the first tweet of the thread appeared on September 1st, 2011 or later and the last tweet appeared on December 31st, 2011 or earlier. As we will discuss in more detail later, ensuring a period of inactivity of one month in the beginning and at the end guarantees that we have all tweets in a news-tweet threads with high probability. After censoring, a total of 2,837,026 tweets (from 402,102 users) survived and were used for conducting experiments.

3.2.3 Relevant Statistics

We now investigate data we gathered and present statistics that are relevant to our later discussion. In particular, we provide answers to the following questions: (1) how widely is a news article circulated on Twitter? and (2) how quickly do people on Twitter lose interest in a news article?

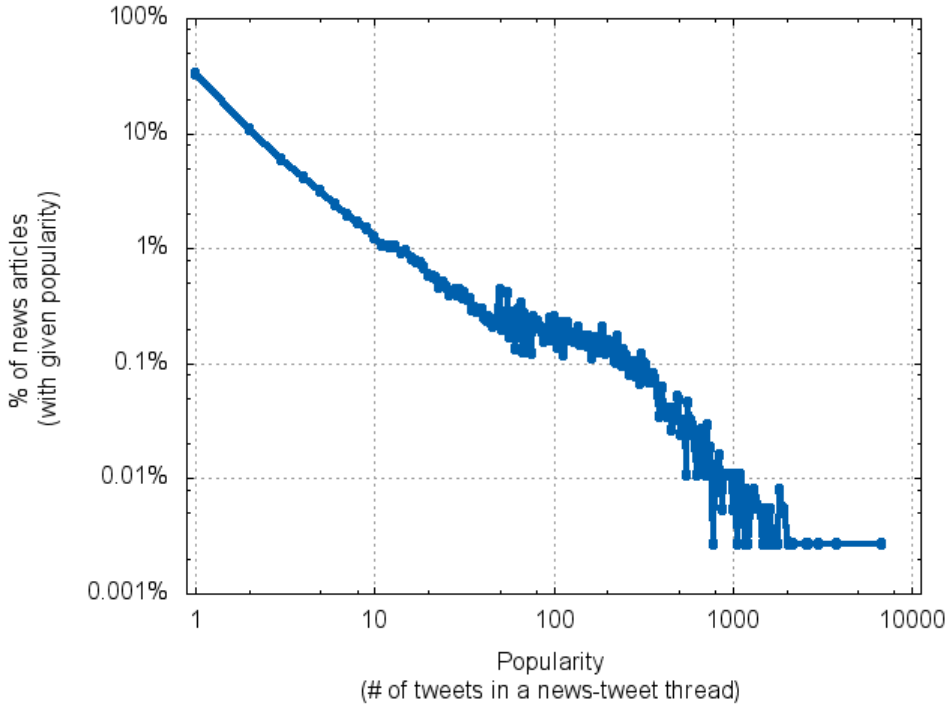

Figure 3.1: Popularity of News Articles on Twitter

3.2.3.1 Popularity of News Articles

We first present the popularity of news articles on Twitter, where popularity is measured by how many tweets appear in the news-tweet thread of each article. Figure 3.1 shows the popularity distribution. The horizontal axis corresponds to popularity (i.e., the number of tweets in a news-tweet thread) and the vertical axis corresponds to the number of news articles (or equivalently the number of news-tweet threads) with the given popularity. Both horizontal and vertical axes are logarithmic. The maximum popularity is 6,535. We see that the graph is roughly straight except the tail end, indicating that it is a power-law distribution, which is commonly observed in human behavioral datasets. We also see that a large number of news articles (38%) appeared only once on Twitter.

Table 3.1 shows the top five popular news titles. Interestingly, none of them originally appeared on the front page of the *The New York Times* website. These stories become popular on Twitter eventually, probably because of the power of forwarding

Title	Category	Short URL	# of tweets
Penn State Said to Be Planning Paterno Exit Amid Scandal	Sports	http://nyti.ms/vC9Ccg	6535
The Joy of Quiet	Opinions	http://nyti.ms/tZr7p2	4933
Google's Lab of Wildest Dreams	Technology	http://nyti.ms/uj0cpu	3286
Touch Of Evil	Gallery	http://nyti.ms/udI0wJ	3218
A Dispute Over Who Owns a Twitter Account Goes to Court	Technology	http://nyti.ms/uEWPRD	2989

Table 3.1: Top five news articles

and sharing on Twitter.

3.2.3.2 Longevity of News-tweet Threads

How long do people show interest in a news story? Intuitively, one may suggest that users' interest in a news article *starts* when the first user shares it with his/her peers. Their interest *ends* when no one visits the news-article page and reads it anymore. While we do not have access to user log data for the *The New York Times* website, we may approximate the birth and the death of people's interest in news by defining the birth as the time we see the first tweet linking to a news article on Twitter and the death as the time we see the last tweet linking to the news article. That is, we define the *longevity of a news-tweet thread* as the time between the first and the last (re)tweets in a news-tweet thread[3] and interpret it as the approximate interval in which people are interested in the corresponding news article.

We show our findings about the longevity of news-tweet threads in Figure 3.2. We show the cumulative longevity distribution both for *all* tweets and for the top 2% most

[3]More precisely, we set the death time to be when we see 90% of tweets being reported on Twitter. This cut-off point of 90% is to avoid an overly extended longevity due to a few outlier tweets.

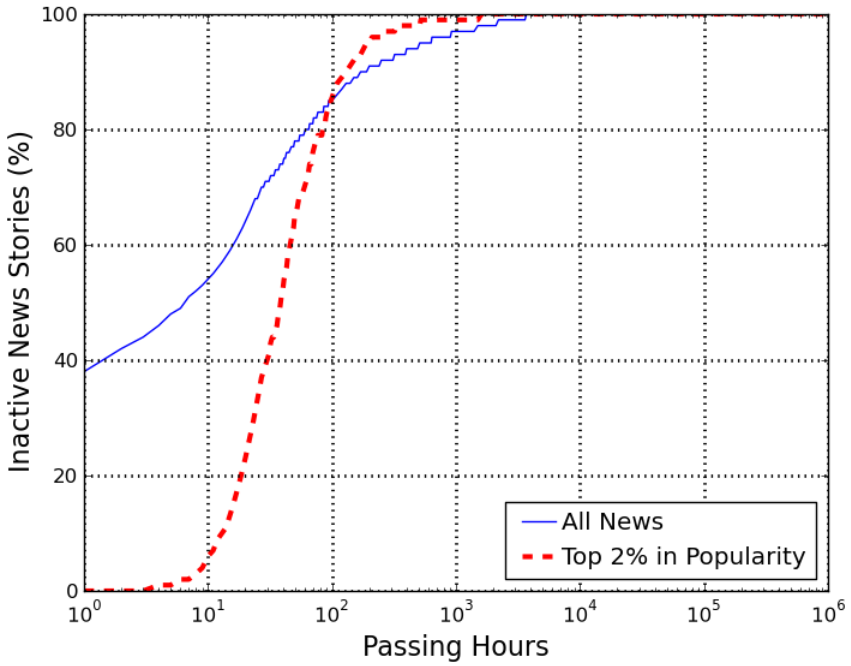

Figure 3.2: Longevity of News-Tweet Threads

popular tweets only.[4] The solid line refers to all tweets and the dashed line refers to the top 2% popular threads. For example, the solid line at 10 hours is at 54%, indicating that 54% of news-tweet threads have longevity of 10 hours or less. The solid line starts at 38% because 38% of news articles appear only once on Twitter and die upon birth.

From Figure 3.2 we see that about 85% of tweets lose public interest in four days, even among very popular news tweets. Furthermore, less than 1% of news articles survive more than a month. This result strongly suggests that using a one month period for data censoring (Section 3.2.2) is reasonable because if a news article does not appear on Twitter for 30 days it is unlikely to have appeared earlier or to appear later.

We also note that for popular news, less than 1% of news articles have longevity of less than 4 hours. That is, people are still interested in more than 99% of top news 4 hours after it first appears on Twitter. This statistic will be important later when we discuss the expert selection process.

[4]The distribution of, say, the top 1% (or 5%) tweets are roughly the same as that of top 2% tweets.

This does not mean that lots of users have bizarre tastes than others. As we study the source of publishing tweets in Section 3.2.3.3, we learned that many news blogger set up automatically publishing services and it turns out that all blogs they write by default publish to Twitter. It becomes a self-promoted behavior then, and possibly these articles themselves are not interesting enough to being spread out.

And ten hours or less seems to be a good choice to conduct polling because most of top news are still active, meaning that only good taste users may consistently report popular news.

Indeed that some articles, for instance, *Is Sugar Toxic?* [5] by Gary Taubes, was tweeted or retweeted for months. An popular essay style article may live for a long period, but only l

3.2.3.3 Examining the Sources of Tweets

How do people publish a news tweet? Do they mostly press the "Tweet Button" on a news article that they want to share? Or do they manually write a news tweet using other mechanisms? To get an idea, we now investigate the sources of news tweets from our dataset, i.e., the client by which a tweet is created.

Our analysis indicates that among 2,837,026 news tweets we collected, 24% are retweets, and 76% are not. Clearly, retweets are created when a user presses the retweet button in a Twitter client, so the source breakdown (of retweets) roughly tells the popularity of each twitter client.

Figure 3.3 shows the statistics for retweets, all "original" tweets (i.e., non-retweets), and the original tweets in the top 2% popularity to see whether there is any difference. The horizontal axis of the graph is labeled with sources of news tweets. Each source is associated with three bars that show the results for, from the left to the right, top 2% original tweets, all original tweets, and all retweets, respectively. For example, the

[5] http://nyti.ms/pXgxcV

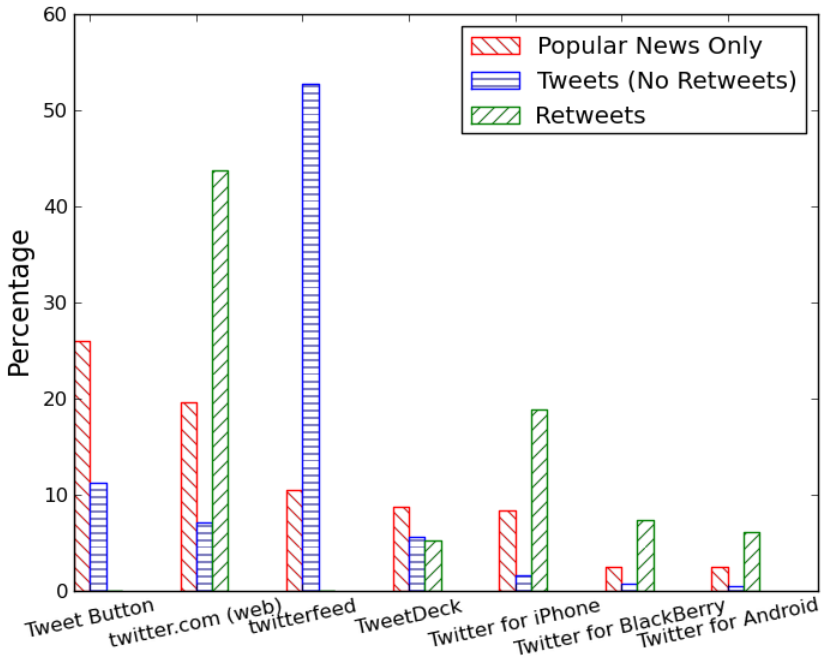

Figure 3.3: Breaking Down Source for All Tweets

left most bar for "Tweet Button" indicates that 25% of the top 2% tweets are created through a "Tweet Button," indicating that this is a major source of news tweets.

As expected, from the graph we see no retweet coming from the source Tweet Button. Retweets are coming from popular Twitter clients, such as Twitter.com, Twitter for iPhone, and Twitter for BlackBerry. What we did not expect is that 53% of *all* original tweets are created from `twitterfeed.com` (the second bar of twitterfeed category), a service that automatically publishes a tweet based on a blog RSS feed. On the contrary, among the top 2% popular tweets, `twitterfeed.com` plays a significantly smaller role (only 10%).

Since `twitterfeed.com` is an auto-posting mechanism that involves no human activity in creating a tweet and thus may not represent any human "endorsement" or "recommendation" of the news, we considered removing `twitterfeed.com` tweets, expecting that this may minimize "noise" in our dataset. But it turns out that either

removing or including `twitterfeed.com` tweets does not make much difference in our results. This is mainly because most `twitterfeed.com` tweets are not widely circulated on Twitter any way, so they are automatically removed from our later-stage analysis as it will be clear from our later discussion. All our results are based on a dataset that *does* include `twitterfeed.com` tweets.

In fact, in some cases it helps are actually helpful in identifying news experts. Digging into the data manually, we found some possible explanations: (1) if many users feed one blog to their Twitter accounts, the blog reach a large audience from the beginning, and (2) often only very high quality blogs lead to many subscriptions of their RSS feeds. Essentially both explanation suggests auto-posting mechanism like `tweeterfeed`, though it sounds counter-intuitive at first glance, is actually a signal of wisdom.

3.3 Experts and the Crowd

The primary goal of our study is to investigate whether we can identify a group of "experts" who can discover and recommend "interesting" news early. In this section, we formalize important notions relevant to this goal.

3.3.1 Interesting News: Golden Set

We first formalize what we mean by "interesting" news. Clearly, the interestingness of a news article is a subjective notion, which may elude any formal definition. Despite this subjectivity, previous research [GKL08, LH10] suggests that quality is often positively correlated with popularity, whose usefulness is repeatedly vindicated by the success of many popularity-based ranking metrics [BP98, Kle99]. Therefore, in this study, we choose to measure the interestingness of a news article by counting the number of tweets in the corresponding news-tweet thread. This count may be viewed as positive "votes" on the article on Twitter.

More precisely, we sort news articles by their tweet counts, and select the top $k\%$ news with the highest counts as the "interesting" news set or the *golden set*. Later, this set will be used to evaluate the recommendation performance of any group, including an expert group and the crowd. That is, when a group somehow recommends many news in this set, we consider the group to show high recommendation performance. More details on our evaluation metric will be given later. Next we discuss how we select an expert group.

3.3.2 Expert Selection Criteria

How can we identify an expert among all Twitter users? What are the distinguishing characteristics of experts? Intuitively, an expert in news recommendation is someone who can discover and recommend *interesting* news *early on*. This intuition leads us to two important criteria for expert selection: *precision* and *promptness*.

1. Precision: Precision is a well-known metric in the IR community. In our context, precision measures the likelihood for a news article to be "interesting" if it has been mentioned in a particular user's tweet. More precisely, given our definition of golden set, we define the precision of a user's news recommendation as

$$precision = \frac{\#\ of\ recommended\ news\ in\ the\ golden\ set}{\#\ of\ all\ recommendations\ by\ the\ user} \tag{3.1}$$

For example, if user A created 10 news tweets and 9 out of 10 point to articles in the golden set, the user's precision is 90%. Clearly, we expect that an expert would exhibit high precision in their news recommendation.

