Part 1

Gathering data for intelligence

Chapter 1 begins the book with a brief overview of what collective intelligence is and how it manifests itself in your application. Then we move on to focus on how we can gather data from which we can derive intelligence. For this, we look at information both inside the application (chapters 2 through 4) and outside the application (chapters 5 and 6).

Chapter 2 deals with learning from the interactions of users. To get the ball rolling, we look at the architecture for embedding intelligence, and present some of the basic concepts related to collective intelligence (CI). We also cover how we can gather data from various forms of user interaction. We continue with this theme in chapter 3, which deals with tagging. This chapter contains all the information you need to build tagging-related features in your application. In chapter 4, we look at the various forms of content that are typically available in a web application and how to derive collective intelligence from it.

Next, we change our focus to collecting data from outside our application. We first deal with searching the blogosphere in chapter 5. This is followed by chapter 6, which deals with intelligently crawling the web in search of relevant content.

1 Understanding collective intelligence

This chapter covers

- The basics of collective intelligence
- How collective intelligence manifests itself in web applications
- Building user-centric applications using collective intelligence
- The three forms of intelligence: direct, indirect, and derived

Web applications are undergoing a revolution.

In this post-dot-com era, the web is transforming. Newer web applications trust their users, invite them to interact, connect them with others, gain early feedback from them, and then use the collected information to constantly improve the application. Web applications that take this approach develop deeper relationships with their users, provide more value to users who return more often, and ultimately offer more targeted experiences for each user according to her personal need.

Web users are undergoing a transformation.

Users are expressing themselves. This expression may be in the form of sharing their opinions on a product or a service through reviews or comments; through sharing and tagging content; through participation in an online community; or by contributing new content.

This increased user interaction and participation gives rise to data that can be converted into intelligence in your application. The use of collective intelligence to personalize a site for a user, to aid him in searching and making decisions, and to make the application more sticky are cherished goals that web applications try to fulfill.

In his book, *Wisdom of the Crowds*, James Surowiecki, business columnist for *The New Yorker*, asserts that "under the right circumstances, groups are remarkably intelligent, and are often smarter than the smartest people in them." Surowiecki says that if the process is sound, the more people you involve in solving a problem, the better the result will be. A crowd's *collective intelligence* will produce better results than those of a small group of experts if four basic conditions are met. These four basic conditions are that "wise crowds" are effective when they're composed of individuals who have diverse opinions; when the individuals aren't afraid to express their opinions; when there's diversity in the crowd; and when there's a way to aggregate all the information and use it in the decision-making process.

Collective intelligence is about making your application more valuable by tapping into *wise crowds*. More formally, collective intelligence (CI) as used in this book simply and concisely means

To effectively use the information provided by others to improve one's application.

This is a fairly broad definition of collective intelligence—one which uses all types of information, both inside and outside the application, to improve the application for a user. This book introduces you to concepts from the areas of machine learning, information retrieval, and data mining, and demonstrates how you can add intelligence to your application. You'll be exposed to how your application can learn about individual users by correlating their interactions with those of others to offer a highly personalized experience.

This chapter provides an overview of collective intelligence and how it can manifest itself in your application. It begins with a brief introduction to the field of collective intelligence, then goes on to describe the many ways it can be applied to your application, and finally shows how intelligence can be classified.

1.1 What is collective intelligence?

Collective intelligence is an active field of research that predates the web. Scientists from the fields of sociology, mass behavior, and computer science have made important contributions to this field. When a group of individuals collaborate or compete with each other, intelligence or behavior that otherwise didn't exist suddenly emerges; this is commonly known as *collective intelligence*. The actions or influence of a few individuals slowly spread across the community until the actions become the norm for the

community. To better understand how this circle of influence spreads, let's look at a couple of examples.

In his book *The Hundredth Monkey*,[1] Ken Keyes recounts an interesting story about how change is propagated in groups. In 1952, on the isolated Japanese island of Koshima, scientists observed a group of monkeys. They offered them sweet potatoes; the monkeys liked the sweet potatoes but found the taste of dirt and sand on the potatoes unpleasant. One day, an 18-month-old monkey named Imo found a solution to the problem by washing the potato in a nearby stream of water. She taught this trick to her mother. Her playmates also learned the trick and taught it to their mothers. Initially, only adults who imitated their children learned the new trick, while the others continued eating the old way. In the autumn of 1958, a number of monkeys were washing their potatoes before eating. The exact number is unknown, but let's say that out of 1,000, there were 99 monkeys who washed their potatoes before eating. Early one sunny morning, a 100th monkey decided to wash his potato. Then, incredibly, by evening *all* monkeys were washing their potatoes. The 100th monkey was that *tipping point* that caused others to change their habits for the better. Soon it was observed that monkeys on other islands were also washing their potatoes before eating them.

As users interact on the web and express their opinions, they influence others. Their initial circle of influence is the group of individuals that they most interact with. Because the web is a highly connected network of sites, this *circle of influence* grows and may shape the thoughts of everybody in the group. This circle of influence also grows rapidly throughout the community—another example helps illustrate this further.

In 1918, as the influenza flu pandemic spread, nearly 14 percent of Fiji's population died in just 16 days. Nearly one third of the native population in Alaska had a similar fate; it's estimated that worldwide, nearly twenty-five million people died of the flu. A pandemic is a global disease outbreak and spreads from person to person. First, one person is affected, who then transmits it to another and then another. The newly infected person transmits the flu to others; this causes the disease to spread exponentially.

In October 2006, Google bought YouTube for $1.65 billion. In its 20 months of existence, YouTube had grown to be one of the busiest sites on the Internet, dishing out 100 million video[2] views a day. It ramped from zero to more than 20 million unique user visits a day, with mainly *viral marketing*—spread from person to person, similar to the way the pandemic flu spreads. In YouTube's case, each time a user uploaded a new video, she was easily able to invite others to view this video. As those others viewed this video, other related videos popped up as recommendations, keeping the user further engaged. Ultimately, many of these viewers also became submitters and uploaded their own videos as well. As the number of videos increased, the site became more and more attractive for new users to visit.

---

[1] http://en.wikipedia.org/wiki/Hundredth_Monkey

[2] As of September 2006

Whether you're a budding startup, a recognized market leader, or looking to take an emerging application or web site to the next level, harnessing information from users improves the perceived value of the application to both current and prospective users. This improved value will not only encourage current users to interact more, but will also attract new users to the application. The value of the application further improves as new users interact with it and contribute more content. This forms a self-reinforcing feedback loop, commonly known as a *network effect*, which enables wider adoption of the service. Next, let's look at CI as it applies to web applications.

1.2 CI in web applications

In this section, we look at how CI manifests itself in web applications. We walk through an example to illustrate how it can be used in web applications, briefly review its benefits, see how it fits in with Web 2.0 and can be leveraged to build user-centric applications.

Let's expand on our earlier definition of collective intelligence. Collective intelligence of users in essence is

- *The intelligence that's extracted out from the collective set of interactions and contributions made by your users.*
- *The use of this intelligence to act as a filter for what's valuable in your application for a user*—This filter takes into account a user's preferences and interactions to provide relevant information to the user.

This filter could be the simple influence that collective user information has on a user—perhaps a rating or a review written about a product, as shown in figure 1.1—or it may be more involved—building models to recommend personalized content to a user. This book is focused toward building the more involved models to personalize your application.

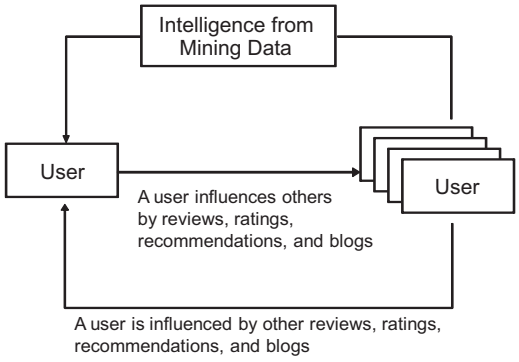


Figure 1.1 A user may be influenced by other users either directly or through intelligence derived from the application by mining the data.

As shown in figure 1.2, there are three things that need to happen to apply collective intelligence in your application. You need to

1. Allow users to interact with your site and with each other, learning about each user through their interactions and contributions.
2. Aggregate what you learn about your users and their contributions using some useful models.
3. Leverage those models to recommend relevant content to a user.

Let's walk through an example to understand how collective intelligence can be a catalyst to building a successful web application.

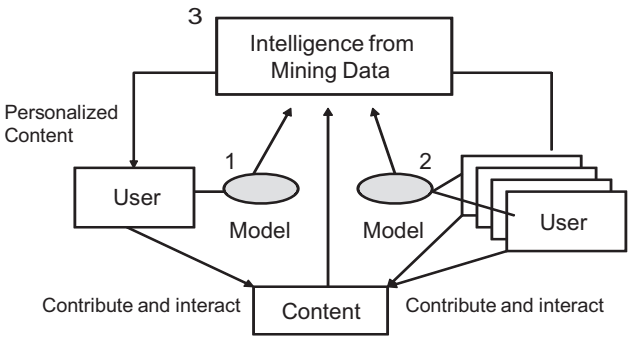


Figure 1.2 Three components to harnessing collective intelligence. 1: Allow users to interact. 2: Learn about your users in aggregate. 3: Personalize content using user interaction data and aggregate data.