2. Promptness: Another important aspect in selecting an expert is how early the user is able to discover interesting news. A user who simply keeps (re)tweeting outdated but well-known news may have high precision, but his recommendation is unlikely to be useful to others since everyone already knows about it. Therefore, being able to *quickly*

sense future trends is a virtue we are looking for.

Based on this intuition, we modify the definition of precision by introducing the *promptness threshold t*; in computing a user's precision, we consider only those tweets that were created quickly enough, that is, within t hours after the corresponding news article first appeared on Twitter.

Example 3. *A user tweeted 100 news articles, and 80 out of 100 tweets belong to the golden set. Among the 100 tweets, 60 were created within 4 hours after the the corresponding news first appeared on Twitter, and 40 out of the 60 tweets belong to the golden set. If we use the promptness threshold of 4 hours, then the user's precision is* $40/60 = 0.67$*, not* $80/100 = 0.8$*.*

Valuing promptness also bring us another virtue: our selection process becomes less vulnerable to users who may "game" the system by tweeting well-known popular news stories from the past to obtain a high precision score. In all our subsequent discussion, we refer to the *precision with promptness threshold* simply as precision.

3.3.2.1 Expert Ranking Model

While precision and promptness captures two important characteristics of experts, what if a user creates only one news tweet within the promptness threshold and it happens to be in the golden set? This user's precision is 100%, but it is easy to see why we may want to exclude her from an expert group. We do not have sufficient evidence to believe her "true" precision to be 100%. To alleviate this issue of "one lucky hitter," we consider three *expert ranking models*.

1. Frequency-threshold model. One simple way to address the one-lucky-hitter issue is to exclude a user from an expert group if her tweeting frequency is less than a certain threshold value f. More precisely, we rank all users by their *precision* and select the top $k\%$ users with the highest precision as experts, excluding a user from the group if

her tweeting frequency is less than the threshold value f. This exclusion ensures that all users in the expert group have created at least a certain number of news tweets.

2. F-score model. Another way to address the one-lucky-hitter issue is to rank a user not only by *how precise* her recommendations are, but also by *how many* good recommendations she makes. This can be done by using the well-known *F-score* metric, which combines both *precision* and *recall* in its formula. Recall is defined as:

$$recall = \frac{\#\ of\ golden\text{-}set\ news\ recommended\ by\ the\ user}{total\#\ of\ news\ in\ the\ golden\ set} \tag{3.2}$$

That is, recall measures what fraction of the news in the golden set have been discovered and recommended by a user. Therefore, a user who has recommended many interesting news will have high recall.

Then F-score combines both precision and recall as follows:

$$F_\beta = (1 + \beta^2) * \frac{precision * recall}{(\beta^2 * precision) + recall} \tag{3.3}$$

Here, β is a tuning parameter that adjusts the relative weights given to precision and recall. When $\beta = 1$, precision and recall are given equal weights.

In the F-score model, we select the top $k\%$ users with the highest F-scores as experts. Under this model, one lucky hitter is unlikely to be included in the expert group because of her low recall (and thus low F-score).

3. Confidence-interval model. Another way to look at the "one-lucky-hitter" issue is if a user has recommended only one news article, there is not enough data to reliably estimate her precision. In other words, there is large uncertainty in her precision when the sample size is small.

A natural way to capture this uncertainty is to use a *confidence interval*. For example, according to the *adjusted Wald one-sided confidence interval* [AC98, BCD99], if a user has recommended one article and it belongs to the golden set, the 95% confidence

interval of her precision is between 0.24 and 1. Note the large gap between the lower bound 0.24 and the upper bound 1 of this interval. It indicates high uncertainty in the estimate, even though 100% of her recommendation has been in the golden set.

Based on this observation, we select the experts as the top $k\%$ users with the *highest lower bounds*. This way, we can avoid including one lucky hitter in the expert group since her lower bound is likely to be low.

3.3.3 The Crowd

Given our definition of experts, we define the crowd as *all* Twitter users *except* those being chosen as experts in *any* expert ranking model. To give an example, when we set each expert group size to 2% of all users, because some users are selected as experts in more than one model, the crowd contains 96.1% users. We choose to exclude experts from the crowd, since this definition makes it easy to compare and contrast expert wisdom and crowd wisdom.

3.4 Experiments

In this section, we discuss the results from our experiments[6]. We start the discussion by describing how we used our data to select an expert group and how we evaluate the wisdom of each group.

3.4.1 Selection Set vs Evaluation Set

As we discussed before, experts are selected based on their performance, so expert selection requires data. For this selection, it is important to note that the data used for

[6]We provide experiment data and more results on our site – `http://oak.cs.ucla.edu/~chucheng/publication/emh/`

evaluation should *not* include any data used for expert selection. Otherwise, the expert group has an unfair advantage because it has "seen" part of evaluation data during its selection process. Therefore, we divide our collected data into two sets, the *selection set* and the *evaluation set*. The selection set contains all news-tweet threads that were born between September 1st and October 31st and is used to select an expert group and the crowd. All other threads, being born between November 1st and December 31st, are assigned to the evaluation set. This set is used to evaluate the wisdom of an expert group and the crowd.

3.4.2 Evaluation Metric: Precision-Recall Curve

In experiments comparing expert wisdom and crowd wisdom, each group is told to provide a list of recommendations of news articles. We created the list in the following steps. To start, based on our evaluation set, we count how many news tweets or "votes" each news article collects from each group within the promptness threshold (say, within 4 hours after the article is first tweeted). We then rank all news articles by the number of votes and form the recommended news list of the group by selecting the top n articles.

The performance of a recommendation list is captured by the precision-recall curve [RBJ89] that compares the list against the golden set (or the popular news set) of the evaluation set. In this section, we create precision-recall curve by increasing the number of selected articles n from one until all news articles in the list are included.

Note that the precision-recall curve has been widely used by researchers to evaluate quality of recommendations or retrieval systems [GS09, HKT04]. In general, when the size of recommended list increases, the recall increases because more popular news could be included; however, the precision decreases because we may introduce some articles not belonging to the golden set.

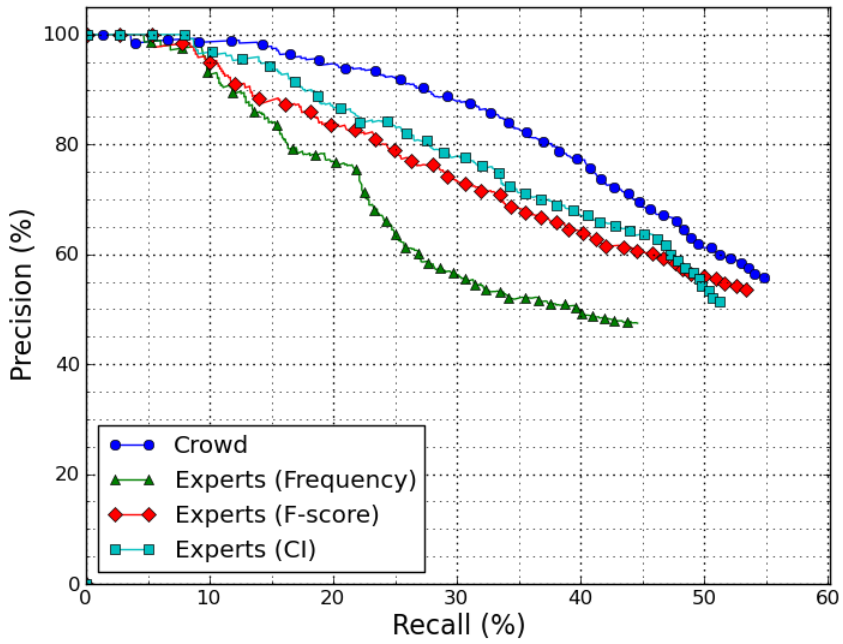

Figure 3.4: Wisdom Comparison (Promptness: 4 hrs; Top News Ratio: 5%; Expert Ratio: 2%)

3.4.3 Expert Wisdom vs Crowd Wisdom

3.4.3.1 Comparison

We now present our result comparing expert wisdom and crowd wisdom in Figure 3.4. In generating the graph, we use the top 5% popular news as the golden set, assigned the expert group size to 2%, and set the promptness threshold t to 4 hours. The results from all three expert ranking models are presented in the graph. For the frequency-threshold model (labeled as "Experts (Frequency)") we consider a user as a candidate expert only if the user is ranked in top 25% in her tweeting frequency. For the F-score model (labeled as "Experts (F-score)"), we set the parameter β to be 2. For the confidence-interval model (labeled as "Experts (CI)"), we use the 95% adjusted Wald one-sided confidence interval.

From the graph, crowd wisdom clearly outperforms the wisdom of all three expert

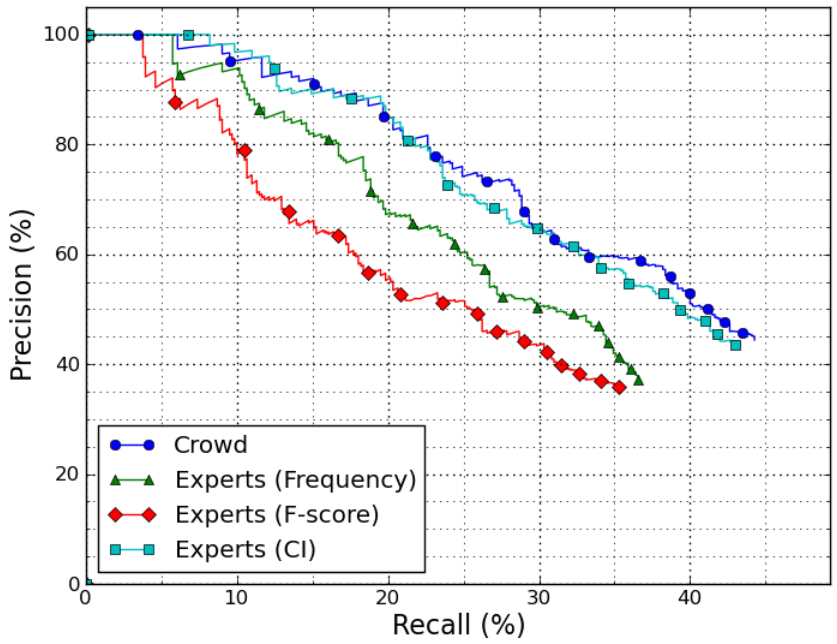

Figure 3.5: Wisdom Comparison (Promptness: 4 hrs; Top News Ratio: 2%; Expert Ratio: 2%)

groups. Among the three expert groups, the confidence-interval model performs the best. We repeated the same experiments for various parameter settings, trying various top news ratios (1%, 2%, 5%, 10%), expert ratios (1%, 2%, 5%, 10%), and promptness thresholds (1 hour, 2 hour, 4 hour, 8 hour), but the conclusions were the same. In all our experiments, (1) we could not identify a case when an expert group consistently outperformed the crowd, and (2) the confidence-interval model performed best among three expert models, which mostly performed worse than the crowd but showed competitive performance in a few cases. For example, Figure 3.5 shows another result with different parameter settings, where the crowd performs best and the confidence-interval model shows equivalent, or slightly worse, performance compared to the crowd.

So what makes the crowd wisdom distinctive? Both the crowd and experts seem to recommend articles that are more formal and informative, such as the article *Navigating*

Love and Autism[7], which chronicles the challenges that austistic children exerience as they grow up. However, articles that are less formal, shorter, and personal may still evoke resonance among average users. An example is the widely circulated article *Confessions of a Tweeter*[8], which journals why the author decided to stop using Twitter. This article was ranked in 14th position in our golden set, in 52nd position by our crowd, but in 620th on average among our expert groups. Our investigation seems to suggest the crowd are more reactive to the later type of articles.

Moreover, we see that the confidence-interval model achieves compareable performance with much fewer users, so who are they? We manually examined the list of experts, and noticed that a majority of accounts could be categorized into the following two main types:

1. Accredited human editors/writers: These professionals make their living by writing/editing news and are likely to dedicate their Twitter accounts to the same purpose. Also, their expertise may empower them to have a better sense of trends than average users. For example, we see *Paul Krugman*, *Larry OĆonnor*, and *Roberta Pennington* rank among the top experts.

2. Automated accounts based on crowd-sourced ranking: These accounts are designed for reporting quality news through crowd-sourcing. When certain criteria are satisfied, the account automatically posts a tweet. For example, the account `@newsyc50` tweets news if it is "voted up" more than 50 times on *Hacker News*[9]. The account `@NYTimesEmailed` tweets news based on the count of forwarded news via email.

[7] http://nyti.ms/rMxGxY
[8] http://nyti.ms/uf3Rcy
[9] `http://news.ycombinator.com`

3.4.3.2 The Source of Crowd Wisdom

Formally, we may consider our popularity-prediction problem as follows: We want to predict the set of news articles that obtain the largest number of recommendations over a long period (two months in our experiments) by *sampling* user recommendations within a time window of t-hour promptness threshold ($t = 4$ in earlier graphs). Cast this way, crowd wisdom corresponds to a brute-force sampling strategy where we sample recommendations by *all* users (except experts) made in t hours and expert wisdom corresponds to a biased-sampling strategy where we sample recommendations from a carefully-selected small subgroup. In general, the larger the sample size, the better the prediction accuracy, but the hope is to compensate for small sample size with a careful selection of the sampling subgroup.