1.2.1 Collective intelligence from the ground up: a sample application

In our example, John and Jane are two engineers who gave up their lucrative jobs to start a company. They're based in Silicon Valley and as is the trend nowadays, they're building their fledgling company without any venture capital on a shoestring budget leveraging open source software. They believe in fast-iterative agile-based development cycles and aren't afraid to release beta software to gain early feedback on their features .[3] They're looking to build a marketplace and plan to generate revenue both from selling ad space and from sharing revenue from sold items.

In their first iteration, they launched an application where users—mainly friends and family—could buy items and view relevant articles. There wasn't much in terms of personalization or user interaction or intelligence—a plain vanilla system.

Next, they added the feature of showing a list of top items purchased by users, along with a list of recently purchased items. This is perhaps the simplest form of applying collective intelligence—*providing information in aggregate to users*. To grow the application virally, they also enabled users to email these lists to others. Users used this to forward interesting lists of items to their friends, who in turn became users of the application.

In their next iteration, they wanted to learn more about their users. So they built a basic user profile mechanism that contained explicit and implicit profile information. The explicit information was provided directly by the users as part of their accounts—first name, age, and so on. The implicit information was collected from the user interaction data—this included information such as the articles and content users viewed and the products they purchased. They also wanted to show more relevant articles and content to each user, so they built a content-based *recommendation engine* that analyzed the content of articles—keywords, word frequency, location, and so forth to correlate articles with each other and recommend possibly interesting articles to each user.

Next, they allowed users to generate content. They gave users the ability to write about their experiences with the products, in essence writing reviews and creating their list of recommendations through both explicit ratings of individual products

[3] Note that beta doesn't mean poor quality; it just means that it's incomplete in functionality.

and a "my top 10 favorite products" list. They also gave users the capability to rate items and rate reviews. Ratings and reviews have been shown to influence other users, and numerical rating information is also useful as an input to a collaborative-based recommendation engine.

With the growing list of content and products available on the site, John and Jane now found it too cumbersome and expensive to manually maintain the classification of content on their site. The users also provided feedback that content navigation menus were too rigid. So they introduced dynamic navigation via a *tag cloud*—navigation built by an alphabetical listing of terms, where font size correlates with importance or number of occurrences of a tag. The terms were automatically extracted from the content by analyzing the content. The application analyzed each user's interaction and provided users with a personalized set of tags for navigating the site. The set of tags changed as the type of content visited by the users changed. Further, the content displayed when a user clicked on a tag varied from user to user and changed over time. Some tags pulled the data from a search engine, while others from the recommendation engine and external catalogs.

In the next release, they allowed the users to explicitly tag items by adding free text labels, along with saving or *bookmarking* items of interest. As users started tagging items, John and Jane found that there was a rich set of information that could be derived. First of all, users were providing new terms for the content that made sense to them—in essence they built *folksonomies*.[4] The tag cloud navigation now had both machine-generated and user-generated tags. The process of extracting tags using an automated algorithm could also be enhanced using the dictionary of tags built by the users. These user-added tags were also useful for finding keywords used by an ad-generation engine. They could also use the tags created by users to connect users with each other and with other items of interest. This is collective intelligence in action.

Next, they allowed their users to generate content. Users could now blog about their experiences, or ask and respond to questions on message boards, or participate in building the application itself by contributing to wikis. John and Jane quickly built an algorithm that could extract tags from the unstructured content. They then matched the interests of users—gained from analyzing their interaction in the applications—with those of other users to find relevant items. They were soon able to learn enough about their users to personalize the site for each user, and to provide relevant content—targeting niche items to niche users. They could also target relevant advertisements based on the user profile and context of interaction.

They also modified the search results to make them more relevant to each user, for which they used the user's profile and interaction history when appropriate. They customized advertising by using keywords that were relevant to both the user and the page content.

To make the application stickier, they started aggregating and indexing external content—they would crawl a select list of external web sites to index the content and present

[4] Folksonomies are classifications created through the process of users tagging items.

links to it when relevant. They also connected to sites that tracked the blogosphere, presenting the users with relevant content from what others were saying in blogs.

They also clustered users and items to find patterns in the data and built models to automatically classify content into one of many categories.

The users soon liked the application so much that they started recommending the application to their friends and relatives and the user base grew *virally*. In our example, after a couple of years, John and Jane retired to Hawaii, having sold the company for a gigantic amount, where they waited for the next web revolution… Web 3.0!

These in essence are the many ways by which collective intelligence will manifest itself in your application, and thus more or less the outline for this book. Table 1.1 summarizes the ways to harness collective intelligence in your application. Each of these is discussed throughout the book.

Table 1.1 Some of the ways to harness collective intelligence in your application

Techniques	Description
Aggregate information: lists	Create lists of items generated in the aggregate by your users. Perhaps, *Top List* of items bought, or *Top Search Items or List of Recent Items*.
Ratings, reviews, and recommendations	Collective information from your users influences others.
User-generated content: blogs, wikis, message boards	Intelligence can be extracted from contributions by users. These contributions also influence other users.
Tagging, bookmarking, voting, saving	Collective intelligence of users can be used to bubble up interesting content, learn about your users, and connect users.
Tag cloud navigation	Dynamic classification of content using terms generated via one or more of the following techniques: machine-generated, professionally-generated, or user-generated.
Analyze content to build user profiles	Analyze content associated with a user to extract keywords. This information is used to build user profiles.
Clustering and predictive models	Cluster users and items, build predictive models.
Recommendation engines	Recommend related content or users based on intelligence gathered from user interaction and analyzing content.
Search	Show more pertinent search results using a user's profile.
Harness external content	Provide relevant information from the blogosphere and external sites

John and Jane showed us a few nice things to apply to their site, but there are other benefits of applying collective intelligence to your application. Let's look at that next.

1.2.2 Benefits of collective intelligence

Applying collective intelligence to your application impacts it in the following manner:

- *Higher retention rates*—The more users interact with the application, the stickier it gets for them, and the higher the probability that they'll become repeat visitors.
- *Greater opportunities to market to the user*—The greater the number of interactions, the greater the number of pages visited by the user, which increases the opportunities to market to or communicate with the user.
- *Higher probability of a user completing a transaction and finding information of interest*—The more contextually relevant information that a user finds, the better the chances that he'll have the information he needs to complete the transaction or find content of interest. This leads to higher click-through and conversion rates for your advertisements.
- *Boosting search engine rankings*—The more users participate and contribute content, the more content is available in your application and indexed by search engines. This could boost your search engine ranking and make it easier for others to find your application.

Collective intelligence is a term that is increasingly being used in the context of Web 2.0 applications. Let's take a closer look at how it fits in with Web 2.0.

1.2.3 CI is the core component of Web 2.0

Web 2.0 is a term that has generated passionate emotions, ranging from being dismissed as marketing jargon to being anointed as the new or next generation of the Internet. There are seven principles that Web 2.0 companies demonstrate, as shown in table 1.2.[5]

Table 1.2 Seven principles of Web 2.0 applications

Principle	Description
The network is the platform	Companies or users who use traditional licensed software have to deal with running the software, upgrading it periodically to keep up with newer versions, and scaling it to meet appropriate levels of demand. Most successful Web 2.0 companies no longer sell licensed software, but instead deliver their software as a service. The end customer simply uses the service through a browser. All the headaches of running, maintaining, and scaling the software and hardware are taken care of by the service provider seamlessly to the end user. The software is upgraded fairly frequently by the service provider and is available 24 x 7.
Harnessing collective intelligence	The key to the success of Web 2.0 applications is how effectively they can harness the information provided by users. The more personalized your service, the better you can match a user to content of her choice.
Hard-to-replicate data as competitive advantage	Hard-to-replicate, unique, large datasets provide a competitive advantage to a company. Web 2.0 is *data* and *software* combined. One can't replicate Craigslist, eBay, Amazon, Flickr, or Google simply by replicating the software. The underlying data that the software generates from user activity is tremendously valuable. This dataset grows every day, improving the product daily.

[5] Refer to Tim O'Reilly's paper on Web 2.0.

Table 1.2 Seven principles of Web 2.0 applications *(continued)*

Principle	Description
The perpetual beta	Web 2.0 companies release their products early to involve their users and gain important feedback. They iterate often by having short release cycles. They involve the users early in the process. They instrument the application to capture important metrics on how a new feature is being used, how often it's being used, and by whom. If you aren't sure how a particular feature should look and have competing designs, expose a prototype of each to different sets of users and measure the success of each. Involve the customers and let them decide which one they like. By having short development cycles, it's possible to solicit user feedback, incorporate changes early in the product life cycle, and build what the users really want.
Simpler programming models	Simpler development models lead to wider adoption and reuse. Design your application for "hackability" and "remixability" following open standards, using simple programming models and a licensing structure that puts as few restrictions as necessary.
Software above the level of a single device	Applications that operate across multiple devices will be more valuable than those that operate in a single device.
Rich user experience	The success of AJAX has fueled the growing use of rich user interfaces in Web 2.0 applications. Adobe Flash/Flex and Microsoft Silverlight are other alternatives for creating rich UIs.

It is widely regarded that harnessing collective intelligence is the key or core component to Web 2.0 applications. In essence, Web 2.0 is all about inviting users to participate and interact. But what do you do with all the data collected from user participation and interaction? This information is wasted if it can't be converted into intelligence and channeled into improving one's application. That's where collective intelligence and this book come in.

Dion Hinchliffe, in his article, "Five Great Ways to Harness Collective Intelligence," makes an analogy to the apocryphal Einstein quote that compound interest was the most important force in the universe. Similarly, web applications that effectively harness collective intelligence can "benefit" in much the same way—harnessing collective intelligence is about those very same exponential effects.

NOTE *Collective intelligence is the heart of Web 2.0 applications.* It's generally acknowledged that one of the core components of Web 3.0 applications will be the use of artificial intelligence.[6] There's debate as to whether this intelligence will be attained by computers reasoning like humans or by sites leveraging the collective intelligence of humans using techniques such as collaborative filtering. Either way, having the dataset generated from real human interactions will be necessary and useful.

In order to effectively leverage collective intelligence, you need to put the user at the center of your application, in essence building a user-centric application.

[6] http://en.wikipedia.org/wiki/Web_3.0#An_evolutionary_path_to_artificial_intelligence

1.2.4 Harnessing CI to transform from content-centric to user-centric applications

Prior to the user-centric revolution, many applications put little emphasis on the user. These applications, known as *content-centric* applications, focused on the best way to present the content and were generally static from user to user and from day to day. User-centric applications leverage CI to fundamentally change how the user interacts with the web application. User-centric applications make the user the center of the web experience and dynamically reshuffle the content based on what's known about the user and what the user explicitly asks for.

As shown in figure 1.3, user-centric applications are composed of the following four components:

- *Core competency*—The main reason why a user comes to the application.
- *Community*—Connecting users with other users of interest, social networking, finding other users who may provide answers to a user's questions.
- *Leveraging user-generated content*—Incorporating generated content and interactions of users to provide additional content to users.
- *Building a marketplace*—Monetizing the application by product and/or service placements and showing relevant advertisements.

The user profile is at the center of the application. A part of the user profile may be generated by the user, while some parts of it may be learned by the application based on user interaction. Typically, sites that allow user-generated content have an abundance of information. User-centric sites leverage collective intelligence to present relevant content to the user.

Figure 1.4 shows a screenshot of one such user-centric application—LinkedIn,[7] a popular online network of more than 20 million professionals.[8] As shown in

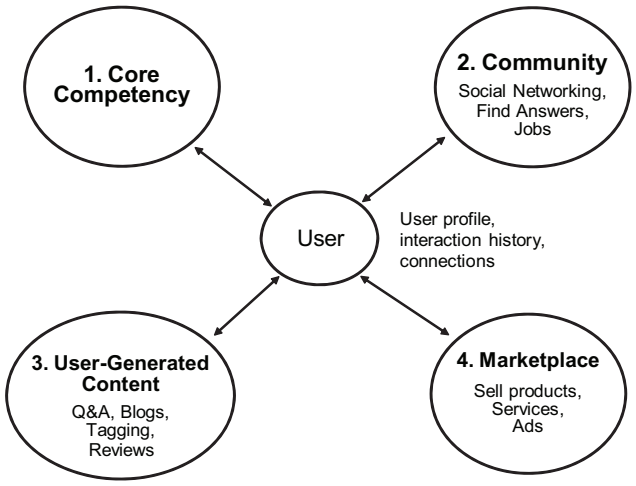


Figure 1.3 Four pillars for user-centric applications

[7] http://www.linkedin.com/static?key=company_info&trk=ftr_abt

[8] As of May 2008

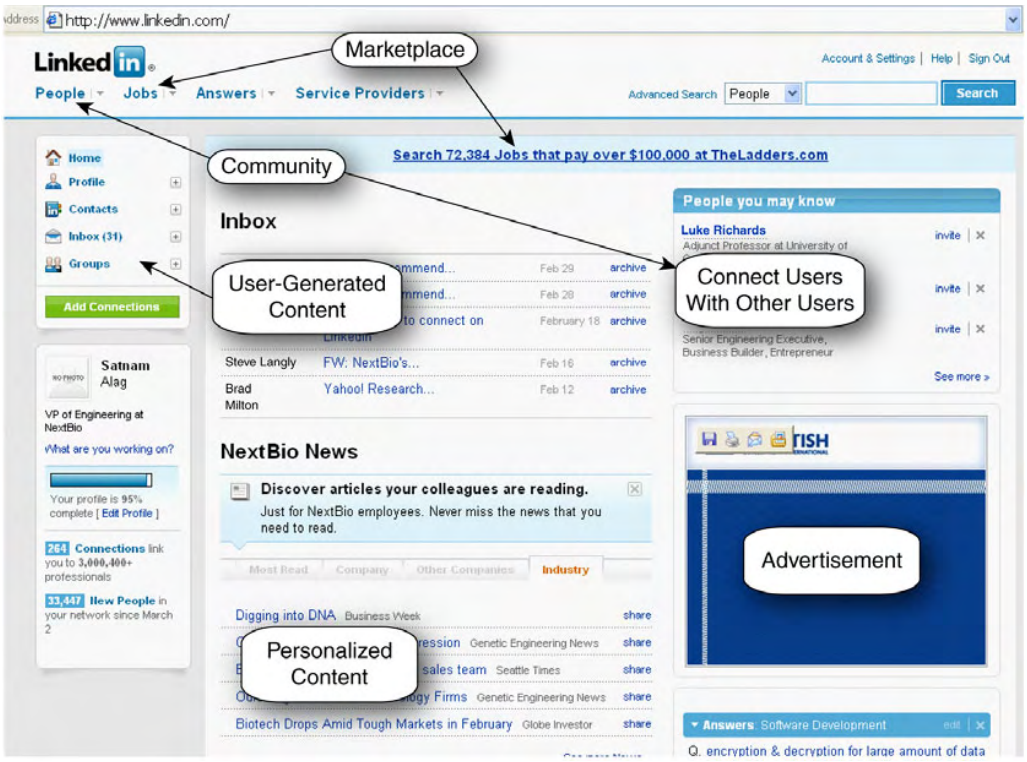


Figure 1.4 An example of a user-centric application—LinkedIn (www.linkedin.com)

the screenshot, the LinkedIn application leverages the four components of user-centric applications:

- *Core competency*—Users come to the site to connect with others and build their professional profiles.
- *Community*—Users create connections with other users; connections are used while looking up people, responding to jobs, and answering questions asked by other users. Other users are automatically recommended as possible connections by the application.
- *User-generated content*—Most of the content at the site is user-generated. This includes the actual professional profiles, the questions asked, the feed of actions—such as a user updating his profile, uploading his photograph, or connecting to someone new.
- *Marketplace*—The application is monetized by means of advertisements, job postings, and a monthly subscription for the power-users of the system, who often are recruiters. The monetization model used is also commonly known as *freemium*[9]—basic services are free and are used by most users, while there's a charge for premium services that a small minority of users pay for.

[9] http://en.wikipedia.org/wiki/Freemium_business_model

For user-centric applications to be successful, they need to personalize the site for each user. CI can be beneficial to these applications. So far in this section, we've looked at what collective intelligence is, how it manifests itself in your application, the advantages of applying it, and how it fits in with Web 2.0. Next, we'll take a more detailed look at the many forms of information provided by the users.

1.3 Classifying intelligence

Figure 1.5 illustrates the three types of intelligence that we discuss in this book. First is explicit information that the user provides in the application. Second is implicit information that a user provides either inside or outside the application and is typically in an unstructured format. Lastly, there is intelligence that's derived by analyzing the aggregate data collected. This piece of derived intelligence is shown on the upper half of the triangle, as it is based on the information gathered by the other two parts.

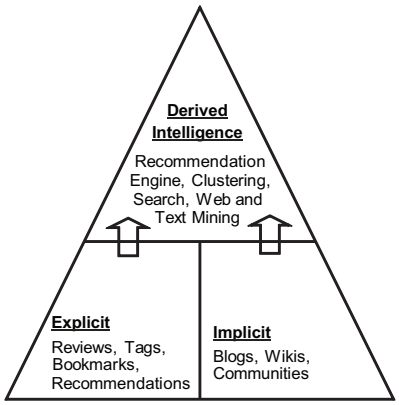


Figure 1.5 Classifying user-generated information

Data comes in two forms: structured data and unstructured data. Structured data has a well-defined form, something that makes it easily stored and queried on. User ratings, content articles viewed, and items purchased are all examples of structured data. Unstructured data is typically in the form of raw text. Reviews, discussion forum posts, blog entries, and chat sessions are all examples of unstructured data.

In this section, we look at the three forms of intelligence: explicit, implicit, and derived.

1.3.1 Explicit intelligence

This section deals with explicit information that a user provides. Here are a few examples of how a user provides explicit information that can be leveraged.

REVIEWS AND RECOMMENDATIONS

A recommendation made by a friend or a person of influence can have a big impact on other users within the same group. Moreover, a review or comments about a user's experience with a particular provider or service is contextually relevant for other users inquiring about that topic, especially if it's within the context of similar use.

TAGGING

Adding the ability for users to add tags—keywords or labels provided by a user—to classify items of interest such as articles, items being sold, pictures, videos, podcasts, and so on is a powerful technique to solicit information from the user. Tags can also be generated by professional editors or by an automated algorithm that analyzes content. These tags are used to classify data, bookmark sites, connect people with each other, aid users in their searches, and build dynamic navigation in your application, of which a tag cloud is one example.