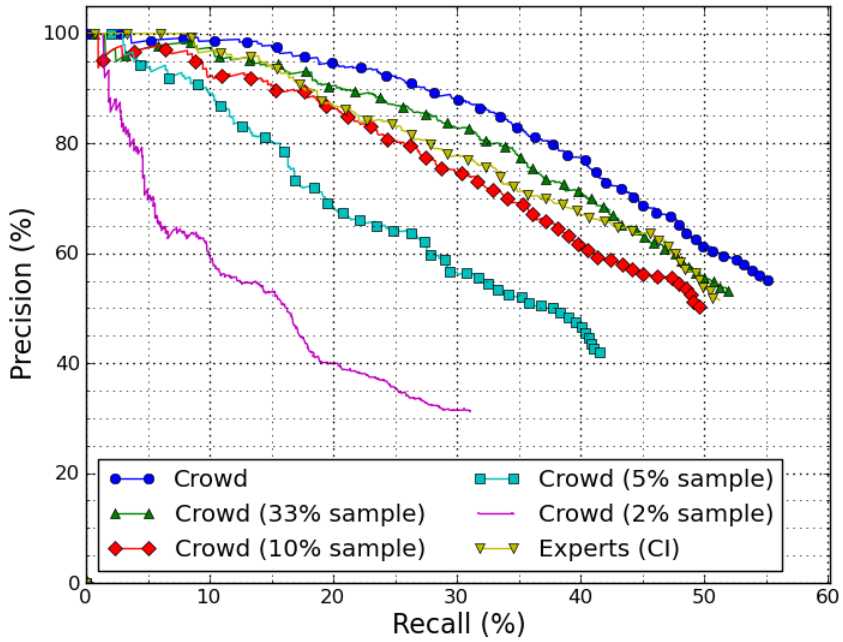

Figure 3.6: Crowd Wisdom After Random Sampling (Promptness: 4 hrs; Top News Ratio: 5%; Expert Ratio: 2%)

In our experiments, our three biased sampling strategies (i.e., three expert ranking models) were not able to overcome their small sample size. The sheer size of the

samples collected by the brute-force sampling strategy (i.e., crowd wisdom) makes it outperform the three biased sampling strategies. To validate this interpretation, we plot the precision-recall curve when the size of the crowd is made much smaller. We conduct random sampling on users to form small crowds (33%, 10%, 5% and 2%) and show changes on the precision-recall curves in Figure 3.6. For comparison, we also include the precision-recall curve of the the best expert group, the confidence-interval model, in the figure.

As we expected, we see a decrease in performance in the precision-recall curve when the size of the crowd decreases. We also observe that we get diminishing returns as the sample size increases. In this case, there was not much performance improvement after the crowd size exceeds 33%.

Finally, we note that the 2%-crowd curve, which has the same number of users as the expert model, shows significantly worse performance than the expert model. That is, biased sampling does lead to a better result when the sample size is the same. It is just that the sheer scale of the samples collected by crowd wisdom cannot be easily compensated for by a good sampling strategy. The expert model shows equivalent or better performance up to 20%-user crowd size.

3.4.4 Domain Experts Performance

One might suspect that experts fail to beat the crowd because being an expert in all news topics is extremely difficult. In this section, we further explore the scenario in which only one news category is considered at a time.

We first limit the competition to only one news category “international politics”, so that identifying knowledgeable persons to be experts in a constrained domain is more realistic. We keep only the news-tweet threads related to international politics, and then repeat the experiment.

In Figure 3.7 we present the result from this experiment, where the golden set is

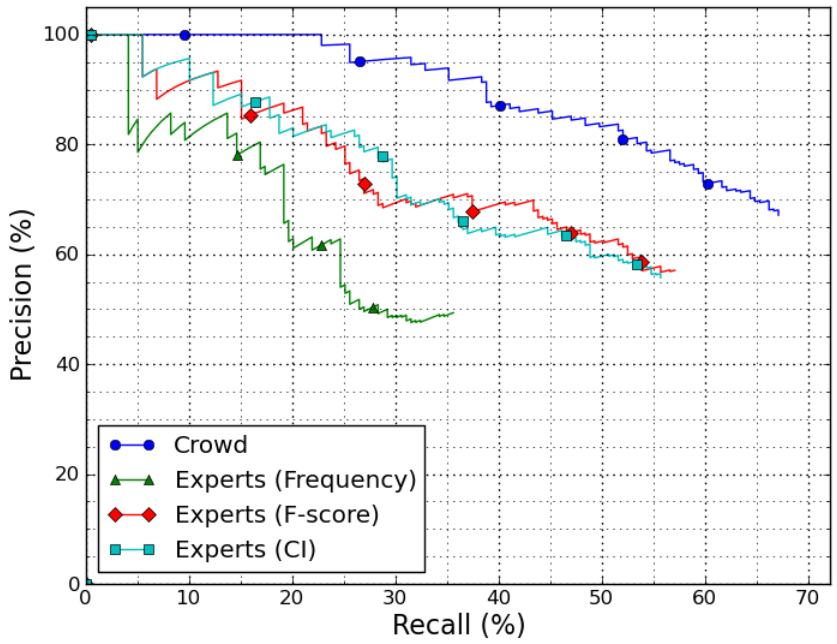

Figure 3.7: Wisdom Comparison - International Politics (Promptness:4 hrs; Top News Ratio:10%; Expert Ratio:2%)

defined to be the top 10% popular news in the international-politics dataset.[10] Other parameter settings are the same as in the previous experiment. From the graph, we observe a similar result — the crowd wisdom outperforms any wisdom from expert groups.

Unfortunately, the results from our topic-constrained experiments were inconsistent. We repeated similar experiments on other topics (sports, technology, science, etc.), and we find that constraining the comparison into some topics occasionally benefits expert groups. For example, in Figure 3.8 we present the result based on the topic "sports" with the same settings as we used in the international-politics experiment. We believe this inconsistency is partly because of small dataset size by the way we collected tweets due to limited resources. For example, we saw less than 5% of users in the crowd participate in tweeting sports news. Thus if we limit our discussion to the

[10] Since the dataset is significantly smaller than our overall dataset, we had to increase the size of the golden set, so that it contains a reasonably large number of news articles.

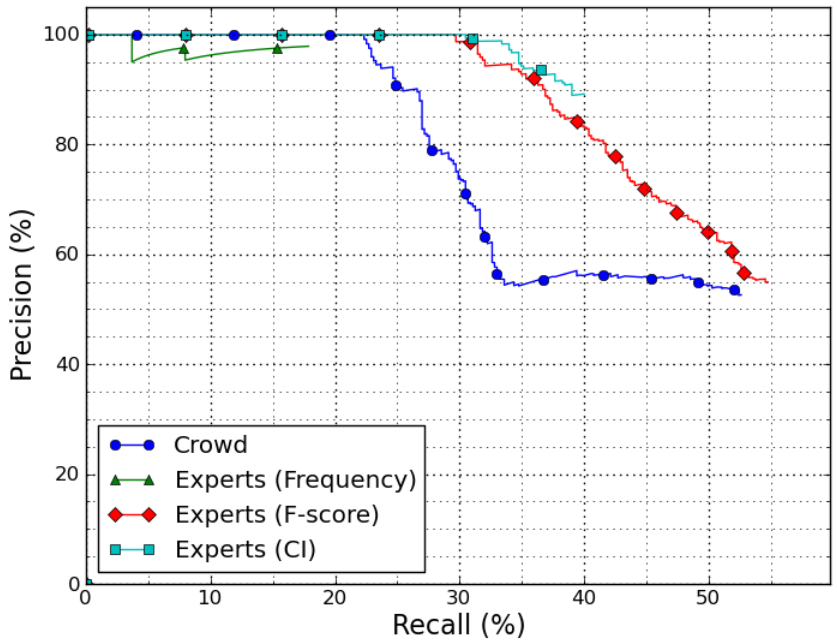

Figure 3.8: Wisdom Comparison - Sports (Promptness:4 hrs; Top News Ratio:10%; Expert Ratio:2%)

topic "sports", our data might be insufficient to lead to a verdict.

For example, note that the graphs from the frequency-threshold model and the confidence-interval model in Figure 3.8 are significantly shorter than others. This is because in the evaluation set, the expert groups made only a few recommendations, so their recommended news list contained only a few articles, leading to significantly lower maximum recall. Further study is necessary to verify whether expert models tend to perform better in topic constrained settings.

3.4.5 Augmenting Crowd Wisdom

In all our earlier experiments, crowd wisdom outperformed expert widom. Can crowd wisdom not be beaten? Is there any way to further improve crowd wisdom? In this section, we investigate possible strategies to improve crowd wisdom.

3.4.5.1 A Mix-up with Expert Wisdom

If individual experts show high precision, it may be still wise to investigate whether we can somehow leverage expert wisdom to "improve" crowd wisdom. As one possible way to mix up all wisdom, we leverage the expert-group precision observed in the selection set.

For example, suppose we observed that an expert group had 100% precision in predicting the top 20 stories in the selection set. Then, in the evaluation set, suppose that a news article is recommended by the crowd in 140th position and by the expert group in 10th position. In this case, even though the article is ranked relatively low by the crowd, we may still want to include it in the recommendation list because we have sufficient evidence from the expert group that it is an interesting news (Recall that all top 20 news from the expert group turned out to be interesting in the selection set). More formally, we assign an estimated "precision-score" to every recommendation based on precision-recall curves learned in the selection phase. We then mix up all recommendations, and form a new recommendation list by sorting their estimated precision-scores.

Figure 3.9 shows the precision-recall curve when this "precision boosting" by the expert group is applied (labeled as "Mixed"). From the graph it is clear that performance gets better with boosting. The curve with boosting always lies above the crowd curve (labeled as "Crowd").

3.4.5.2 Removal of Talkative Users

From the investigation of our dataset, we found most users tweet quite infrequently. For example, our statistic shows that about 75% users in the crowd tweeted less than once a month on average, The other 25% of users are quite talkative. An example of talkative users is the official Twitter account of *The New York Times*(`@nytimes`), which tweets more than twelve times a day. Do those talkative users provide useful information (wisdom) to the crowd? In general, we suspect they may be a source of noise. Since

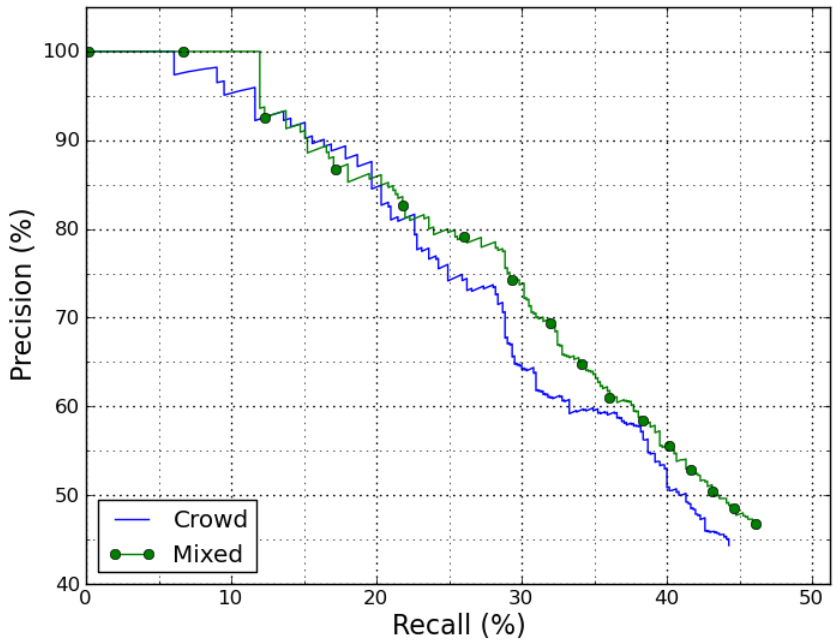

Figure 3.9: Mixed Wisdom (Promptness: 4 hrs; Top News Ratio: 2%; Expert Ratio: 2%)

these users tweet news articles indescriminately, many of their recommendations are not likely to be interesting, which will introduces noise to crowd recommendations.

To evaluate this hypothesis, we sort all users based on their tweeting frequency in our selection set, and select the bottom k% users with the lowest tweeting frequency as the *inactive crowd*. We then compare the wisdom of the inactive crowd against our regular definition of the crowd.

Figure 3.10 shows the result from this experiment, where $k = 75\%$. When the top 25% of users (sorted by frequency of news tweets) in the crowd are removed, we see an improvement in performance. Through deeper investigation of the news recommended by these "inactive" users, we also find that they are more likely to be circulated among Twitter for a long time. For example, a widely circulated article "Is Sugar Toxic?[11]" was recommended by many of these "inactive" users. Likely these users are very se-

[11] http://nyti.ms/pXgxcV

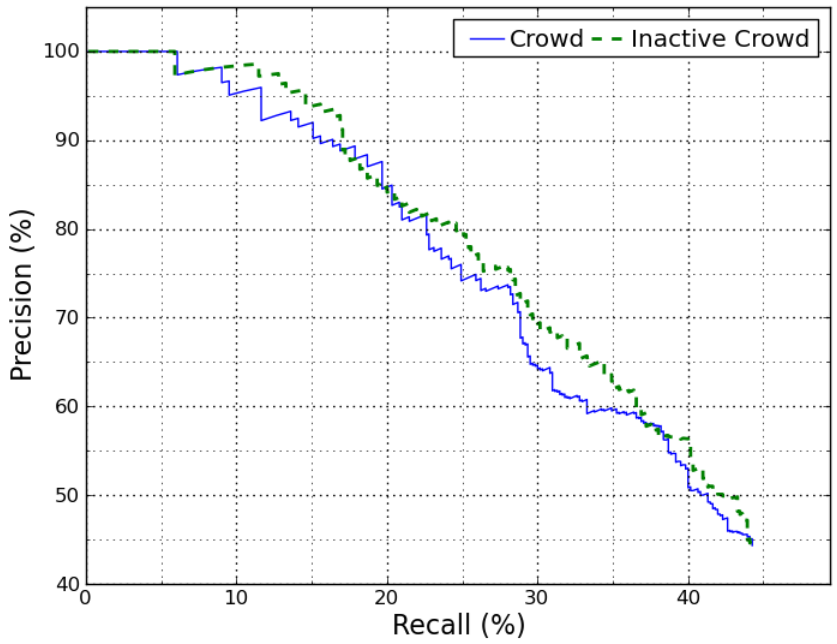

Figure 3.10: Inactive Crowd Wisdom (Promptness: 4 hrs; Top News Ratio: 2%; Expert Ratio: 2%)

lective in their recommendations, and thus when they concur (by tweeting) on a news article, the article is likely to be interesting and resonate longer amongst the Twitter community.

3.4.5.3 Combining Dual Strategies

Figure 3.11 shows what happens when we apply both strategies simultaneously. As shown in the figure, combing them leads to better performance. For example, in the combined strategy, the precision stays at 100% up to a recall value of 13%, which is a signifiant improvement from the performance of the regular crowd.

Note that assuming the cost of monitoring a user is identical, collecting tweets from experts first is efficient resource-wise. Although we bisect users into experts (2%) and the crowd (98%) for comparison and investigations in this study, to recommend potentially interesting news (i.e. future popular news) in practice, a simple strategy is

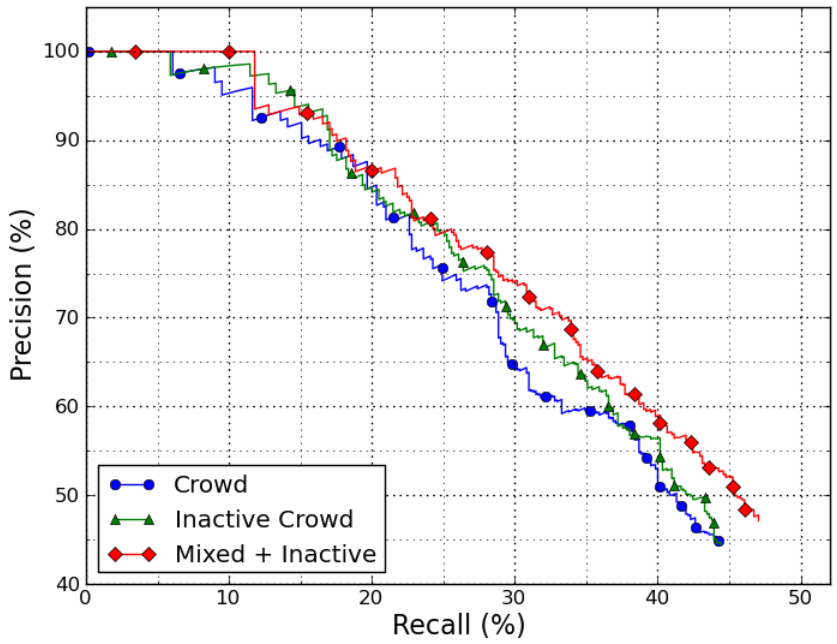

Figure 3.11: Augumenting Crowd Wisdom (Promptness:4 hrs; Top News Ratio:2%; Expert Ratio:2%)

to allocate resources based on expert weightings, and collect as many users as one can afford.

3.4.6 Future works: Individual Influence and Group Size Bias

3.4.6.1 Individuals with Powerful Influence

Hitherto we have been neglecting the discussion about a user's influence and we selected expert groups purely based on their past performance. This is mainly because we use popularity as the evaluation metric. Under this metric, if we consider a user's influence in selecting experts, it may unfairly bias our evaluation since a news article's popularity is affected by the influence of the users who recommend it. That is, we may indirectly use the evaluation metric itself in selecting the expert group.

The influence of a user has been shown to be an important factor that affects the

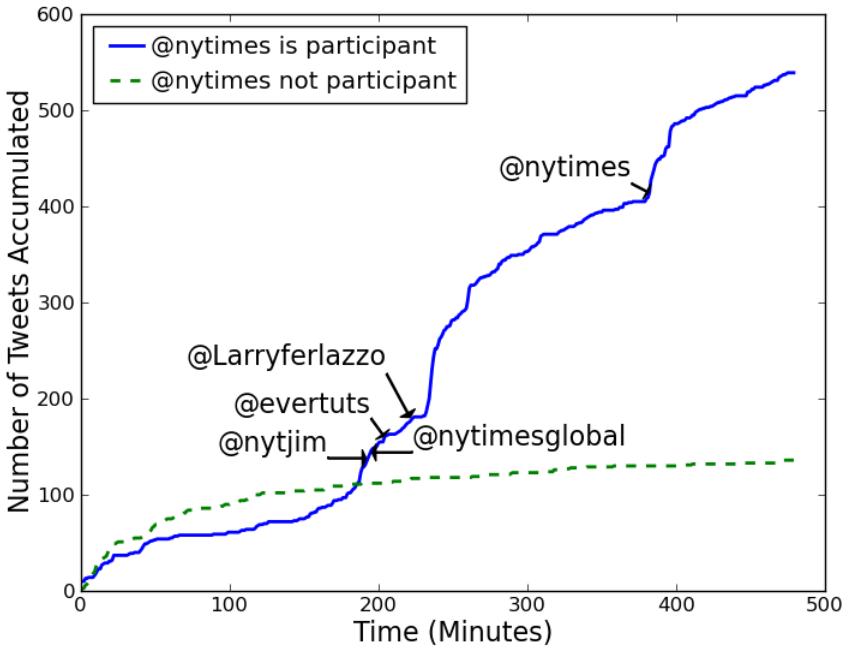

Figure 3.12: Tweets Accumulation Over Time

spreading of news [BHM11, CMP11], and ultimately affects the popularity of the news. We use Figure 3.12 to discuss our observations about a user's influence on the popularity of a news tweet thread. In this figure, of two news stories we draw the number of accumulated tweets over time after seeing the first tweet. One story, represented by the solid blue line, was retweeted by some famous twitter accounts. The other (the green dashed line), is mostly tweeted or retweeted by average users.

As shown in Figure 3.12, the user `@nytimes` (with 4,617,318 followers) tweeted the story at the 380th minute, and right after the user's tweet, we observed a sharp increasing of the number of tweets. Such a super influential user as `@nytimes` greatly augments the size of audience and therefore leads to the surge in the number of upcoming tweets. In addition, it is possible to see a sharp increasing after a few influential users like `@evertuts` (a Portuguese user with 150,557 followers) tweet the story.

For practical purposes, however, we believe a user's influence may be useful for expert selection and would like to investigate it further in the future.

3.5 Related Research

News recommendation on Twitter: Researchers [KLP10, LKF05, RMK11] have found that micro-blogging website like Twitter help circulate information. Recent studies [DBA10, PMS09, MSP12] show that Twitter data is a great source for identifying popular topics, trends, and breaking news; Kwak et al. [KLP10] reported 85% of hot topics on Twitter are actually headline news.

Researchers have shown success in detecting headlines or breaking news, either through analyzing activities of an individual or streaming data from a crowd. Some work is based on a user's profile or persona. For example, Morales et al. [DGL12] proposed how to provide personalized headlines. Some makes recommendations though analyzing trend activities (Twitter streaming data). For example, Teevan et al. [TRM11] demonstrated how to user Twitter to find temporally relevant information like breaking news, and Petrovic et al. [POL10] demonstrated how to identify the first news story about a news event. Similar to outcomes of those excellent works, our study also produces a list of news recommendations, but we have a different focus – we see news recommendation as a process of decision making in a group, and put emphasis on their precision and recall in predicting future popularity.

Common people vs experts: Researchers across multiple disciplines have been interested in exploring whether a group of common people could make better decisions than a competent individual over a long time. Psychologists like Hill et al. [Hil82] argued that the ability of a group in problem solving does not always surpass the most competent individual. However, Frederking et al. [FN94] found that a three person team demonstrated better reasoning skills and were more effective in solving math problems than the most competent individual. Our study pursues the same direction, with the goal of expanding the size of competition. In those works, people were interested in knowing whether a small group was better than an individual, but we are interested in knowing whether many heads are better than a few.

The debate "whether experts make better decisions" also took place in the finance domain, especially after Fama et al. [Fam70] introduced the Efficient market hypothesis(EMH) in 1969. Hundreds of follow-up studies were conducted to support or oppose the hypothesis. We are also interested in knowing whether the market (the entire crowd) is beatable. In this study, we take a different approach, gathering experts as one group against the crowd. We are more interested in knowing the synergy of many experts as opposed to one expert.

Crowd wisdom on Twitter: The emergence of social network websites has brought the usage of crowd wisdom to the next level, because researchers have found that crowd wisdom has great potential for prediction [CDS10], summarization [MWL12], and recommendation [YLW12]. Recently, Meng et al. [MWL12] used Twitter data, proposing an opinion summarization framework for entities. Yin et al. [YLW12] demonstrates the possibility of taking advantages of early votes to improve the ranking of poplar items. Ma et al. [MZL11] shows that social network information can benefit traditional recommender systems. Work by Cataldi et al. [CDS10] detected emerging popular topics in the world based on Twitter. The ongoing success of harnessing crowd wisdom does not resolve the aforementioned debate.

Researcher have started investigating the role of experts in a crowd. Kittur et al. [KPS07] examined experts vs common users to see which party contributes more wisdom over time on Wikipedia. Bakshy et al. [BHM11] investigated influential individuals (to some extent, experts) to see their roles in helping spread information. And recently, Ghosh et al. [GSB12] mined Twitter List information for recommending topic experts. Our work advances investigation along the same direction, further exploring the wisdom of experts and seeking the possibility of exceeding the crowd with experts.

3.6 Summary

In this chapter, we explored whether a group of experts can make better decisions than a much larger group of ordinary people. Through studying user tweeting activities, we identified two desired characteristics of experts (precision and promptness) and discussed strategies for selecting experts based on these characteristics. To measure the quality of group decisions, we conducted experiments on Twitter and compared the wisdom of groups by gauging their abilities in discovering and recommending popular news articles early. We draw a similar conclusion to the that of the efficient-market hypothesis: crowd wisdom exceeds expert wisdom in predicting future events. Nevertheless, we also find that the voice from expert groups can be used to improve crowd decisions if used carefully and removal of the voice from overly talkative users boosts the quality of crowd decisions.

CHAPTER 4

Motivating Online Shoppers with Advantageous Query Suggestion

4.1 Introduction

These days, several enhancements have been made to web pages in an effort to improve the web searching experience and to also help users effectively express their needs. Providing users with *query suggestions* is one of the most known approaches. We commonly see query suggestions implemented in two scenarios: (1) as an auto-complete list underneath a search box, and (2) as a list of related queries on the search result pages, which gives the users ideas for the next search.

Query suggestions have been extensively studied in the past. One popular strategy[CJP08, **?**] is based on clustering similar queries into groups, and another is based on co-occurrence analysis[HCO03], i.e., examining whether two queries often appear in one session. Both clustering queries and maximizing co-occurrence probabilities work towards the same goal: helping users to effectively express their intentions.

Today, search interfaces are provided everywhere, and one of the most notable applications is to aid online shoppers searching for goods to buy. Although an online shopping scenario can resemble a web page search scenario, researchers[HPS11] have begun to investigate the differences. For example, in a shopping scenario we may not want to suggest queries that lead to products that are currently out-of-stock.

The value of a suggestion varies between different perspectives. From a website

owner's perspective, some "related queries" are more valuable than others, especially if they have financial impact. Similarly, from a shopper's perspective, some queries are more inspirational or educational than others (e.g. reminding a user to consider a screen protector after purchasing a smart phone). In our work, we study algorithms aiming to satisfy both perspectives at the same time.

We evaluate our work by conducting experiments on eBay[1] data, one of the leading online marketplaces in the world. Nevertheless, our algorithm can be applied to any search interface designed for shoppers. In other words, an online retailer can be considered as a special case of marketplaces in which only a few sellers (perhaps as little as one) are allowed to participate in transactions.

We summarize our main contribution as follows:

- Our work empowers a website owner to describe the owner's interest. By associating every query with some estimated value (Section 4.2.3), our algorithm incorporates the interests of the owner to select query suggestions. (Section 4.4.1)

- We propose an algorithm that can be set to identify query expansions (specialization reformations), e.g. "*coach* $\rightarrow$ *coach handbags*", which is suitable for generating an auto-complete list, or to identify query substitutions (parallel-movement reformations), e.g. "*coach* $\rightarrow$ *Louis Vuitton*". (Section 4.5.2.1)

- We discuss the possibility of promoting high value query reformations for ambiguous queries (e.g. "*dress*" or "*toys*") by displaying visualized query suggestions on top of our algorithm. (Section 4.5.2.2)

[1] `http://www.ebay.com`

User ID	Time Stamp	Event	Value
User 1	20120501070813	QUERY	coach
User 2	20120501070818	QUERY	iphone white
User 1	20120501070973	VIEWITEM	99998176
User 2	20120501071078	EXIT	0
...			

Table 4.1: A search log example

4.2 Problem Formulation

We start our discussion by introducing our terminology, followed by an explanation of how we summarize logs into a transition graph. Then, we formulate the problem by introducing the notion of value estimation.

4.2.1 Definitions

In our study, we target an online marketplace, and assume that we have access to completely anonymized query logs. Table 4.1 shows an artificial example.

4.2.1.1 Session

Websites track visitor activities and store them as offline logs. Naturally, records in a log can be grouped by user ID. Often search logs can be divided into smaller pieces, namely, sessions. Each user session begins when an event is recorded on the corresponding user and ends when an "EXIT" event is encountered, which is either an idle timeout or logout event.

Previous research[CJP08] sometimes assumes that one session corresponds to one intent. However, for online shopping scenarios, it is common to see multiple intents in

a single session. For example, a smart phone purchase can inspire the need to search for a *screen protector*. Therefore, we do not limit a session to have only one intent, and intent shifting is accepted and considered in our model.

4.2.1.2 Conversion

We use the symbol q to denote a query and the symbol e (together with some subscript) to denote an action executed by a user. For example, the symbol $e_V(q)$ (subscript V for "VIEWITEM") indicates that we see a user browse a product page (i.e. an "action") and the last query the user issued in front of this event is the query q. For example, the record with the value 99998176 (some page identifier) in Table 4.1 shall be denoted as the action $e_V(q = coach)$. Thereby, logs in Table 4.1 can be converted into the following form:

$session\ 1 : q_1 =$ coach $\rightarrow e_V(q_1) \rightarrow \ldots$

$session\ 2 : q_2 =$ iphone white $\rightarrow e_X(q_2)$

(other sessions . . .)

To reduce computational complexity and to focus on generalizations, we discard the choice of having event values as "differentiators" to actions, e.g. 99998176 in the above example. Instead, we assign some related query to every action event, so our conversion naturally records the relation that a user found something interesting after issuing a query, say, because the user then browsed a product page ($e_V(q)$) or purchased a product($e_B(q)$).

Furthermore, associating an action with its accredited query helps us track cause-and-effect relations in general. For instance, we would expect to see that "$q = screen\ protector$" follows the action $e_B(q = iPhone)$ in quite a few sessions if buying a screen protector is often motivated by an iPhone purchase. In other words, the query $screen\ protector$

Symbol	Meaning
e_B	Making a purchase based on the list price. (direct purchase)
e_O	Making a bid or a bargain to an auction. (negotiation)
e_W	Tracking an item by putting it to the watch list. (bookmark)
e_V	Viewing a product page.
e_X	Log out or idle for a long period. (an exit)

Table 4.2: A List of tracked events

is not caused by an individual item purchase, but is likely a common pattern for all iPhone purchases.

Differentiating between the same kinds of actions based on event values (especially, some management identifier like "*page id*") can be inefficient and annoying. A management identifier is usually transient, meaning that it disappears quickly before we can obtain sufficient information for analysis.

Dealing with logs from a online marketplace exacerbates the problem. Firstly, the supply of an auction can be very limited, especially for used goods. Secondly, one product sold by two users must be associated with different page identifiers for further management. Finally, the on-sale period is often constrained (to only a few days) in practice, and reposting a sale creates another identifier.

For simplicity, we use $F(q)$ / $F(e_V(q))$ to denote in a dataset, all possible successors (queries or actions) that have ever appeared in the next position of a query q / an event $e_V(q)$ in some user session(s), respectively. Table 4.2 provides a summary of major

actions (symbols and their meanings) being analyzed in our experiments. Ideally if resources are permitted, all activities that matter to interested parties should be recorded and investigated.