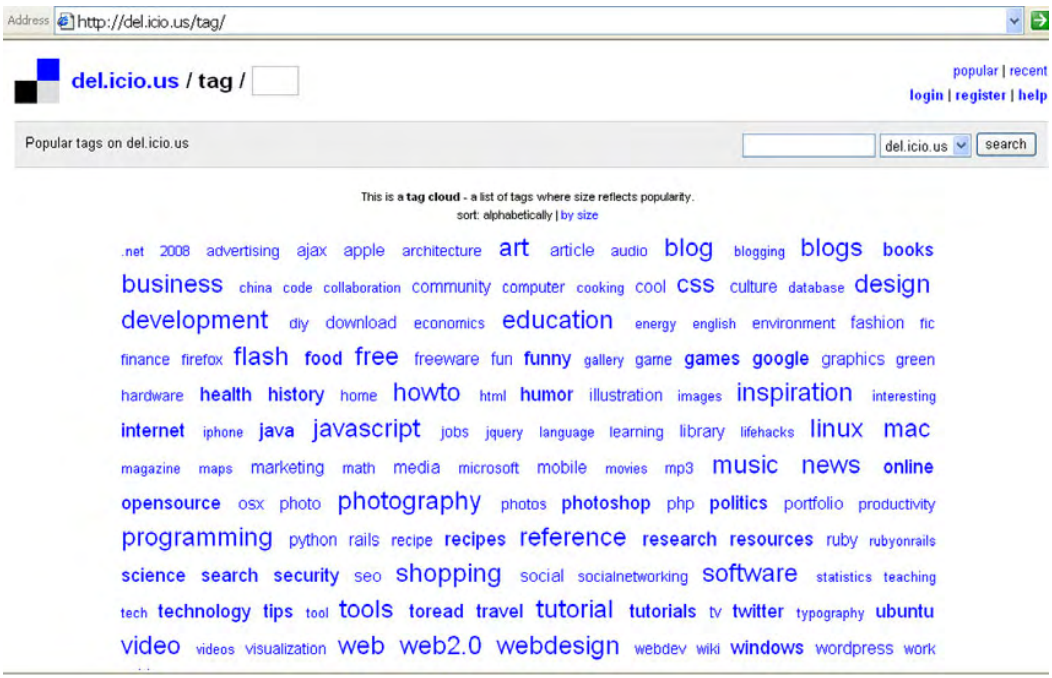


Figure 1.6 This tag cloud from del.icio.us shows popular tags at the site.

Figure 1.6 shows a tag cloud showing popular tags at del.icio.us, a popular bookmarking site. In a tag cloud, tags are displayed alphabetically, with the size of the font representing the frequency of occurrence. The larger the font of the tag, the more frequently it occurs.

VOTING

Voting is another way to involve and obtain useful information from the user. Digg, a web site that allows users to contribute and vote on interesting articles, leverages this idea. Every article on Digg is submitted and voted on by the Digg community. Submissions that receive many votes in a short period tend to move up in rank. This is a good way to share, discover, bookmark, and promote important news. Figure 1.7 is a screenshot from Digg.com showing news items with the number of Diggs associated with each.

1.3.2 Implicit intelligence

This section deals with indirect information that a user provides. Here are a few examples of how a user provides this information.

Information relevant to your application may appear in an unstructured free-form text format through reviews, messages, blogs, and so forth. A user may express his opinion online, either within your application or outside the application, by writing in his blog or replying to a question in an online community. Thanks to the power of search engines and blog-tracking engines, this information becomes easily available to others and helps to shape their opinions.

You may want to augment your current application by aggregating and mining external data. For example, if your area is real estate applications, you may want to augment your application with additional data harvested from freely available external

Figure 1.7 Screen shot from Digg.com showing news items with the number of diggs for each

sites, for example, public records on housing sales, reviews of schools and neighborhoods, and so on.

Blogs are online journals where information is displayed in reverse chronological order. The blogosphere—the collection of blogs on the net—is huge and growing fast. As of August 2008, Technorati, a private company that tracks blogs, was tracking 112.8 million blogs. With a new blog being created virtually every second, the blogosphere is an important source of information that can be leveraged in your application. People write blogs on virtually every topic.

Next, let's look at the third category of intelligence, which is derived from analyzing the data collected.

1.3.3 Derived intelligence

This section deals with information derived from the data you collect from users. Here are a few examples of techniques and features that deal with derived intelligence.

DATA AND TEXT MINING

The process of finding patterns and trends that would otherwise go undetected in large datasets using automated algorithms is known as *data mining*. When the data is in the form of text, the mining process is commonly known as *text data mining*. Another

related field is *information retrieval*, which deals with finding relevant information by analyzing the content of the documents. Web and text mining deal with analyzing unstructured content to find patterns in them. Most applications are content-rich. This content is indexed by search engines and can be used by the recommendation engine to recommend relevant content to a user.

CLUSTERING AND PREDICTIVE ANALYSIS

Clustering and predictive analysis are two main components of data mining. Clustering techniques enable you to classify items—users or content—into natural groupings. Predictive analysis is a mathematical model that predicts a value based on the input data.

INTELLIGENT SEARCH

Search is one of the most commonly used techniques for retrieving content. In later chapters, we look at *Lucene*—an open source Java search engine developed through the Apache foundation. We look at how information about the user can be used to customize the search through intelligent filters that enhance search results when appropriate.

RECOMMENDATION ENGINE

A recommendation engine offers relevant content to a user. Again, recommendation engines can be built by analyzing the content, by analyzing user interactions (collaborative approach), or a combination of both. Figure 1.8 shows a screenshot from Yahoo! Music in which a user is recommended music by the application.

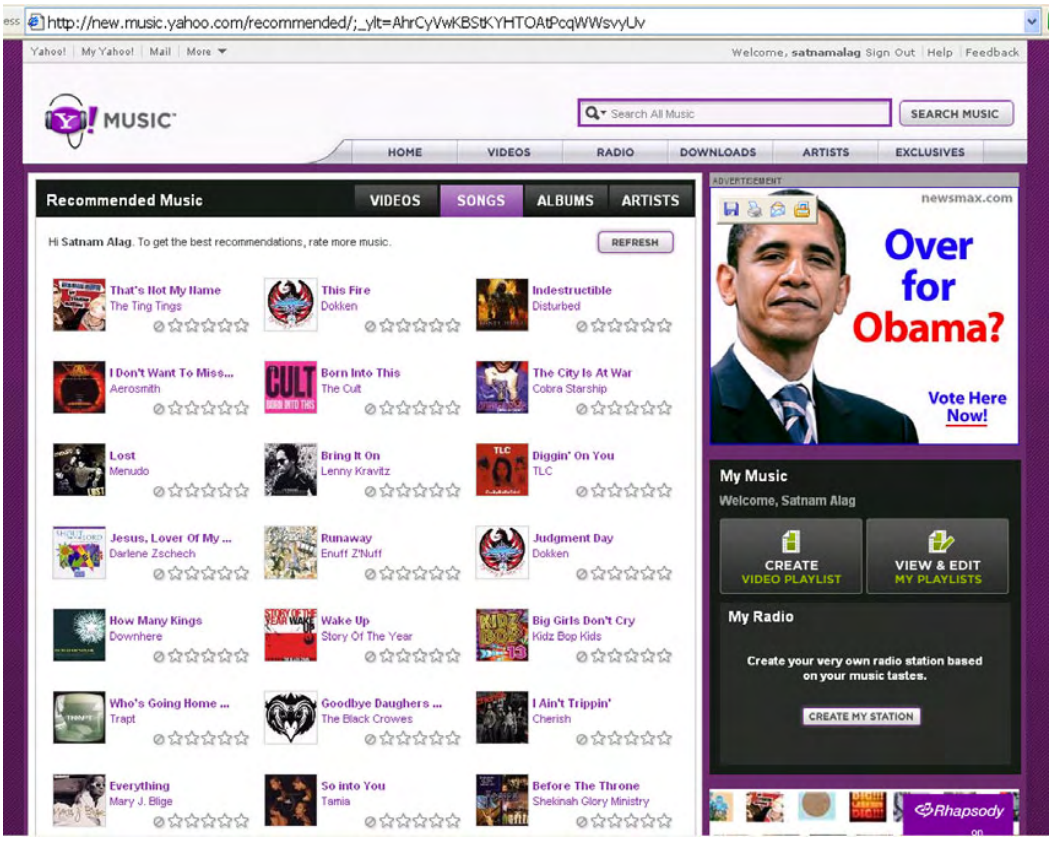


Figure 1.8 Screenshot from Yahoo! Music recommending songs of interest

Recommendation engines use inputs from the user to offer a list of recommended items. The inputs to the recommendation engine may be items in the user's shopping list, items she's purchased in the past or is considering purchasing, user-profile information such as age, tags and articles that the user has looked at or contributed, or any other useful information that the user may have provided. For large online stores such as Amazon, which has millions of items in its catalog, providing fast recommendations can be challenging. Recommendation engines need to be fast and scale independently of the number of items in the catalog and the number of users in the system; they need to offer good recommendations for new customers with limited interaction history; and they need to age out older or irrelevant interaction data (such as a gift bought for someone else) from the recommendation process.