4.2.2 Transition Graph

Now we discuss how to construct a directed graph to consider transitions among queries and events from logs. Every vertex in the graph corresponds to a query or an event (including an "exit" event), and every edge is associated with a number that signifies the probability of observing such a state transition. We use the following example to explain how we convert logs into a transition graph.

Example 4. *Consider a log dataset of three sessions, each of which corresponds to one of the following records:*

(1) $q_1 \to e_V(q_1) \to e_V(q_1) \to e_B(q_1) \to q_1 \to e_V(q_1)$
$\to e_X(q_1)$

(2) $q_1 \to e_V(q_1) \to e_V(q_1) \to e_O(q_1) \to q_2 \to e_V(q_2)$
$\to e_X(q_2)$

(3) $q_1 \to e_V(q_1) \to e_B(q_1) \to q_2 \to e_V(q_2) \to q_2$
$\to e_V(q_2) \to e_V(q_2) \to q_1 \to e_X(q_1)$

One may interpret Example 4 as follows. In record (1), some user searched for a query q_1, and in the search result she browsed two product pages. Then she made an offer to the seller associated with the second product page. She issued the search q_1 again, viewed another product page, and then she left. In record (2) & (3), two queries are in play. Note that actions are always associated with the nearest query in the front, and there is only one exit event in the end of a session.

We draw three independent transition graphs based on the above three records respectively, as shown in Figure 4.1. Under the hood, a session is first broken into pairs, e.g. in session (1), it becomes a sequence of pairs "$q_1 \to e_V(q_1)$, $e_V(q_1) \to e_V(q_1)$, ...,

$q_1 \rightarrow e_V(q_1)$". Then, based on the count of pairs, we then compute transition probabilities between two vertices. Every edge is labeled with a transition probability and (inside parentheses) its count of appearance(s) in a session.

To convert several sessions in a dataset to a graph, we first break all sessions into pairs. Each pair consists of a "from" vertex, say x_F, and a "to" vertex, say x_T. The transition probability from x_F to x_T is then calculated by dividing the counts of the pair "$x_F \rightarrow x_T$" by the total counts of pairs where their "from" vertex is x_F. As a result, given a "from" vertex x_F, the sum of its transition probabilities to other vertices will be 1.

We now introduce a man-made example (containing all sessions in a dataset) as presented in Figure 4.2. All queries in the example are fabricated, and so are the transition probabilities. This example is used to highlight characteristics of transition graphs we observed in our experiments (i.e. a real world scenario).

What is our interpretation of a transition probability? In Figure 4.2, we observed "$q = coach \rightarrow$ q=coach handbag" to be 20%. This means that in a query log dataset, twenty percent of the query *coach* is immediately followed by a query reformation *coach handbag*.

In practice, most transitions are bi-directional between two query nodes, and a *specialization* transition (e.g. $coach$ to $coach\ shoe$) is likely to have a higher transition probability than its reverse. Furthermore, we may spot some very low (but non-zero) transition probabilities between two non-related nodes such as from the node $coach$ to the node $mouse\ pad$. Although a shopper's change of intent is unpredictable, the chance that two users happen to share the same next move is very low.

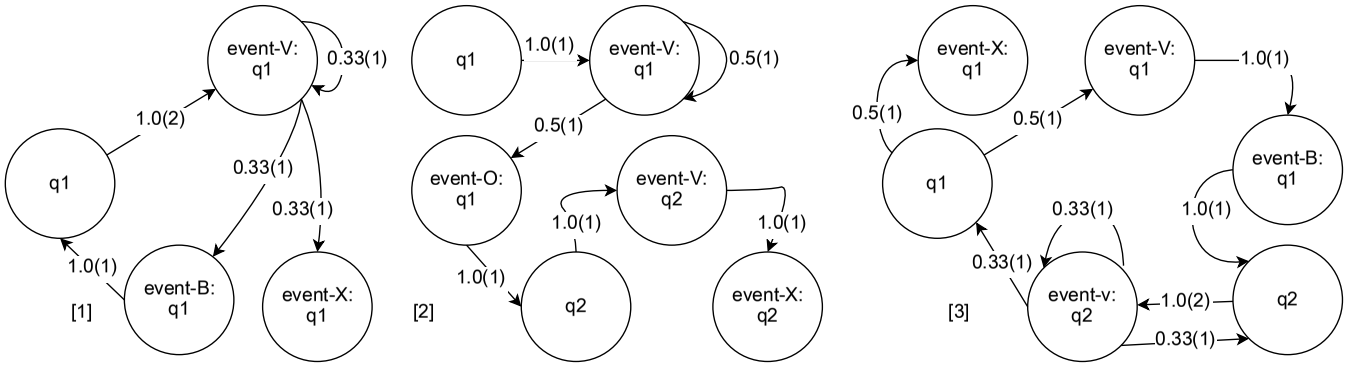

Figure 4.1: Transition Graphs from Individual Sessions (Example 1)

Furthermore, all EXIT actions (denoted as **event-X** in Figure 4.2) become sinking nodes if performing a random walk. In other words, sinking nodes are those with no out-going edges. Lastly, it is worth mentioning that the graph G is not *strongly connected* in practice because some queries might be unreachable from another query due to the sparsity of query logs.

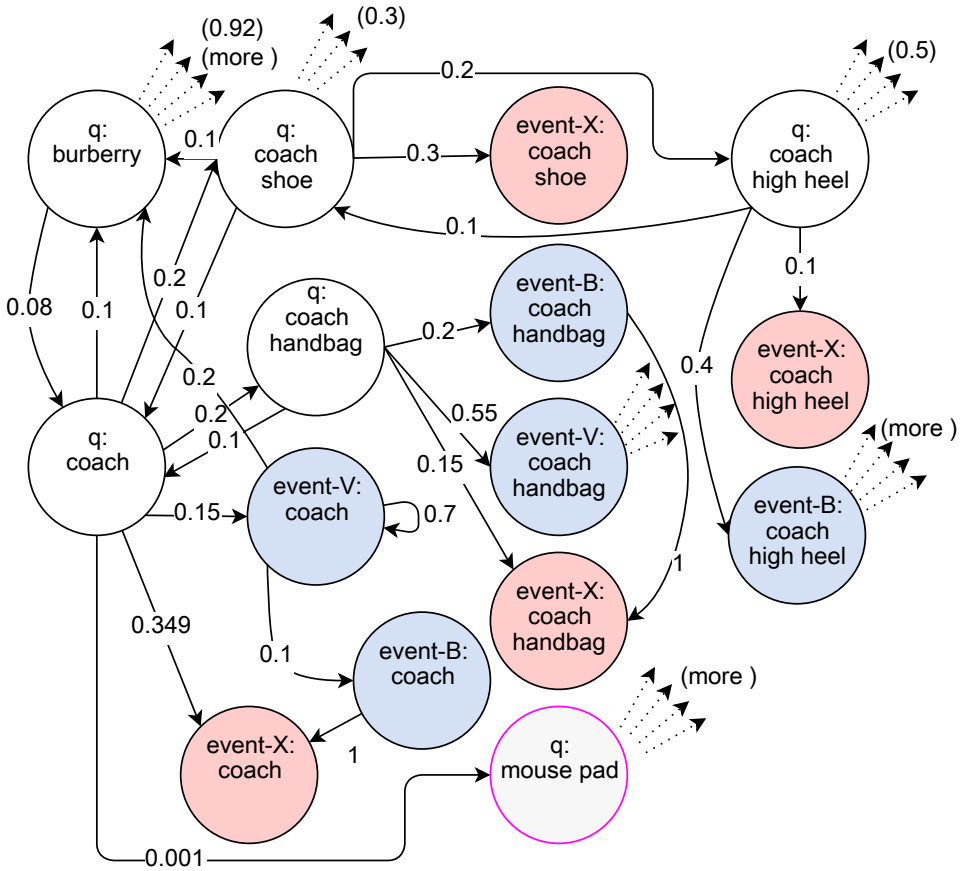

Figure 4.2: Transition Graph (Integrated All Sessions)

4.2.3 Value Estimation Function

We use the symbol $V(x)$ to represent the estimated "value" associated with the node x in a transition graph. That is to say, the symbol $V(e_V(q))$ represents to the estimated "value" associated with the action $e_V(q)$, and similarly the symbol $V(q)$ represents the value of the query q.

What is $V(x)$ (the value of a node x)? The value consists of two parts: the direct (assigned) part, denoted as $V_D(x)$, and the indirect counterpart, denoted as $V_I(x)$. Namely,

$$V(x) \sim f_V(V_D(x), V_I(x)) \tag{4.1}$$

, where the symbol f represents some unknown function.

For the direct part, it is some value assigned to a node based upon the agreement of all interested parties. For instance, if the goal is to maximize the number of sales, one might assign e_B and e_O (in Table 4.2) nodes with some values and set the rest (of nodes) to zero.

For the indirect part, it is some accumulated value associated with future graph transitions in accordance with related visiting probabilities. Suppose that a user is viewing a product page. There exists some possibility that the user make a purchase in her next (or future) moves. For example, in Figure 4.2, we see that 10% of $e_V(coach)$ events are followed by a purchase, and this fact conveys an intuition that $V(e_V(coach))$ should be non-zero. We consider corresponding values from the future, as shown in the following:

$$V_I(x) \sim f_I(V(x_1), V(x_2), ...|\{x_1, x_2, ...\} \in F(x)) \tag{4.2}$$

Essentially, the value function becomes a customized measurement of satisfaction to interested parties. That is to say, $V(q)$ indicates the degree of expected satisfaction when a query q is issued. One can see the value function as some objective to pursue, and the underlying idea of our work is to provide query suggestions that aligns with the objective, which is defined by carefully planing the direct (assigned) value to each state.

We can see that varying goals will influence how values are assigned. For example, users aiming to maximize a financial goal might have assigned values that are proportional to the profits generated from sales. On the other hand, if the goal is about maximizing user retention (a user experience oriented goal), a non-transactional action such as page browsing will be play a larger role in determining the assigned value.

4.2.4 Goal

The intuition underlying our work is to boost the estimated value $V(q)$, by increasing $V_I(q)$, i.e. the value from future moves, through the "promotion" of some query suggestions, denoted as $S(q)$, in front of or somewhere easily visible in the search results. That is to say,

$$V_I^*(q) \sim f_I^*(V(x_1), V(x_2), ...|\{x_1, x_2, ...\} \in F(q) \cup S(q)) \tag{4.3}$$

The goal of our work is to identify a set of query suggestions, denoted by $S^*(q)$, such that $V^*(q)$ (the value of a query) is maximized when $S^*(q)$ is displayed to users. In other words, we are lifting $V^*(q)$ by increasing $V_I^*(q)$, the estimated value from future transitions when $S^*(q)$ is displayed.

To compute $V_I^*(q)$, we ask the following two questions: (1) how likely is a shopper going to click on our query suggestions and (2) how much do we expect to gain from the click? The first question reflects the consideration on query relevance — we should recommend queries ($r \in S$) that are as relevant to the query q as possible. The second question reflects the tendency to recommend valuable query reformations that maximize a website owner's objectives.

Problem Statement 1. *Given a graph G, a query q, and a number k (representing the expected number of query suggestions), identify a list of "queries": $\{r \in S^*(q)\}$, such that $|S^*(q)| = k$ and $V^*(q)$, the estimated value function of q, is maximized.*

If $V^*(q) < V(q)$, naturally the query suggestions should be avoided. In other words, we should consider displaying the suggestions only when the "expected benefit" is greater than its loss.

4.3 Model

In this section, we explain how to interpret transition graph G as a Markov model, and then explain how to estimate the value of a query based on a hop-limited random walk process. We end this section by introducing some approximation strategies for accelerating the calculation in practice.

4.3.1 Markov Chain Interpretation

The way we construct a transition graph G naturally leads it to a Markov model interpretation. Every node (query or event) becomes a state in a Markov chain, and every edge is associated with a transition probability between two states.

Markov Chain Properties: In a transition graph G, any node referring to the "EXIT" action is in an *absorbing* [BT02] state (p. 337) because it is impossible to escape from it. On the contrary, except sinking nodes (i.e. "EXIT" actions), every other node in a *transient* [BT02] state (p. 322) since there is a possibility of never returning to it.

Every session in our logs eventually ends at an "EXIT" action. In other words, at least one absorbing state is accessible no matter which state we start from. Namely, the Markov chain is *absorbing*; if we perform a random walk in a transition graph G, eventually the walk should stop at some absorbing state.

Transition Probabilities: As explained in Section 4.2.2, when constructing the transition graph, we normalized co-occurrences in sessions into transition probabilities. Naturally all transition probabilities are greater than zero.

Our transition graph G (containing m states) can be encoded into a transition probability matrix T, a two-dimensional $m \times m$ array in which the element $T[i, j]$ is the probability of transitioning from the state i to the state j. Given the state i, the sum of all its outgoing (including self-pointing) transition probabilities is equal to one, i.e.

$$\sum_{j=1}^{m} T[i,j] = 1.$$

To compute the 2-step probability from the state i to the state j, we compute the matrix product $T \times T$, and the element $T^2[i,j]$ then represents the 2-step transition probability from state i to state j. In a similar fashion, T^n corresponds to the n-step consequence.

Note that our model is absorbing, which means that when $n \to \infty$, only the absorbing states (i.e. "EXIT" actions) have non-zero values. This matches the intuition that all user sessions ends eventually.

4.3.2 Hop-limited Random Walk

When a Markov chain is absorbing, the probability of absorption in some state $e_X(q)$ is conditional on the initial state. For example, in Figure 4.2, the likelihood of being trapped at the state $e_X(coach\ handbag)$, an exit action, is higher if a random walk starts at the query *coach handbag* than at the query *coach*. Similarly, the initial state also affects the probability of visiting a transient state. For the same reasons as before, we would expect a higher likelihood of visiting the state $e_B(coach\ handbag)$ if a random walk starts at *coach handbag*.

Since a random walk stops when it visits some absorbing state, the nodes it visited before reaching a stop are also subject to the initial state. For example, in Figure 4.2, the chance of a random walk visiting *coach handbag* (before reaching a stop) is much higher if the walk starts from *coach* than from *mouse pad*. We now use the fact that the visiting probability is subject to the initial state to determine $V(q)$, the value of query q.

Specifically, when repeating random walks starting at the query q (corresponding to the state i), we can obtain a specific distribution $\mathcal{A}(i,n)$ of m entries (m is the total number of states in a graph). Each entry in the distribution refers to, from the state i,

the average of visiting probabilities of bypassing a state j before it is trapped or reaches a pre-defined hopping limit, i.e. n hops. The distribution $\mathcal{A}(i,n)$ can be computed by using a transition probability matrix T, as presented in the following:

$$\mathcal{A}(i,n)[j] = \frac{\sum_{U=1}^{n} T^U[i,j]}{n} \tag{4.4}$$

, where the denominator, n, is used to ensure that the sum of all elements in $\mathcal{A}(i,n)$ is always one. In other words, $\mathcal{A}(i,n)$ conforms to the second axiom of probability. Thereby we may interpret the individual element in $\mathcal{A}(i,n)$ as some (visiting) probability.