1.4 Summary

Collective intelligence is powering a new breed of applications that invite users to interact, contribute content, connect with other users, and personalize the site experience.

Users influence other users. This influence spreads outward from their immediate circle of influence until it reaches a critical number, after which it becomes the norm. Useful user-generated content and opinions spread virally with minimal marketing.

Intelligence provided by users can be divided into three main categories. First is direct information/intelligence provided by the user. Reviews, recommendations, ratings, voting, tags, bookmarks, user interaction, and user-generated content are all examples of techniques to gather this intelligence. Next is indirect information provided by the user either on or off the application, which is typically in unstructured text. Blog entries, contributions to online communities, and wikis are all sources of intelligence for the application. Third is a higher level of intelligence that's derived using data mining techniques. Recommendation engines, use of predictive analysis for personalization, profile building, market segmentation, and web and text mining are all examples of discovering and applying this higher level of intelligence.

The rest of this book is divided into three parts. The first part deals with collecting data for analysis, the second part deals with developing algorithms for analyzing the data, and the last part deals with applying the algorithms to your application. Next, in chapter 2, we look at how intelligence can be gathered by analyzing user interactions.

1.5 Resources

"All things Web 2.0." http://www.allthingsweb2.com/component/option,com_mtree/Itemid,26/

Anderson, Chris. *The Long Tail: Why the Future of Business Is Selling Less of More.* 2006. Hyperion

Hinchliffe, Dion. "The Web 2.0 Is Here." http://web2.wsj2.com/web2ishere.htm

"Five Great Ways to Harness Collective Intelligence." January 17, 2006, http://web2.wsj2.com/five_great_ways_to_harness_collective_intelligence.htm

"Architectures of Participation: The Next Big Thing." August 1, 2006, http://web2.wsj2.com/architectures_of_participation_the_next_big_thing.htm

Jaokar, Ajit. "Tim O'Reilly's seven principles of web 2.0 make a lot more sense if you change the order." April 17, 2006, http://opengardensblog.futuretext.com/archives/2006/04/tim_o_reillys_s.html

Kroski, Ellyssa. "The Hype and the Hullabaloo of Web 2.0." http://infotangle.blogsome.com/2006/01/13/the-hype-and-the-hullabaloo-of-web-20/

McGovern, Gerry. "Collective intelligence: is your website tapping it?" April 2006, New Thinking, http://www.gerrymcgovern.com/nt/2006/nt-2006-04-17-collective-intelligence.htm

"One blog created 'every second'." BBC news, http://news.bbc.co.uk/1/hi/technology/4737671.stm

"Online Community Toolkit." http://www.fullcirc.com/community/communitymanual.htm

O'Reilly, Tim. "What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software." http://www.oreilly.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

"The Future of Technology and Proprietary Software." December 2003, http://tim.oreilly.com/articles/future_2003.html

"Web 2.0: Compact Definition?" October 2005, http://radar.oreilly.com/archives/2005/10/web_20_compact_definition.html

Por, George. "The meaning and accelerating the emergence of CI." April 2004, http://www.community-intelligence.com/blogs/public/archives/000251.html

Surowiecki, James. *The Wisdom of Crowds*. 2005. Anchor

Web 3.0. Wikipedia, http://en.wikipedia.org/wiki/Web_3.0#An_evolutionary_path_to_artificial_intelligence

On Aggregating Labels from Multiple Crowd Workers to Infer Relevance of Documents

Mehdi Hosseini[1], Ingemar J. Cox[1], Nataša Milić-Frayling[2], Gabriella Kazai[2], and Vishwa Vinay[2]

[1] Computer Science Department, University College London, UK
{m.hosseini,i.cox}@cs.ucl.ac.uk
[2] Microsoft Research, Cambridge. UK
{natasamf,v-gabkaz,vvinay}@microsoft.com

Abstract. We consider the problem of acquiring relevance judgements for information retrieval (IR) test collections through crowdsourcing when no true relevance labels are available. We collect multiple, possibly noisy relevance labels per document from workers of unknown labelling accuracy. We use these labels to infer the document relevance based on two methods. The first method is the commonly used majority voting (MV) which determines the document relevance based on the label that received the most votes, treating all the workers equally. The second is a probabilistic model that concurrently estimates the document relevance and the workers accuracy using expectation maximization (EM). We run simulations and conduct experiments with crowdsourced relevance labels from the INEX 2010 Book Search track to investigate the accuracy and robustness of the relevance assessments to the noisy labels. We observe the effect of the derived relevance judgments on the ranking of the search systems. Our experimental results show that the EM method outperforms the MV method in the accuracy of relevance assessments and IR systems ranking. The performance improvements are especially noticeable when the number of labels per document is small and the labels are of varied quality.

1 Introduction

In information retrieval (IR), test collections are typically used to evaluate the performance of IR systems. A test collection consists of a document corpus, a set of search queries, and a set of relevance judgments for each query. Relevance judgments indicate which documents in the corpus are relevant to a query and are usually created by employing human assessors.

Traditionally, the assessors are trained experts. However, as the corpus and the number of test queries grow, the cost of acquiring relevance labels from expert judges for a sufficiently large number of documents becomes prohibitive. In response to this problem, the IR community has recently been exploring the use of crowdsourcing services to obtain relevance judgments at scale, see e.g., the TREC Relevance Feedback track.

R. Baeza-Yates et al. (Eds.): ECIR 2012, LNCS 7224, pp. 182–194, 2012.

Web services, such as Amazon's Mechanical Turk (www.mturk.com), facilitate the collection of relevance judgments by temporarily hiring thousands of crowd workers. While the labels provided by the workers are relatively inexpensive to acquire, they vary in quality, introducing noise into the relevance judgments and, consequently, causing inaccuracies in the system evaluation [1]. In order to address the issue of noisy labels, it is common to collect multiple labels per document from different workers, in the hope that the consensus across multiple labels would lead to more accurate relevance assessments.

In this paper we assume that a set of labels is collected for each document from multiple crowd workers and that the accuracy of each worker is unknown, as is the true relevance of the documents. Our aim is to estimate both the true relevance of the documents and the accuracy of the workers. We evaluate two methods: the majority voting (MV) that has been frequently used for label aggregation in IR ([1],[2],[3]) and the probabilistic model for estimating both the relevance of the documents and the workers accuracy using the expectation maximization algorithm (EM) as in [19].

The main contributions of this paper are (1) the use of the EM based method to create relevance judgments for IR test collections and (2) empirical evidence that the EM method offers more reliable relevance estimations than the MV method, especially when labels collected for a document are few or varied in quality.

In the following section we discuss the related work. In Section 3 we present details of the EM method. In Section 4 we describe simulations with synthetic data and experiments with the crowdsourced labels from the INEX 2010 Book Search track to compare the MV and EM methods. We consider crowdsourced labels from two task designs that lead to different level of noise and observe their effect on estimating relevance judgments and system rankings. In Section 5 we summarize the results and conclude with a discussion of future research directions.

2 Related Work

Relevance assessment errors in IR test collections have been considered by the IR community since the early Cranfield experiments [4]. Voorhees [5] studied the effects of variability in relevance judgments on the stability of comparative IR systems evaluation. She considered three sets of relevance judgments for the TREC-4 test collection and observed a significant disagreement among them. She explored the effects of judgments on the ranking of the systems that participated in TREC-4 and observed no significant changes in the systems ranking. This was attributed to the stability of the average precision (AP) metric [6] that was used to evaluate the systems' performance. Indeed, AP is calculated based on *deep pools* of documents obtained from the participating systems. Thus, a few incorrect judgments in a ranked list do not significantly affect the values of AP and, therefore, do not perturb the ordering of the systems. In our experiments (in Section 4) we confirm that a deep pool of judged documents can reduce the effect of noisy crowdsourced labels in the systems evaluation.

Recent trends in IR evaluations involve the use of large numbers of topics to enhance the reliability of the evaluation [7] while reducing the pool depths and, with

that, the cost of acquiring relevance judgments [8]. However, the use of the AP metric with shallow document pools becomes more sensitive to assessment errors and leads to significant changes in systems rankings [9]. This has motivated studies of the factors that cause assessment errors such as the level of assessors' expertise [10], the presentation of the documents for assessment, such as the sequence in which the documents are shown to the assessors ([1],[11]), and the assessors' behavior [9].

Awareness of the assessment errors has further increased with the use of crowdsourcing services to supplement or replace the traditional ways of collecting relevance judgments. In crowdsourcing, the relevance assessment task is expressed in terms of a *human intelligence task* (HIT) that is presented to crowd workers through a crowdsourcing platform to solicit their engagement, typically for a specified fee. The effectiveness of the crowdsourcing approach has been investigated in terms of various factors, including (*i*) the agreement with relevance judgments from trusted assessors [3], (*ii*) quality assurance techniques for detecting and removing unreliable workers [1], and (*iii*) the cost incurred due to redundant relevance assessments that are needed for quality assurance, e.g., [12] and [13].

The use of multiple labels per document to improve the quality of relevance judgments involves label aggregation across the assessors, e.g., by arriving at a consensus through majority voting ([2], [14]). The effectiveness of the consensus approach has been assessed by Kazai et al. [1] for IR tasks involving TREC and INEX test collections. Kumar and Lease [15] investigated the relationship between the document relevance and the workers' accuracy by using a set of documents with known relevance as training data for a naïve Bayes method. The trained model estimated the relevance of new documents by aggregating labels based on worker accuracy.