Suppose the state i corresponds to a query q, the intuition underlying $\mathcal{A}(i,n)$ can be interpreted as follows: if a user starts a session with the query q, and we expect the user to execute n moves in this session, $\mathcal{A}(i,n)[j]$ is the indicator to measure how likely a state j (thereby, its correspondent $node(j)$ — a query or an action) appears in the user session.

Later in our experiments, we learn that when a user spots some unsatisfactory query results, what often happens next is likely some specification reformation if the input is a short query. A specification reformation, e.g. "$coach \rightarrow coach\ handbag$", is more likely to follow a query q because a short query often returns thousands of results. Thereby when $n = 1$, we observed that specification query reformations are associated with higher value in the distribution $\mathcal{A}$ than parallel movements. On the contrary, a parallel movement, e.g. "$coach \rightarrow gucci$", often happens in the later stages. When $n = 3$ or higher, parallel movements start lifting their values in the distribution $\mathcal{A}$.

Assuming that every state j is associated with an assigned value $V_D(node(j))$ (Section 4.2.3), we propose the following equation to compute the estimated value of a query ($V(q)$):

$$V(q) = V_D(node(i)) + \sum_{j=1}^{m} V_D(node(j)) * \mathcal{A}(i,n)[j] \tag{4.5}$$

The second term in Equation 4.5 should be viewed as our estimation of $V_I(q)$, the "indirect" value as presented in Equation 4.1.

In Equation 4.5, n acts as a parameter for tuning how we appraise future moves. When $n = 1$, only the "next" following moves matter. On the contrary, when $n = 3$ or 4, likely we incorporate the value from its next query reformations (aligning with their visiting probabilities). For example, when $n = 3$, we essentially consider this prominent pattern: $q \rightarrow$ (any event)$\rightarrow$ reformation $\rightarrow$ (any event).

4.3.3 Approximation and Acceleration

Usually a query log dataset is very large. When converting logs into a transition probability matrix T, if the dimension of T is very high, operations on T can be expensive, especially if T cannot fit into memory. Now we discuss some possible approximations for reducing the dimension of T.

It is common to observe some random intent shifts in practice. For example, in Figure 4.2 the refinement *mouse pad* follows the query *coach* with some non-zero probability. We consider these edges as a type of noise which can be removed. Thus, one may assign a transition threshold θ_E, say, 0.2%, to eliminate these edges.

Following the same fashion, we introduce an occurrence threshold θ_N, eliminating rarely-seen queries (or typos) and their events to reduce the matrix dimension. We believe transition probabilities associated with these nodes are unreliable because in statistics, observing only few occurrences implies our estimation of an unknown population parameter (i.e transition probability) is inaccurate.

To ensure our graph satisfies a Markov chain after elimination, the probability associated with these eliminations are reassigned to "event-X" ($e_X(q)$). For example, in Figure 4.2, the transition probability $q = coach \rightarrow e_X(q = coach)$ increases to 35% if the node $q = mouse\ pad$ is removed from the graph.

Note that, with these thresholds, a log dataset is more likely to be converted into

disconnected (and thereby smaller) transition graphs, where each graph is represented by an individual transition probability matrix of lower dimension than the original matrix. Also, the above elimination strategies focus on improving performance rather than improving correctness. We aim to remove edges and nodes that are likely either to be noise or at least insignificant.

4.4 Query Suggestions

In this section, we describe how we determine the query suggestions to display. Two strategies are proposed: the first is based on incorporating both query relevance and value improvements, and the second introduces a tuning parameter to control the importance of high-value choices.

4.4.1 Promoting Valuable Queries

To estimate the value improvement brought by displaying some query suggestions(s), we divide the estimation into two parts: and (1) the value improvement (difference) between the query and its reformation and (2) the probability of a shopper clicking on the query suggestion. We use the multiplication of the above two parts to rank query suggestions.

We may assess a value to the first part by defining the value improvement as the difference of their value estimations. That is, with the help of Equation 4.5, we consider the value improvement to be $V(r) - V(q)$, where q stands for a query and r stands for the reformed query being clicked.

After the transition from q to r, the expected value becomes $V(r)$ (previously $V(q)$). As a result, the difference signifies an improvement (or a decrement) on values if we assume the cost of a user clicking on r in q's search result page, i.e. transition expense, is negligible.

To assess the second part, a recent study has reported that with regards to web search, the click-through rate for query suggestions that are displayed next to the search box is on average, about 7% for single-term queries and 4.5% for multi-term queries. The study also suggests that many other factors such as the ambiguity of the query will also affect the click-through rate, which makes the value difficult to predict.

Our belief suggests that the click-through rate of r for q should be proportional to the likelihood of a shopper issuing some query r after issuing a query q, also known as a co-occurrence tendency. Intuitively, if r frequently follows q in the past, when we display r as a query reformation, its click-through rate is likely to be higher than other less frequently connected queries. Specifically, we define co-occurrence tendency as the sum of visiting probabilities over random walks (Equation 4.4). In other words, when r is displayed, its click-through rate is positively related to the likelihood of observing r following q in user sessions (i.e. a visit).

Suppose for a moment that only one query reformation will be displayed to the user. Based on our definition above, we can say that a "best-click-through-rate" (best-CTR) scenario occurs when we display a query reformation (state j) which corresponds to the highest visiting probability. It can also be seen as the selection of r that will maximize the co-occurrence of q and r in a single user session.

We then measure $V^*(q)$ (Equation 4.3), i.e. the estimated value of the query q when a query reformation r is displayed, with the following equation:

$$\begin{aligned} V^*(q) &= [1 - \mathcal{CTR}(r)] * V(q) + \mathcal{CTR}(r) * V(r) \\ &= V(q) + \mathbb{C} * \frac{\mathcal{A}(i,n)[j]}{max\ \mathcal{A}(i,n)[j^*]}[V(r) - V(q)] \end{aligned} \tag{4.6}$$

, where $\mathbb{C}$ stands for the click-through rate in the best-CTR scenario.

The above equation can be interpreted as follows. When the most frequent r is displayed as a suggestion, we expect to have $\mathbb{C}\%$ clicking through. Otherwise, the click-through rate is normalized with respect to the difference in their frequencies, namely, $\mathcal{A}(i,n)[j]/max\ \mathcal{A}(i,n)[j^*]$.

Note that in the above equation, for a given q, both $\mathbb{C}$ and $max\ \mathcal{A}(i,n)[j^*]$ are constant. Therefore, to identify the best query reformation candidates to q, we sort every r based on

$$\mathcal{A}(i,n)[j] * [V(r) - V(q)] \tag{4.7}$$

in descending order, and report the top k queries. The first term in the equation reflects the consideration of click-through rates (based on co-occurrence tendency) between q and r, and the second term refers to the "benefit" of having r if a click happens. Naturally, if $V(r) < V(q)$, we should avoid displaying suggestions because a negative effect on shopping experience is expected.

So far, our strategy is developed based on the assumption that only one query reformation is displayed. In other words, we do not consider the challenge of diversifying query suggestions[AGH09] nor clustering similar query suggestions[SMW10] into one. Due to page length constraints, we invite interested readers to consider viewing an online appendix[2] for more information on our on-going research in this area.

4.4.2 Advertising Factor

While we believe that the click-through rate of a query reformation candidate is positively correlated to its co-occurrence tendency to the target query, some previous research[Kou02] has reported another observation: clicking behavior on a query suggestion may be also correlated to some factor other than a shopper's original intention. The observation of impulsive shopping behavior[Ste62], i.e. unplanned purchasing, is a classical example.

Although the cause is inconclusive, it is possible that a shopper clicks on a query suggestion for other reasons (but query relevance). Here we are interested in considering a scenario where we may promote high value items when we believe a shopper is open to our advice. Specifically, for queries like "*toys*" and"*gifts*", we believe that a shopper's intention is not a particular product, but some direction (i.e. anything reason-

[2] `http://bit.ly/ZAejMO`

able will probably be considered). In such a case, a website owner may want to place more weight on promoting high value reformations to increase profit.

As a result, we introduce the notion of *advertising factor*, denoted by d, and revise Equation 4.7 to:

$$\{d * \mathbb{A} + (1 - d) * \mathcal{A}(i, n)[j]\} * [V(r) - V(q)] \tag{4.8}$$

, where $\mathbb{A}$ can be assigned as "$max\ \mathcal{A}(i, n)[j^*]$", or simply some constant.

Originally, we proposed d in order to consider impulsive shopping behaviors. That is, our query suggestion algorithm should promote more valuable reformations if, for a query, it believes a shopper to be relatively open to advice. However, in practice d can be used for a website owner to decide how aggressive the service provider would like to promote high valuable reformations to a query. When $d = 0$ (very conservative), the ranking algorithm tends to recommend choices based on co-occurrence tendency. On the contrary, when $d = 1$, our algorithm (naively) assumes that relevance does not matter and thus the ranking is dominated by how much value improvement a query reformation is associated with.

4.5 Experiment

We conduct experiments for various queries based on user session data gathered from daily logs of eBay U.S.'s website portal. To respect eBay's policy on data privacy, we will disclose the results of our algorithms mainly on three queries: *coach*, *toy*, and *iphone*. These are very popular queries issued by eBay visitors every day. Each one corresponds to a dataset that includes user sessions where the query appears at least once (case insensitive).

These selected datasets are important because they represent a wide variety of challenges. A query like *toy* is a broad concept. On the other hand, *iphone* refers to a specific product. Finally, *coach* is a general term but is also a brand name, meaning

that it can be viewed as a filter constraint so that only products sold by Coach, Inc. are considered.

4.5.1 Setup

4.5.1.1 Data

Each of these datasets is generated based on a random sampling process from user logs across a 4 week period. To avoid violating eBay's data policy, both the sampling ratio and all user information (including user ids) are not disclosed, and each dataset is created based on a different sampling ratio. However, the number of provided user sessions is enough to examine our hypothesis and to prove our concept. The *coach* dataset, *toy* dataset, and *iphone* dataset contain 185232, 364474, and 487938 user sessions, respectively.

	$q = coach$		$q = toy$		$q = iphone$	
	$n = 1$	$n = 4$	$n = 1$	$n = 4$	$n = 1$	$n = 4$
(1) Equation 4.7 (Co-occurrence + Estimated Value):						
1	coach handbags	dooney	toys lot	ford	iphone 3gs	iphone 3gs
2	coach wallet	michael kors	baby toys	tractor	iphone 4 case	blackberry
3	coach purse	coach wallet	lot	toys lot	iphone motherboard	iphone broken
4	coach wristlet	kate spade	farm toys	lot	iphone case	iphone charger
5	coach ashley	cole haan	toy tractor	watch	iphone 3gs unlocked	htc
6	coach sunglasses	louis vuitton	spongebob toys	dress	iphone charger	iphone 4 case
(2) Equation 4.4 (Co-occurrence Only):						
1	coach handbags	coach handbags	lego	ford	iphone 4	iphone 4
2	michael kors	michael kors	baby toys	lego	iphone 4s	iphone 4s
3	coach wallet	coach wallet	sex toy	baby toys	iphone 3gs	iphone 3gs
4	coach shoes	gucci	boy toys	sex toy	unlocked iphone	ipad
5	burberry	burberry	vintage toys	vintage toys	blackberry	blackberry
6	gucci	louis vuitton	toys lot	boy toys	ipad	iphone unlocked

Table 4.3: Query Suggestion Results

4.5.1.2 Configuration

Query Clean-Up: Before we construct a transition graph, we go through several clean-up steps. To start with, we convert all queries into lowercase, effectively considering them to be case insensitive. Next, we keep only alphanumeric letters and "negative symbols", i.e. all other symbols are dropped out. We keep negative symbols because in most search interfaces it represents a **NOT** operator. For example, the query "*iphone white -case*" indicates a user is asking to return search results not containing the keyword "*case*".

The previous process preserves negative symbols so that we can further use it to refine our granularity. Specifically, we discard keywords leading with a negative symbol. For example, the query "*iphone white -case*" query would be converted into "*iphone white*". Naturally, this setting greatly reduces the number of states in a transition graph since we merge several queries, where the only differences are keywords with NOT operators, into one.

Assigned Value($V_D(node(j))$): To compute the value of a query (Equation 4.5), direct values ($V_D(node(j))$) to every node j in a transition graph must be assigned.

For the initial setting we choose to examine, we hypothesize that all queries and exit events ($e_X(q)$) are of no value, and for events of the same type, we always assign the same score for simplicity.

Our intuition suggests that in the real world an e-commerce website would value earning financial profits the most. For an auction website, its profit is mainly affected by the sum of transaction amounts, which dominates commissions. However, we are unable to acquire details on commissions due to our data usage policy, and thus in our experiments, we evaluate our work with the following assumption: we place the most value (6 points) on purchasing ($e_B(q)$), and the least value (1 point) on "page-viewing" ($e_V(q)$). We assign the rest (e.g. putting a bid or adding to a user's watch list) 2 points.

Finally, we consider an exit action as a neutral move, which receives a score of 0.

At our initial setting, the score assignment is purely based on the event type, so we are unable to favor high-priced items even though their sales will generate higher commissions. Ideally we can incorporate correlated commission information to produce a better result. Overall this initial setting is based on a naive assumption that all products (and their sales) are equally valuable. Later on we will discuss the effects of changing our point assignments.

Robustness: Naturally, our log files contained some noisy information. For example, there were instances where a very long session had several different, unrelated queries. Instead of adding an additional cleanup step for our data, we adopt the following principle: when breaking a session into transition pairs, e.g. "state $i \rightarrow$ state j" (as discussed in Section 4.2.2) we count the same pair only once per user session.

4.5.2 Empirical Evaluation

4.5.2.1 The Initial Assessment

For each dataset we present the top 6 suggestions generated by our algorithm (Table 4.3). Note that because most websites only have enough room on the page to show a few related queries, the top results from any query suggestion algorithm will matter the most.

Table 4.3 contains two sections. The first section shows to the top results based on Equation 4.7, which incorporates the consideration on estimated query reformation value and visiting probabilities. The second section refers to query suggestion results based on Equation 4.4, which is essentially purely visiting probabilities.

Default Category: It is worth mentioning that the eBay website displays a predicted category for queries. For the query $coach$, the default (dominant) category is "*Clothing,*

Shoes & Accessories". Similarly, for the query toy, the default one is "*Toys & Hobbies*".