In this paper, we expand the existing body of research by applying the EM method from [19] to infer the relevance of documents from multiple, possibly noisy labels. In contrast to [15], we assume that no authoritative relevance assessments are available and estimate both the accuracy of the workers and the document relevance from the crowdsourced labels. Similar probabilistic models have been used in other research areas. For instance, Kasneci et al. [16] used a Bayesian probabilistic model in knowledge extraction systems to infer the relationships between entities from the input of multiple assessors. Welinder and Perona [17] proposed a probabilistic model for labeling images using crowd workers. Ipeirotis et al. [18] take a similar approach to identify systematic errors made by workers in the crowdsourcing experiments. Our aim is to infer true relevance judgments from crowdsourced labels and use them for evaluation of IR systems.

3 Aggregating Multiple Labels

In this section we describe the MV and EM methods that we use to aggregate multiple relevance labels from crowd workers and estimate both the true relevance judgments and the reliability of the workers.

Consider a set of N documents and a set of M workers that provide relevance labels for the documents. We assume that the relevance of a document is a discrete variable

with values in $\{0,1,\dots,G\}$. If the relevance value of a document i is k ($k \in \{0,1,\dots,G\}$), then its G+1 dimensional vector R_i is a binary vector with *k-th* component 1 and the rest 0, i.e., R_{ik}=1 and R_{ij}=0, ($\forall j \neq k$). We now define a matrix R of all the relevance vectors as $R \in \{0,1,\dots,G\}^{N\times(G+1)}$, comprising N binary R_i vectors.

Now consider a set of M workers with the corresponding accuracies $A = \{a_1, a_2, \dots, a_M\}$, where a_j represents the accuracy of the worker j. Both the document relevance R and the workers accuracy A are unknown to us. Instead, we have a set of relevance labels provided by the workers, i.e., $l_{ij} \in \{0,1,\dots,G\}$ is a relevance assessment of the document i by the worker j. A worker may provide relevance labels for some or all the documents. Our goal is to estimate the true relevance value of the documents and the accuracy of the workers' assessments from a given set of labels L. We assume that each document receives at least one label and the accuracy of the labels is unknown. Thus, in contrast to [15], we assume no initial information regarding the workers' accuracy or the relevance of the documents.

3.1 Majority Voting

Consider a document i with the corresponding labels provided by a set of workers. Let n_{ig} be the number of times document i is labeled as g, $g \in \{0,1,\dots,G\}$ by a set of workers. The majority voting assigns g as the document's true relevance label if n_{ig} is maximum. This technique is commonly used in IR experiments ([1], [2], [3]).

3.2 Concurrent Estimation of Relevance and Accuracy

As an alternative to MV we consider the EM method for concurrent estimation of the document relevance and the workers accuracy. In this model the document relevance R and the workers' accuracy A are unknown variables and the labels L provided by the workers are the observed data.

We take the same approach as [19] and consider the label aggregation model that assigns to each worker a $(G+1) \times (G+1)$ *latent confusion matrix* where G+1 is the number of different relevance grades. Each row refers to the true relevance value and each column refers to a relevance value assigned by a worker. Once the confusion matrix is calculated, we can determine the worker's expertise based on metrics such as accuracy, the true positive ratio and the true negative ratio [14].

Let π_{kl}^{j} , ($\forall\ k\ \&\ l \in \{0,\dots,G\}$) be the probability that the worker j provides a label l given that k is the true relevance value of an arbitrary document. The probability π_{kl}^{j} is computed based on the confusion matrix for the worker j. One estimator of π_{kl}^{j} is:

$$\pi_{kl}^{j} = \frac{\text{number of times worker } \boldsymbol{j} \text{ provides label } \boldsymbol{l} \text{ while the correct label is } \boldsymbol{k}}{\text{number of labels provided by worker } \boldsymbol{j} \text{ for documents of relevance } \boldsymbol{k}} \qquad (1)$$

where

$$\sum_{l=0}^{G} \pi_{kl}^{j} = 1 \qquad (\forall\ k \in \{0,\dots,G\}, and\ j \in \{1,\dots,M\}).$$

Of course, the calculation of π_{kl}^{j} assumes that R is known. In the following we show how π_{kl}^{j} and R can be simultaneously estimated.

Concurrent Estimations of R and $\boldsymbol{\pi_{kl}^{j}}$. Let p_k be the probability that a document drawn at random has a true relevance grade of k ($p_k = Pr[R_{ik} = 1]$; $i \in \{1, \dots, N\}$). Now let n_{il}^{j} be the number of times worker j provides label l for document i; for our purpose n_{il}^{j} is binary, so if a worker labels the document $n_{il}^{j} = 1$ otherwise $n_{il}^{j} = 0$. If g is the true relevance grade of document i, R_{ig}=1, then the probability of the worker j giving a grade l is π_{gl}^{j} and the probability of doing so n_{il}^{j} times is $\left(\pi_{gl}^{j}\right)^{n_{il}^{j}}$. Thus, the number of labels of each grade $\{0,1, \dots, G\}$ provided by worker j is distributed according to a multinomial distribution and its likelihood is proportional to

$$\Pr(n_{i0}^{j}, \dots, n_{iG}^{j};\ \pi_{g0}^{j}, \dots, \pi_{gG}^{j} | R_{ig} = 1) \propto \prod_{l=0}^{G} \left(\pi_{gl}^{j}\right)^{n_{il}^{j}}. \tag{2}$$

Under the assumption that M workers independently label documents, the likelihood of labels provided for document i when $R_{ig} = 1$ is also proportional to

$$\prod_{\mathrm{j}=1}^{\mathrm{M}} \Pr(n_{i0}^{j}, \dots, n_{iG}^{j};\ \pi_{g0}^{j}, \dots, \pi_{gG}^{j} | R_{ig} = 1) \propto \prod_{j=1}^{M} \prod_{l=0}^{G} \left(\pi_{gl}^{j}\right)^{n_{il}^{j}}$$

Since the value of g is unknown, we compute the expectation of $\Pr(n_{i0}^{j}, \dots, n_{iG}^{j};\ \pi_{g0}^{j}, \dots, \pi_{gG}^{j})$ over all possible values of g, i.e., we compute the marginal probability over all possible values of g:

$$\sum_{k=0}^{G} p_k \prod_{j=1}^{M} \prod_{l=0}^{G} \left(\pi_{kl}^{j}\right)^{n_{il}^{j}}. \tag{3}$$

Also as the data from all documents are assumed to be independent, the joint probability distribution over all N documents is

$$\prod_{i=1}^{N} \left(\sum_{k=0}^{G} p_k \prod_{j=1}^{M} \prod_{l=0}^{G} \left(\pi_{kl}^{j}\right)^{n_{il}^{j}} \right). \tag{4}$$

Equation (4) comprises mixtures of multinomial distributions. In order to estimate the quantities of interest, p_k, π_{kl}^{j} and R_{ig}, we apply the expectation maximization (EM) [19]. In the EM algorithm we treat π_{kl}^{j} and $p_k$as model parameters and R_{ik} as missing data. The EM algorithm then involves the following steps:

1. Initialize R_{ik} values, e.g., randomly choose g and set $R_{ig} = 1$, and $R_{ik} = 0\ (\forall\ k \neq g)$.
2. Given the current estimate of R_{ik}, compute the *maximum likelihood* estimates of π_{kl}^{j} and p_k, as

$$\hat{\pi}_{kl}^{j} = \frac{\sum_{i=1}^{N} R_{ik}\, n_{il}^{j}}{\sum_{l=0}^{G}\sum_{i=1}^{N} R_{ik}\, n_{il}^{j}} \; ; \qquad \hat{p}_k = \frac{\sum_{i=1}^{N} R_{ik}}{N}.$$

3. Calculate the new estimate of R_{ig} ($\forall g \in \{1, \ldots, G\}$) based on $\hat{\pi}_{kl}^{j}$ and $\hat{p}_k$, as

$$\Pr\left(R_{ig} = 1 | n_{i1}^{\forall j}, \ldots, n_{iG}^{\forall j};\, \pi_{g1}^{\forall j}, \ldots, \pi_{gG}^{\forall j}\right) = \frac{p_g \prod_{j=1}^{M}\prod_{l=0}^{G}\left(\pi_{gl}^{j}\right)^{n_{il}^{j}}}{\sum_{k=0}^{G} p_k \prod_{j=1}^{M}\prod_{l=0}^{G}\left(\pi_{kl}^{j}\right)^{n_{il}^{j}}}. \tag{5}$$

4. Repeat steps 2 and 3 until the results converge.
5. Finally, for each document *i,* set $R_{ig} = 1$ for the g with the maximum probability as calculated in equation (5), and $R_{ik} = 0$ ($\forall\, k \neq g$).

Note that by combining $\hat{\pi}_{kl}^{j}$ values we can compute the accuracy of the worker j or other statistics of interest, e.g., the true positive ratio. Accuracy is estimated as $\hat{a}_j \cong \frac{\sum_{l=0}^{G} \hat{\pi}_{ll}^{j}}{\sum_{l,k} \hat{\pi}_{lk}^{j}}$.

4 Experiments

In this section we describe a set of experiments that compare the aggregation of relevance labels based on the MV and EM methods and the implications for the IR systems evaluation. The experiments are based on both synthetic and crowdsourcing data collected for INEX 2010 Book Search evaluation track[1].