Our further investigation on logs show that the reformation "*ford*", ranking in the top when $n = 4$ in both sections of the query "*toy*", is actually constrained by the same category (*Toys & Hobbies*). The search "ford in *Toys & Hobbies*" leads to a result containing many die-cast car toys. As a general rule, when interpreting or using our query suggestions presented in this section (all tables), one has to assume that, when a query reformation is fired, the dominant category assignment stays unchanged.

Specification vs Parallel Movement: To begin with, we focus on section (2), a recommendation strategy based on only co-occurrence.

Previous research[BBC09b, KST12] suggests that there exists two types of query suggestions: *specialization*, e.g. from "*toy*" to "*boy toy*", and *parallel movement*, e.g. from "*coach*" to "*gucci*".

When a shopper spots an unsatisfactory search result which contains various types of matched items, naturally, an intuitive next move is to refine the query to narrow down the search scope, especially if q return thousands of items. Therefore when $n = 1$ (i.e. one hop away from a query q), specialization reformation is expected.

In contrast, after some exploration (i.e. a few query reformations) a shopper may start to consider alternatives or competitors, e.g. $coach$ to $gucci$, or vice versa, if her need is still unsatisfied. As a result, when n is increasing, it is more likely for a random walk to land on alternatives (i.e. parallel movements).

We identify an expected trend when looking at the query *coach* in Table 4.3: when $n = 1$, the algorithm tends to recommend specialization reformations; in contrast, when $n = 4$, it tends to recommend parallel-movement reformations. In a shopping scenario, these parallel movements often refer to competitors, e.g. *michael kors* to *coach* or *blackberry* to *iphone*.

However, for some queries it is irrational to expect parallel movements, especially

if the query is a broad concept like *toy*. In these situations, we expect that specialization reformations are favored over parallel movements since there are no clear alternatives to the given query. When comparing the results of the query *toy* in section (2) (when $n = 1 \rightarrow 4$), we find that the results follow this intuition; we observe that five out of six suggestions (specialization) remain the same.

Comparing Section (1) and (2): For all three queries, when $n = 1$, the reformed query contains the original query, meaning that a specialization has occurred. When $n = 4$, a competitor or alternative is introduced. Although the query *toy* refers to a broad concept with no distinct alternative, reformations like *dress* and *watch* are still promoted. Simply put, when $n = 4$, we believe that our algorithm encourages distinction by promoting non-specialization transformations, i.e. reformations not containing the original query.

In the column where $q = coach$ and $n = 4$, we observe more alternates. We have competing brands to $coach$ such as *dooney*, *kate spade*, and *cole hann* that sell similar products but at a cheaper price. More expensive brands such as *gucci* and *burberry* are pushed farther down in the ranking. When looking at the column for $q = iphone$, we can see that our algorithm suggests queries that include keywords like $cases$, $charger$, and $broken$. These keywords are typically associated with cheaper accessories or auctions which may lead to a larger number of sales.

In principle, if the score assignments of products are to accurately reflect their prices, then luxury brands such as $gucci$ and $burberry$ will have a higher rank. However, because we are unable to access detailed commission and pricing information, we have to base our value estimates on the number of sales for a product instead.

Auto-complete Suggestions vs Related Queries: Modern search interfaces often provide query suggestions in two places: (1) popping up a auto-complete list when user are typing the query and (2) displaying a list of related queries in the search result page.

Our work is well suited for both applications.

Our work in Section (1) has shown that when $n = 1$, query reformations are specializations that will preserve the original query. This works well with the auto-completion scenario. When $n = 4$, our process begins to suggest alternate keywords for users to explore, improving parallel mobility. This is best suited for the second scenario.

4.5.2.2 Case Study: Aggressive Promotion

We now discuss how using the advertisement factor (Section 4.4.2) will impact our results. As previously explained, the factor d in Equation 4.8 determines how aggressively we will promote high value reformed queries. We believe that the value of d should be related to the ambiguity of the query in question. For example, when a user issues a broad query like "*dress*" or "*gift*", it is more likely that the user will be open to suggestions. Therefore, we can be a bit more aggressive and not run the risk of turning away the user.

We present a query suggestion example for the query "*coach*" in Table 4.4. In Table 4.4, as d increases, we begin to see reformed queries that fall under one of two patterns. Either the result has several keywords added (e.g. "*coach kyra signature tote*") or the result has a very specific keyword inserted (e.g. "*coach carly*", where *carly* refers to a specific style of handbags). Compared to the reformed query when $d = 0$ ("*coach handbags*"), these examples (when $d = 20\%$) are much more specific.

Intuitively there is a trade-off here: a very specific query suggestion has a higher chance of not matching the user's original intention. However, because its specificity greatly reduces the number of matched results (from 1 million to hundreds) and likely improves the overall quality on search results, a shopper is more willing to browse through each auction and make a purchasing decision.

Visualized Query Reformations: Previous research has reported that quite often shop-

pers will have some previously unplanned purchases[Ste62] or impulse purchases[ZPS07], especially when presented with visual stimuli[PVW09] (e.g. images). Therefore, when using an aggressive d value, we can increase the click-through rate by using images to create more of a "window-shopping" experience.

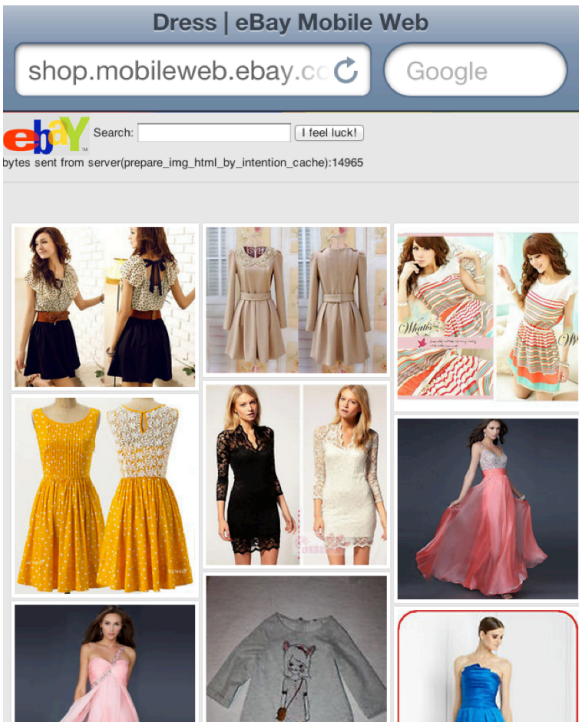

Figure 4.3: Visualized Query Suggestion

We explored this idea by creating a prototype[3] to demonstrate a possible use case for aggressive advertisement factors.

A screen shot (searching for "*dress*") of our prototype is presented in Figure 4.3. Under the hood, each query reformation is replaced with some product picture in its first search result. A click on the picture then rewrites the query.

[3]The prototype is for eBay internal demonstration only and is not open to public.

	$d = 0$		$d = 5\%$		$d = 20\%$	
	$n = 1$	$n = 4$	$n = 1$	$n = 4$	$n = 1$	$n = 4$
1	coach handbags	dooney	coach wallet	dooney	coach kyra signature tote	dooney
2	coach wallet	michael kors	coach handbags	coach pink	coach carly	coach sabrina
3	coach purse	coach wallet	coach purse	coach sabrina	coach signature stripe tote	coach pink
4	coach wristlet	kate spade	coach wristlet	michael kors shoes	coach wallet	michael kors shoes
5	coach ashley	cole haan	coach ashley	michael kors purse	coach soho	michael kors purse
6	coach sunglasses	louis vuitton	coach carly	coach ashley	coach ashley satchel	used coach handbags

Table 4.4: Query Suggestion Results (*q*=*coach*)

4.5.2.3 Case Study: Lingering vs. Immediate Action

We mentioned in Section 4.2.3 that query suggestions are affected by how a website owner assigns specific values to states. To further investigate this matter, we compare results between two artificial scenarios: a user-centric one in which the goal is to encourage a shopper to linger longer on a website, and an owner-centric one in which the goal is to urge shoppers to make a purchase immediately.

For the user-centric "lingering" goal, we set -1 to all EXIT events (i.e. $e_X(q)$) and the value 1 to all other actions, including queries themselves. Intuitively speaking, we penalize a reformation if it drives away a shopper (i.e. an EXIT event). For the owner-centric "selling" goal, we assign the value 5 to all purchasing events (i.e. $e_B(q)$), 1 to all bid / bargain events (i.e. $e_O(q)$), and set all the rest to zero, thus valuing only sale-related events.

We present our results when $n = 1$ and $d = 0$ in Table 4.5. We can see that (1) the query suggestions are significantly affected by value assignments, and (2) more parallel-movements are introduced in both settings than in the initial setting.

For the lingering goal, introducing parallel-movements is intuitive because a shopper is likely to consider additional search attempts along the lines of the newly suggested query. For the selling goal, we encountered many less relevant reformations because if the algorithm focuses on only "sales numbers" it tends to suggest irrelevant results. Although neither of these two man-made goals is ideal for an e-commerce website, their results justify the statement that the owner's goal is considered accordingly in our algorithms.

	$q = coach$		$q = toy$		$q = iphone$	
	lingering	*selling*	*lingering*	*selling*	*lingering*	*selling*
1	michael kors	coach wallet	lego	lot	ipad	cell phones
2	gucci	dkny	hello kitty	baby toys	blackberry	galaxy s ii
3	burberry	coach leather	angry birds	hello kitty	ipod	nokia
4	juicy couture	ralph lauren	lot	baby	ipod touch	iphone screen
5	louis vuitton	coach ashley	toys lot	dress	htc	iphone lot
6	kate spade	burberry	hot wheels	hot wheels	samsung galaxy s ii	ssd

Table 4.5: Query Suggestion Results (Lingering vs. Immediate Action)

4.6 Related Research

Query Log Conversion: Converting query logs into a Markov model for query suggestion tasks has been investigated before. One of the most relevant study to ours is the construction of a query-flow graph[BCD09, BBC09a]. The query-flow graph is a directed graph where nodes are queries, and edges are associated with some weight that indicates the strength of transition. Comparing to their work, our conversion focuses more on the user events, but not query transitions.

Query Suggestions: During the early days of search engines, Bruce et al. published a work[BD97] about query reformation. Since then, providing query suggestions for web users has been an active research topic. A popular strategy is based on clustering queries on top of logs of URL clicks [BHM05, CJP08]. Later on researchers[DLK09, MYK08, MZC08] conducted analysis on bipartite graphs (e.g. consisting of "query – clicked url" pairs) and incorporated the notion of random walks to model user behaviors.

Other related areas to query suggestions include query expansions and diversifications. Query expansions[CR12, XC96] are used to recommend synonyms and morphological forms to expand possible results. Although query expansions and parallel-movement suggestions both try to help users explore more, the latter focuses its attention on recommending competitors and alternatives. Diversification[MLK10, SZH11] puts effort on reducing redundancy, which can be viewed as a possible post-process or extension to improve query suggestion results.

Recently, researchers[BBC09b, KST12] have investigated query suggestions of specialization and parallel movements. While their work stressed on the distinction and presentation of both types, our work centers on e-commerce shopping scenarios and make no distinctions in terms of presentation.

Recommendations for Online Shoppers: Hasan et al.[HPS11] performed research

that also focused on providing query suggestions for online shoppers. However, their work targeted the challenge of long-tail query distributions and parallelizing their computation, while our work is centered around incorporating a website owner's goals into query suggestions.

4.7 Conclusion

Providing query suggestions is a challenging task that most search interfaces need to have to satisfy their users. In this work, we studied query suggestion strategies for helping shoppers. As search interfaces become a standard feature on e-commerce websites, we took into considering both the query relevance and the best interests of website owners. We formulated the problem as random walks on a transition graph based on a Markov process and proposed ranking algorithms that are suited for applications such as auto-complete suggestion lists, parallel-movement related keywords, and visualized query suggestions for windows-shopping users.

CHAPTER 5

Conclusion

In this dissertation, we describe algorithms, metrics, and analysis that can be used to improve the user search experience. Specifically, three scenarios are targeted: alternative choices, popular choices, and win-win choices.

In terms of providing alternative choices, we build a prototype website allowing users to search over all entries in Wikipedia based on tag information, and then collect 600 valid questionnaires from 69 students to create a benchmark for evaluating our algorithms based on user satisfaction. Our results show that the presented techniques are promising and surpass the leading commercial product, Google Sets, in terms of user satisfaction.

In terms of providing popular choices, after conducting repeated experiments, we draw a similar conclusion to the EMH – we could not identify an expert group whose news recommendation performance was consistently better than that of the crowd. Meanwhile, we study how to improve crowd wisdom further, and our analysis suggests two promising strategies: (1) "augmenting" crowd wisdom with expert wisdom when there is sufficient evidence and (2) reducing noise in crowd wisdom by removing "overly talkative" users from the crowd.

In terms of leading users to win-win choices, we propose algorithms that are suitable for three applications: generating an auto-complete list, e.g. "$coach \rightarrow coach\ handbag$", related keywords, e.g. "$iphone \rightarrow blackberry$", and displaying query suggestions in visualization when it is appropriate, e.g. "$dress$".

5.1 Future work

We will now discuss possible areas for future works. In Chapter 2, we assume that the social tag data is of reasonable quality before our analysis. In our experiments, we evaluate our work against a dataset collected from Wikipedia. When many users can agree on certain tags, the resulting quality of the tags is usually high. However, when any user is able to create any tag for an entity, as commonly seen in many social-tagging websites, its expected that the amount of noise will increase, prompting the need for more noise-reduction strategies.

In Chapter 3, we investigated the question, "when comparing experts versus the crowd, who will win in predicting future trends?" An interesting follow-up research is to figure out the best strategy for selecting experts when resources are flexible. While we examined three basic, intuitive strategies in selecting experts, some advanced, mixed strategies can also be considered. For example, we can extract experts based on following the results of social-graph analysis, i.e. who follows whom on Twitter.

In Chapter 4, we studied how to recommend query suggestions. However, our work did not put user segmentation aspects into account. For example, shopping behaviors between men and women might be very different. Further clustering users into groups and investigating their differences may lead to better results because user behaviors are then more accurately modeled. Moreover, our evaluation is based on historical data. A feedback collection mechanism could be incorporated to track which query suggestions are associated with high click-through rates, and we may make use of user-feedback data to further enhance our algorithms.

REFERENCES

[AC98] Alan Agresti and Brent A. Coull. "Approximate is Better than "Exact" for Interval Estimation of Binomial Proportions." *The American Statistician*, **52**(2):119–126, May 1998.

[AGH09] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. "Diversifying search results." In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pp. 5–14, New York, NY, USA, 2009. ACM.