In the first experiment we use synthetic data and simulate the characteristics of the MV and EM methods. In the second experiment we assess the performance of the two methods based on crowdsourcing data. We then assess the accuracy of the MV and EM relevance assessments relative to the INEX official judgments. In the third experiment we investigate the impact of MV and EM relevance judgments on the system ranking using several performance metrics.

4.1 Experiment Design

In our experiments we use the test collection and crowdsourced relevance data from the INEX 2010 Book Search evaluation track [1]. The test collection comprises 50,239 books containing over 17 million scanned pages and 21 search topics with 169 judged pages per topic, on average. This amounts to 3,557 judged pages that serve as a gold standard set for IR systems evaluation. Each page is assigned a relevance judgment based on four grades {0,1,2,3}. In our experiments we assume that the relevance is binary and collapse labels {1,2,3} into label 1.

Crowdsourcing Experiments. Crowdsourced labels were collected for a subset of the test collection using the Mechanical Turk platform. The specific INEX task was

[1] http://www.inex.otago.ac.nz/tracks/books/books.asp

prove it: for a given search query, the user had to confirm whether the presented book page contains an answer to the search question. A search query and corresponding pages were presented to the crowd workers for relevance judgments in the form of HITs (Human Intelligence Tasks). Each HIT consisted of 10 pages including up to 3 pages judged as relevant by the INEX assessors. Two HIT designs, referred to as *simple* HIT and *full* HIT design, were used to control the workers' behavior and with that the label accuracy.

The *simple* HIT design included a minimal quality control using a single test question to capture random assignment of relevance labels by a worker. Furthermore, all the HITs were presented to the worker in a single batch, using the same generic HIT title, description, and keyword.

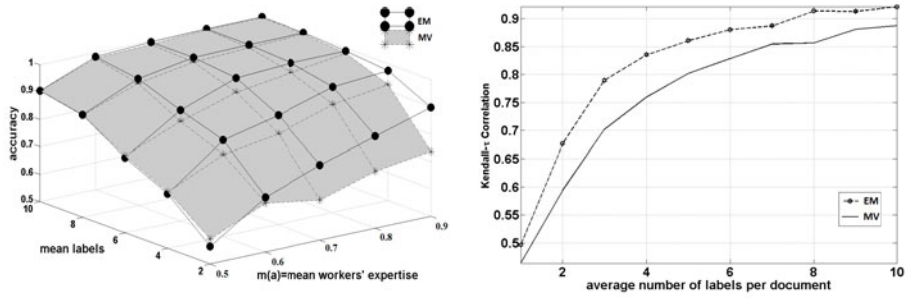
The *full* HIT design included several quality controls and qualified workers at different stages of the task. Since the HIT titles have an effect on the workers' recruitment, the full HITs were grouped into 21 topic-specific batches and included topic details in the title, description, and keywords. This was likely to attract workers interested in and knowledgeable about a particular topic. Each HIT included two test questions to detect sloppy behaviour: "Please tick here if you did NOT read the instructions" at the top of the HIT form and "I did not pay attention" as a relevance label option. Furthermore, to enforce the requirement that the workers needed to read a page before deciding about its relevance, a captcha was included asking them to enter the first word of the sentence that confirmed or refuted the relevance of the page.

On average, 6 labels from distinct workers were collected per document, 3 labels by simple HITs and 3 labels by full HITs. That amounts to 2179 labels for 727 topic-document pairs from simple HITs and 2060 labels for 683 topic-document pairs from full HITs. Also, 98% of topic-document pairs labelled in the full HIT were among those labelled in the simple HIT. The workers were paid $0.25 to complete a simple HIT and $0.50 for a full HIT.

For evaluation of the relevance labels obtained by the MV and EM methods we consider three commonly used measures [14]: (*i*) the *accuracy*—the proportion of judged documents that are assigned the correct relevance label, (*ii*) the *true positive ratio* (TPR)—the proportion of judged relevant documents that are correctly assigned the 'relevant' label, and (*iii*) the *true negative ratio* (TNR)—the proportion of judged non-relevant documents that are correctly assigned the 'non-relevant' label.

4.2 Simulation

We conduct simulations of multiple label aggregations to investigate the effects of (*i*) the number of labels collected for a document, and (*ii*) the level of the workers' expertise on the performance of the MV and EM methods. We consider a set of 1000 hypothetical documents with associated true relevance judgments. We also consider a set of 100 hypothetical workers, each with a particular level of expertise. We define workers' expertise as their accuracy of labeling a randomly chosen document. Similarly to [9], we randomly sample the workers' expertise from a Beta distribution and randomly assign documents to workers. We then apply the MV and EM methods to the collected labels in order to estimate the relevance of the documents. We use the measures defined in Section 4.1 to assess the performance of the two methods.



(a) (b)

Fig. 1. (a) Comparison of the judgment accuracy for MV and EM on synthetic data for varying number of labels per document and different levels of workers accuracy. (b) Kendall-τ correlation between the true and estimated workers expertise. Workers expertise is drawn from a beta distribution with the mean of 0.7.

We repeat the simulations by varying (*i*) the mean $m(a)$ of the Beta distribution from which a worker's expertise is drawn, and (*ii*) the average number of labels collected per document. The results are shown in Figure 1(a) where the workers' mean expertise varies between 0.5 and 0.9 and the average number of labels varies between 1 and 10.

It can be seen that, when the workers average expertise is nearly random, $m(a)$=0.5, and the average number of labels per document is only 2, both methods exhibit poor accuracy. As the number of labels or the level of expertise increases, the performance of both methods improves. When the number of labels per document is small, e.g., 2 or 4 labels, but the workers average expertise is increased to 0.6 or higher, the EM method outperforms the MV. Finally, as the number of labels approaches 10 and the workers average expertise increases to 1, both methods obtain perfect accuracy. The simulations clearly show that the EM approach generally performs the same or better than MV.

We also assessed the performance of the MV and EM methods in estimating the workers accuracy. We use the gold standard set to determine the workers' true accuracy and compare with the estimated accuracies based on the relevance labels from the MV and EM methods. We compute Kendall-τ between the workers ranking induced by the EM or MV relevance judgments and the ranking based on gold standard set. Figure 1(b) shows that for the set of workers with $m(a)$=0.7, EM outperforms MV for a range of labels per document. We observed similar results for $m(a)$=0.6, 0.8, and 0.9.

4.3 Relevance Agreement with INEX Judgments

We apply the MV and EM methods to the labels collected from the two crowdsourcing experiments and compare the derived relevance judgments with the INEX gold standard set. We provide results for three sets of relevance judgments derived from: (*i*) the labels from the simple HITs, (*ii*) labels from the full HITs, and (*iii*) combined

labels from both sets of HITs. For each of the sets we use samples of 1000, 1500, or 2000 labels to estimate the document relevance. The samples are randomly selected but guarantee that at least one label per document is included. We report average performance of the methods over 10 random trials.

Table 1. Comparison of MV and EM relevance judgments based on (*i*) accuracy, (*ii*) true positive ratio (TPR) and (*iii*) true negative ratio (TNR). INEX 2010 relevance judgements are used as the gold standard set. Statistically significant differences are marked by ǂ.

HIT	~No. of Labels	accuracy		TPR		TNR	
		MV	EM	MV	EM	MV	EM
Simple	1000	0.58	0.64ǂ	0.52	0.67ǂ	0.60	0.64ǂ
	1500	0.62	0.69ǂ	0.54	0.76ǂ	0.65	0.68ǂ
	2000	0.69	0.75ǂ	0.58	0.79ǂ	0.70	0.74ǂ
Full	1000	0.68	0.72ǂ	0.72	0.88ǂ	0.71	0.72
	1500	0.73	0.79ǂ	0.78	0.90ǂ	0.76	0.78ǂ
	2000	0.80	0.82	0.86	0.93ǂ	0.84	0.82
Simple+Full	1000	0.66	0.76ǂ	0.61	0.85ǂ	0.62	0.67ǂ
	1500	0.70	0.78ǂ	0.66	0.88ǂ	0.69	0.74ǂ
	2000	0.71	0.81ǂ	0.69	0.91ǂ	0.75	0.79ǂ

Note in case when there is an equal number of votes for relevant and non-relevant labels, the MV method will declare the associated document as non-relevant. We make the same assumption when the EM method estimates the relevance of a document as 0.5. This decision is based on the fact that the number of non-relevant documents for a query is typically higher than the number of relevant documents. The experimental results are shown in Table 1 for each of the three evaluation measures, the accuracy, TPR, and TNR. Statistically significant differences in the performance of the two methods are identified using a two sample z-test with a significance level of p=0.05.

As seen in Table 1, for the labels from the simple HIT task, the EM method significantly outperforms MV across all the samples and evaluation measures. The average improvement of EM over MV is 0.06 in accuracy, 0.19 in TPR, and 0.04 in TNR.

For the full HIT labels, the performance improvement of EM over MV is significant for most of the measures across the three samples. Only in three instances did we not get statistically significant differences. The average performance improvement of EM across the samples is smaller than for the simple HIT: 0.04 in accuracy, 0.12 in TPR, and 0.003 on TNR. This is expected since the labels from the full HITs are of higher quality due to more elaborate quality assurances tests. Indeed, there is 70% agreement between the full HIT labels and the INEX official judgments compared to 55% for the labels from the simple HITs.

This observation is consistent with the simulation results in Section 4.2: when the labels are provided by quality workers and the number of labels is large, both MV and

EM perform well. Indeed this is confirmed by the accuracy scores for the full HITs in Table 1. When the number of labels is 1000 or 1500, the accuracy of EM is significantly higher than that of MV. However, for a larger sample of 2000 labels there is no significant difference in the accuracy scores.

Table 2. Kendall-τ scores for MV and EM rankings of 10 runs from the INEX 2010 Book Search track by using the precision at 5 cut-off levels

HIT	P@10		P@20		P@30		P@50		P@100	
	MV	EM	MV	EM	MV	EM	MV	EM	MV	EM
Simple	0.45	0.62	0.55	0.71	0.77	0.89	0.91	0.98	0.90	0.99
Full	0.68	0.76	0.71	0.80	0.88	0.93	1.00	1.00	0.99	0.99

Table 3. Kendall-τ scores for MV and EM rankings of 10 runs from the INEX 2010 Book Search track. The average precision (AP) is calculated over all available judgments. stat-AP is calculated for the subsets of documents using corresponding relevance judgments.

HIT	statAP						AP	
	10%		30%		50%			
	MV	EM	MV	EM	MV	EM	MV	EM
Simple	0.58	0.67	0.64	0.77	0.91	0.91	0.84	0.91
Full	0.66	0.72	0.67	0.79	0.80	0.89	0.79	0.87
INEX	0.95		0.96		1.0			

Finally, we consider combined labels from the simple and full HITs. For each sample size 50% of labels are randomly selected from the simple HITs labels and 50% from the full HITs labels. For all three samples and performance measures, the EM method shows statistically significant improvements over MV. The average improvement across sample sizes is 0.09 in accuracy, 0.22 in TPR, and 0.05 in TNR. This is a larger improvement than for the simple HITs labels.

4.4 Impacts on Systems Ranking

We now observe the effect of MV and EM relevance judgments on the system ranking. For the crowdsourced labels collected from the simple and full HITs we apply MV and EM methods to create final sets of relevance judgments. These sets are used to rank the performance of 10 retrieval runs from the systems that participated in the INEX 2010 *prove it* task. We compare the runs based on the achieved precision at the top 10, 20, 30, 50 and 100 ranked documents. For consistency, when calculating the performance of runs based on INEX judgments, we consider only relevance judgments for the documents involved in the crowdsourcing experiments.

Table 2 summarizes the Kendall-τ correlations between the ranking of runs based on the INEX official judgments and the ranking obtained from MV or EM relevance judgments. We see for all cut-off levels, the rank correlation is higher for EM than for

MV. The average improvement for EM across the cut-off levels is 0.12 for simple HITs and 0.04 for the full HITs labels.

Generally, we see a considerable effect of the cut-off levels on Kendall-τ. This is expected since, when the cut-off level is small, e.g. p@10, even a few misjudged documents represents a high percentage of error and therefore significantly affects the ranking. As the cut-off level increases, for the same number of misjudged documents the percentage of error is relatively smaller and the ranking is not as affected.

We also explore the impacts of MV and EM on systems ranking when the average precision (AP) is used to evaluate the systems performance. The result is shown in Table 3. Once again EM outperforms MV for both the simple HITs and the full HITs labels. We investigate the effects of MV and EM on measuring AP with a shallow pool of documents, which is a common practice in IR experiments. We use statAP [8] to select subsets of 10%, 30% or 50% of documents labeled by the crowd workers. We apply MV and EM to the labels and then use the statAP metric to estimate the AP scores. For each sample size we calculate statAP based on the corresponding INEX judgments, MV judgments, and EM judgments and obtain systems rankings. In Table 3 we show the correlation between INEX systems ranking and the MV or EM systems rankings. The last row in Table 3 shows the Kendall-τ scores between the INEX systems ranking based on the statAP and the AP scores with the full set of the INEX judgments. Generally, we note that, as the sample size increases from 10% to 50%, the Kendall-τ scores increase correspondingly, similarly to the results in Table 2.

5 Summary and Concluding Remarks

In this paper, we consider the problem of creating relevance judgments using crowdsourcing experiments to collect multiple, possibly noisy relevance labels for documents. We assume that the workers' judgments are varied and of unknown accuracy. We also assume that the true relevance labels for documents are not available. We compare two methods for inferring document relevance from multiple labels. The MV method treats all the workers equally and assigns the relevance label that has received the most votes. The EM method simultaneously infers document relevance and workers' accuracy. We conduct a series of simulations with synthetic data and experiments with crowdsourced labels from the INEX 2010 Book Search track. Our experiments show that the relevance judgments inferred by the EM method are the better estimations of true document relevance and lead to more accurate systems ranking. EM performance improvements over MV are particularly noticeable when judgments are noisy and the number of relevance labels is small.

This research can be extended in several directions. In the evaluation of system performance we exploited the aggregation of noisy labels. However, the EM method provides estimation of the workers accuracy which can be used to grade workers and optimize the quality of additional labels by carefully selecting crowd worker. Furthermore, it can be used to compute workers' pay based on the quality of their work. Furthermore, our experiments were focused on the binary relevance judgments while the model supports multi-grade relevance. Thus, the future experiments will investi-

gate the performance of the MV and EM methods for graded relevance and the sensitivity of the graded metrics, e.g., nDCG, to noise. Finally, the full potential of the EM method could be realized through an iterative model of selecting workers and collecting relevance labels. Thus, it is beneficial to extend the crowdsourcing experiments and evaluate the dynamic and real time collection of relevance judgments.

References

[1] Kazai, G., Kamps, J., Koolen, M., Milic-Frayling, N.: Crowdsourcing for Book Search Evaluation: Impact of HIT Design on Comparative System Ranking. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information, pp. 205–214 (2011)

[2] Alonso, O., Mizzaro, S.: Can we get rid of TREC assessors? Using Mechanical Turk for relevance assessment. In: SIGIR 2009: Workshop on the Future of IR Evaluation, Boston (2009)

[3] Smucker, M.D., Jethani, C.P.: Measuring assessor accuracy: a comparison of nist assessors and user study participants. In: Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, pp. 1231–1232 (2011)

[4] Cuadra, C.A., Katter, R.V.: Opening the Black Box of 'Relevance'. Journal of Documentation 23(4), 291–303 (1967)

[5] Voorhees, E.: Variations in relevance judgments and the measurement of retrieval effectiveness. Inf. Process. Manage. 36(5), 697–716 (2000)

[6] Buckley, C., Voorhees, E.: Evaluating evaluation measure stability. In: Proceedings of SIGIR, pp. 33–40 (2000)

[7] Carterette, B., Pavlu, V., Kanoulas, E., Aslam, J.A., Allen, J.: Evaluation Over Thousands of Queries. In: Proceedings of SIGIR, pp. 651–658 (2008)

[8] Carterette, B., Soboroff, I.: The effect of assessor error on IR system evaluation. In: Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 539–546 (2010)

[9] Aslam, J.A., Pavlu, V., Yilmaz, E.: A Statistical Method for System Evaluation Using Incomplete Judgments. In: Proceedings of SIGIR, pp. 541–548 (2006)

[10] Bailey, P., et al.: Relevance assessment: are judges exchangeable and does it matter. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 667–674 (2008)

[11] Scholer, F., Turpin, A., Sanderson, M.: Quantifying Test Collection Quality Based on the Consistency of Relevance Judgements. In: Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1063–1072 (2011)

[12] Winter, M., Duncan, W.: Financial incentives and the "performance of crowd". In: Proceedings of the ACM SIGKDD Workshop on Human Computation, pp. 77–85 (2009)

[13] Snow, R., O'Connor, B., Urafsky, D., Ng, A.Y.: Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Stroudsburg, PA, USA, pp. 254–263 (2008)

[14] Smucker, M.D., Jethani, C.P.: The Crowd vs. the Lab: A Comparison of Crowd-Sourced and University Laboratory Participant Behavior. In: Proceedings of the SIGIR 2011 Workshop on Crowdsourcing for Information Retrieval, Beijing (2011)

[15] Kumar, A., Lease, M.: Modeling Annotator Accuracies for Supervised Learning. In: WSDM 2011 Workshop on Crowdsourcing for Search and Data Mining, Hong Kong (2011)
[16] Kasneci, G., Gael, J.V., Stern, D.H., Graepel, T.: CoBayes: bayesian knowledge corroboration with assessors of unknown areas of expertise. In: Proceedings of the Forth International Conference on Web Search and Web Data Mining, pp. 465–474 (2011)
[17] Welinder, P., Perona, P.: Online crowdsourcing: rating annotators and obtaining cost-effective labels. In: CVPR 2010: IEEE Conference on Computer Vision and Pattern, pp. 1526–1534 (2010)
[18] Ipeirotis, P.G., Provost, F., Wang, J.: Quality Management on Amazon Mechanical Turk. In: Proceedings of the ACM SIGKDD Workshop on Human Computation, pp. 64–67 (2010)
[19] Dawid, P., Skene, A.M.: Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm. Applied Statistics 28(1), 20–28 (1979)
[20] Bernstein, Y., Zobel, J.: Redundant documents and search effectiveness. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pp. 736–743 (2005)