[AK08] Avi Arampatzis and Jaap Kamps. "A study of query length." In Sung-Hyon Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat-Seng Chua, and Mun-Kew Leong, editors, *SIGIR*, pp. 811–812. ACM, 2008.

[Bat86] T. St. John N. Bates. "Parliament, Policy and Delegated Power." *Statute Law Review*, **7**:114–123, 1986.

[BBC09a] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, and Sebastiano Vigna. "Query suggestions using query-flow graphs." In *Proceedings of the 2009 workshop on Web Search Click Data*, WSCD '09, pp. 56–63, New York, NY, USA, 2009. ACM.

[BBC09b] Paolo Boldi, Francesco Bonchi, Carlos Castillo, and Sebastiano Vigna. "From "Dango" to "Japanese Cakes": Query Reformulation Models and Patterns." In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '09, pp. 183–190, Washington, DC, USA, 2009. IEEE Computer Society.

[BCD99] Lawrence D. Brown, T. Tony Cai, and Anirban Dasgupta. "Interval Estimation for a Binomial Proportion.", July 30 1999.

[Lawrence D. Brown (University of Pennsylvania , Purdue University and Purdue University); T. Tony Cai (University of Pennsylvania , Purdue University and Purdue University); Anirban Dasgupta (University of Pennsylvania , Purdue University and Purdue University);.]

[BCD09] Ranieri Baraglia, Carlos Castillo, Debora Donato, Franco Maria Nardini, Raffaele Perego, and Fabrizio Silvestri. "Aging effects on query flow graphs for query suggestion." In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pp. 1947–1950, New York, NY, USA, 2009. ACM.

[BD97] Peter D Bruza and Simon Dennis. "Query reformulation on the internet: Empirical data and the hyperindex search engine." *Proceedings of RIAO97, Computer-Assisted Information Searching on Internet, Montreal*, 1997.

[BHM05] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. "Query recommendation using query logs in search engines." In *Current Trends in Database Technology-EDBT 2004 Workshops*, pp. 588–596. Springer, 2005.

[BHM11] Eytan Bakshy, Jake M. Hofman, Winter A. Mason, and Duncan J. Watts. "Everyone's an influencer: quantifying influence on twitter." In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pp. 65–74, New York, USA, 2011.

[BP98] Sergey Brin and Lawrence Page. "The anatomy of a large-scale hypertextual Web search engine." In *Proceedings of the seventh international conference on World wide web*, WWW '07, pp. 107–117, 1998.

[BT02] Dimitri P Bertsekas and John N Tsitsiklis. *Introduction to probability*. Athena Scientific Belmont, 2002.

[CDS10] Mario Cataldi, Luigi Di Caro, and Claudio Schifanella. "Emerging topic detection on Twitter based on temporal and social terms evaluation." In *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, MDMKDD '10, pp. 4:1–4:10, New York, USA, 2010.

[CJP08] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. "Context-aware query suggestion by mining click-through and session data." In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pp. 875–883, New York, NY, USA, 2008. ACM.

[CMP11] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. "Information credibility on twitter." In *Proceedings of the 20th international conference on World wide web*, WWW '11, pp. 675–684, New York, USA, 2011.

[CR12] Claudio Carpineto and Giovanni Romano. "A survey of automatic query expansion in information retrieval." *ACM Computing Surveys (CSUR)*, **44**(1):1, 2012.

[DBA10] Murat Demirbas, Murat Ali Bayir, Cuneyt Gurcan Akcora, Yavuz Selim Yilmaz, and Hakan Ferhatosmanoglu. "Crowd-sourced sensing and collaboration using twitter." In *International Symposium on a World of Wireless Mobile and Multimedia Networks*, WOWMOM'10, pp. 1–9. IEEE, 2010.

[DEM11] Antonina Dattolo, Davide Eynard, and Luca Mazzola. "An integrated approach to discover tag semantics." In William C. Chu, W. Eric Wong, Mathew J. Palakal, and Chih-Cheng Hung, editors, *SAC*, pp. 814–820. ACM, 2011.

[DGL12] Gianmarco De Francisci Morales, Aristides Gionis, and Claudio Lucchese. "From chatter to headlines: harnessing the real-time web for personalized news recommendation." In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pp. 153–162, New York, USA, 2012.

[DLK09] Hongbo Deng, Michael R. Lyu, and Irwin King. "A generalized Co-HITS algorithm and its application to bipartite graphs." In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pp. 239–248, New York, NY, USA, 2009. ACM.

[Fam70] Eugene Fama. "Efficient Capital Markets: A Review of Theory and Empirical Work." *Journal of Finance*, May 1970.

[FN94] Robert Frederking and Sergei Nirenburg. "Three heads are better than one." In *Proceedings of the fourth conference on Applied natural language processing*, ANLC '94, pp. 95–100, Stroudsburg, PA, USA, 1994.

[FTA08] Ariel Fuxman, Panayiotis Tsaparas, Kannan Achan, and Rakesh Agrawal. "Using the wisdom of the crowds for keyword generation." In Jinpeng Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang, editors, *WWW*, pp. 61–70. ACM, 2008.

[GH06] Scott A. Golder and Bernardo A. Huberman. "Usage patterns of collaborative tagging systems." *J. Information Science*, **32**(2):198–208, 2006.

[GKL08] Vicenç Gómez, Andreas Kaltenbrunner, and Vicente López. "Statistical analysis of the social network and discussion threads in slashdot." In *Proceedings of the 17th international conference on World wide web*, WWW '08, pp. 645–654, New York, USA, 2008.

[GL09] Sharon Givon and Victor Lavrenko. "Large Scale Book Annotation with Social Tags." In Eytan Adar, Matthew Hurst, Tim Finin, Natalie S. Glance, Nicolas Nicolov, and Belle L. Tseng, editors, *ICWSM*. The AAAI Press, 2009.

[GS09] Asela Gunawardana and Guy Shani. "A Survey of Accuracy Evaluation Metrics of Recommendation Tasks." *Journal of Machine Learning Research*, **10**:2935–2962, 2009.

[GSB12] Saptarshi Ghosh, Naveen Sharma, Fabricio Benevenuto, Niloy Ganguly, and Krishna Gummadi. "Cognos: crowdsourcing search for topic experts in microblogs." In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, pp. 575–590, New York, USA, 2012.

[HCO03] Chien-Kang Huang, Lee-Feng Chien, and Yen-Jen Oyang. "Relevant term suggestion in interactive web search based on contextual information in query session logs." *J. Am. Soc. Inf. Sci. Technol.*, **54**(7):638–649, May 2003.

[Hil82] Gayle W. Hill. "Group versus individual performance: Are N+1 heads better than one?" *Psychological Bulletin*, **91**(3):517–539, May 1982.

[HKT04] Jonathan L. Herlocker, Joseph A. Konstan, Loren Terveen, and John T. Riedl. "Evaluating collaborative filtering recommender systems." *ACM Trans. Inf. Syst.*, **22**(1):5–53, 2004.

[HPG10] Paul Heymann, Andreas Paepcke, and Hector Garcia-Molina. "Tagging human knowledge." In Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu, editors, *WSDM*, pp. 51–60. ACM, 2010.

[HPS11] Mohammad Al Hasan, Nish Parikh, Gyanit Singh, and Neel Sundaresan. "Query suggestion for E-commerce sites." In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pp. 765–774, New York, NY, USA, 2011. ACM.

[JSF07] Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. "Why we twitter: understanding microblogging usage and communities." In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, WebKDD/SNA-KDD '07, pp. 56–65, New York, USA, 2007.

[Kle99] Jon M. Kleinberg. "Authoritative sources in a hyperlinked environment." *J. ACM*, **46**(5):604–632, September 1999.

[KLP10] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. "What is Twitter, a social network or a news media?" In *Proceedings of the 19th international conference on World wide web*, WWW '10, pp. 591–600, New York, USA, 2010.

[Kou02] Marios Koufaris. "Applying the technology acceptance model and flow theory to online consumer behavior." *Information systems research*, **13**(2):205–223, 2002.

[KPS07] Aniket Kittur, Bryan A. Pendleton, Bongwon Suh, and Todd Mytkowicz. "Power of the few vs. wisdom of the crowd: Wikipedia and the rise of the bourgeoisie." *World wide web*, **1**(2):19, 2007.

[KST12] Makoto P. Kato, Tetsuya Sakai, and Katsumi Tanaka. "Structured query suggestion for specialization and parallel movement: effect on search behaviors." In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pp. 389–398, New York, NY, USA, 2012. ACM.

[LH10] Kristina Lerman and Tad Hogg. "Using a model of social dynamics to predict popularity of news." In *Proceedings of the 19th international conference on World wide web*, WWW '10, pp. 621–630, New York, USA, 2010. ACM.

[LHY09] Dong Liu, Xian-Sheng Hua, Linjun Yang, Meng Wang, and Hong-Jiang Zhang. "Tag ranking." In Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl, editors, *WWW*, pp. 351–360. ACM, 2009.

[LKF05] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. "Graphs over time." In *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pp. 177–187, New York, USA, 2005.

[This paper was cite by Kwak (WWW2010).]

[Mat04] Adam Mathes. "Folksonomies — Cooperative Classification and Communication Through Shared Metadata." Technical Report LIS590CMC, University of Illinois, Urbana-Champaign, Illinois, December 2004.

[MLK10] Hao Ma, Michael R Lyu, and Irwin King. "Diversifying query suggestion results." In *Proc. of AAAI*, volume 10, 2010.

[MSP12] Yasuko Matsubara, Yasushi Sakurai, B. Aditya Prakash, Lei Li, and Christos Faloutsos. "Rise and fall patterns of information diffusion: model and implications." In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pp. 6–14, New York, NY, USA, 2012. ACM.

[MWL12] Xinfan Meng, Furu Wei, Xiaohua Liu, Ming Zhou, Sujian Li, and Houfeng Wang. "Entity-centric topic-oriented opinion summarization in twitter." In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pp. 379–387, New York, NY, USA, 2012. ACM.

[MYK08] Hao Ma, Haixuan Yang, Irwin King, and Michael R. Lyu. "Learning latent semantic relations from clickthrough data for query suggestion." In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pp. 709–718, New York, NY, USA, 2008. ACM.

[MZC08] Qiaozhu Mei, Dengyong Zhou, and Kenneth Church. "Query suggestion using hitting time." In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pp. 469–478, New York, NY, USA, 2008. ACM.

[MZL11] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. "Recommender systems with social regularization." In *Proceedings of*

the fourth ACM international conference on Web search and data mining, WSDM '11, pp. 287–296, New York, USA, 2011.

[PBW90] Roy M. Poses, Carolyn Bekes, Robert L. Winkler, W. Eric Scott, and Fiore J. Copare. "Are Two (Inexperienced) Heads Better Than One (Experienced) Head? Averaging House Officers' Prognostic Judgments for Critically Ill Patients." *Arch Intern Med*, **150**(9):1874–1878, September 1990.

[PMS09] Owen Phelan, Kevin McCarthy, and Barry Smyth. "Using twitter to recommend real-time topical news." In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pp. 385–388, New York, USA, 2009. ACM.

[POL10] Sasa Petrovic, Miles Osborne, and Victor Lavrenko. "Streaming First Story Detection with application to Twitter." In *Proceedings of the 2010 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, HLT-NAACL '10, pp. 181–189, 2010.

[PVW09] D Veena Parboteeah, Joseph S Valacich, and John D Wells. "The influence of website characteristics on a consumer's urge to buy impulsively." *Information Systems Research*, **20**(1):60–78, 2009.

[PW08] Alex Penev and Raymond K. Wong. "Finding similar pages in a social tagging repository." In Jinpeng Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang, editors, *WWW*, pp. 1091–1092. ACM, 2008.

[RBJ89] Vijay Raghavan, Peter Bollmann, and Gwang S. Jung. "A critical investigation of recall and precision as measures of retrieval system performance." *ACM Trans. Inf. Syst.*, **7**(3):205–229, July 1989.

[RJ76] Stephen E. Robertson and Karen Sparck Jones. "Relevance weighting of search terms." *Journal of the American Society for Information Science*, **27**:129–146, 1976.

[RMK11] Daniel M. Romero, Brendan Meeder, and Jon Kleinberg. "Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter." In *Proceedings of the 20th international conference on World wide web*, WWW '11, pp. 695–704, New York, USA, 2011.

[SKK10] Markus Strohmaier, Christian Körner, and Roman Kern. "Why do Users Tag? Detecting Users' Motivation for Tagging in Social Tagging Systems." In William W. Cohen and Samuel Gosling, editors, *ICWSM*. The AAAI Press, 2010.

[SMW10] Eldar Sadikov, Jayant Madhavan, Lu Wang, and Alon Halevy. "Clustering query refinements by user intent." In *Proceedings of the 19th international conference on World wide web*, WWW '10, pp. 841–850, New York, NY, USA, 2010. ACM.

[Ste62] Hawkins Stern. "The significance of impulse buying today." *The Journal of Marketing*, pp. 59–62, 1962.

[Sur05] James Surowiecki. *The Wisdom of Crowds*. Anchor Books. Anchor, 2005.

[SZH11] Yang Song, Dengyong Zhou, and Li-wei He. "Post-ranking query suggestion by diversifying search results." In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pp. 815–824, New York, NY, USA, 2011. ACM.

[TMS08] Karen H. L. Tso-Sutter, Leandro Balby Marinho, and Lars Schmidt-Thieme. "Tag-aware recommender systems by fusion of collaborative filtering algorithms." In Roger L. Wainwright and Hisham Haddad, editors, *SAC*, pp. 1995–1999. ACM, 2008.

[TRM11] Jaime Teevan, Daniel Ramage, and Merredith Ringel Morris. "#TwitterSearch: a comparison of microblog search and web search." In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pp. 35–44, New York, USA, 2011.

[WYY09] Lei Wu, Linjun Yang, Nenghai Yu, and Xian-Sheng Hua. "Learning to tag." In Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl, editors, *WWW*, pp. 361–370. ACM, 2009.

[WZB10] Robert Wetzker, Carsten Zimmermann, Christian Bauckhage, and Sahin Albayrak. "I tag, you tag: translating tags for advanced user models." In Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu, editors, *WSDM*, pp. 71–80. ACM, 2010.

[XC96] Jinxi Xu and W Bruce Croft. "Query expansion using local and global document analysis." In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 4–11. ACM, 1996.

[YLW12] Peifeng Yin, Ping Luo, Min Wang, and Wang-Chien Lee. "A straw shows which way the wind blows: ranking potentially popular items from early votes." In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pp. 623–632, New York, USA, 2012.

[ZPS07] Xiaoni Zhang, Victor R Prybutok, and David Strutton. "Modeling influences on impulse purchasing behaviors during online marketing transactions." *The Journal of Marketing Theory and Practice*, **15**(1):79–89, 2007.