Chapter 15

A SURVEY OF GRAPH MINING FOR WEB APPLICATIONS

Debora Donato
Yahoo! Research
Avd Diagonal 177, Barcelona, Spain
debora@yahoo-inc.com

Aristides Gionis
Yahoo! Research
Avd Diagonal 177, Barcelona, Spain
gionis@yahoo-inc.com

Abstract Graph structures provide a general framework for modeling entities and their relationships, and they are routinely used to describe a wide variety of data such as the Internet, the web, social networks, metabolic networks, protein-interaction networks, food webs, citation networks, and many more. In recent years, there has been an increasing amount of literature on studying properties, models, and algorithms for graph data. In this chapter we provide a brief overview of graph-mining algorithms for web and social-media applications. We review a wide range of algorithms, such as those for estimating reputation and popularity of items in a network, mining query logs and performing query recommendations. The main goal of the chapter is to provide the reader with an understanding of how graph structural mining algorithms can be exploited in the context of web applications. This highlights the challenges of, and provides an understanding of the power of graph mining in the context of web and social-media applications.

Keywords: Graph Mining, Link Mining, Web Mining, Social Network Analysis, World Wide Web, Query-Log Mining, Query Recommendation

1. Introduction

Graph mining has been widely used to study relationships among various types of entities. Real-world graphs are also referred to as *networks*, and the interactions between the entities represented in the networks are modeled as *links*. The problems of studying the properties of real-world networks, designing algorithms for mining such networks, and developing applications on top of network data has been of increasing interest in the past few years. This has led to the birth of a very active area of scientific research, which is known as *analysis of complex networks* [7, 16, 55].

One of the most pervasive properties of real-world networks is the emergence of *power-law* distributions that tend to characterize many of networks statistical properties [6, 26]. Power laws have intrigued the interest of researchers, who have proposed various models that attempt to explain the presence of power-law distributions in real graphs. For examples of such models, see [6, 25, 40].

In this chapter, we deviate from the classical exposition of properties and generative models for complex networks, and we focus on graph-mining applications that appear in the context of the web and social-media. Such graphs include data that model the interaction of users in a social network. For example, this may correspond to comments of users in a blog, user activity in a question-answering portal, or query-log data that summarize the interaction of users with a search engine. Understanding the structure of such graphs, modeling the complex interactions between entities, and designing algorithms for leveraging the latent knowledge (also known as *the wisdom of the crowds*) in those graphs introduces new challenges in the field of graph mining. One important difference with networks that have been previously studied, is that in social-media and web-usage graphs the links represent many different types of interactions and activities among nodes. For instance in a question-answering portal, users ask questions, answer questions for other users, vote for favorite answers, interesting questions, assign answers to categories of a hierarchy, and much more. Hence graphs from such applications are characterized by having different types of nodes and high degree of heterogeneity in the types of interactions among nodes. Consequently, algorithms and methodologies widely applied in the web and other complex networks have to be adapted to this new multifaceted scenario, which allows for the different meanings that are implicitly or explicitly captured by each link.

This chapter is organized as follows. In Section 2 we briefly introduce measures and algorithms that have been extensively used as basic tools for graph mining. Then we focus on two different areas of graph mining in the context of social-media and web applications. In Section 3, we review techniques for identifying items of high quality in social-media networks. We discuss two concrete examples: (1) predicting the number of citations of authors in a bibliographic data set, and (2) finding high-quality items in a question answering system. In both cases, the examples rely on adapting link-mining algorithms for computing authoritativeness scores in linked environments. In Section 4 we discuss algorithms for mining graph structures that represented information collected in the query logs of search engines. We first discuss various graph representations of query logs, and then discuss how to use these representations in order to perform the task of query recommendation. The conclusions are presented in Section 5.

2. Preliminaries

An undirected graph $\mathcal{G} = (V, E)$ consists of a set of nodes V, also called vertices, and a set E of pairs of distinct nodes, which are called edges or arcs. A directed graph, or digraph, is distinguished from the undirected version by the fact that its edges are *ordered pairs* of nodes. In an undirected graph, the *degree* of a node is the number of edges incident to it. For a directed graph, we define the in-degree and the out-degree of a node to be the number of incoming and out-going edges, respectively.

In an undirected graph $\mathcal{G}$, a set of nodes S forms a *connected component* (CC), if for every pair of nodes $u, v \in S$ there exists a path from u to v (which is also a path from v to u). In a directed graph $\mathcal{G}$, a set of nodes S forms a *strongly connected component* (SCC), if for every pair of nodes $u, v \in S$, there exists a (directed) path from u to v, and a path from v to u. A set of nodes S forms a *weakly connected component* (WCC), if and only if the set S is a connected component in the undirected graph $\mathcal{G}_u$ that is obtained by ignoring the directionality of the edges in $\mathcal{G}$.

Power laws and scale-free networks. Power-law distributions ubiquitously characterize real-world networks. We say that a discrete random variable X follows a power-law distribution if the probability distribution is defined for each discrete value k as follows:

$$\Pr[X = k] \propto k^{-\gamma}$$

The value γ is called the exponent of the power-law. We assume that $\gamma \geq 0$. Detailed surveys on power laws may be found in [45] and [46].

If a random variable X follows a power-law distribution, then we know that the conditional probability $\Pr[X \geq k \mid X \geq m]$ is the same as $\Pr[X \geq k]$. In other words, conditioning on the size does not yield any additional information. For this reason, networks that have attributes that follow a power-law distribution are also called *scale-free* networks.

Degree and Assortativeness. The degree of the nodes of a graph can be of great interest in social-media applications. The out-degree of a node might provide an indication of its capacity to influence his neighbors. This property is called *expansiveness* [58]. On the other hand, the in-degree is the most straightforward measure for the *popularity* of each node in the network. Complex networks exhibit large variance in the values of their degrees: very few nodes have the capacity of attracting a large fraction of links while the largest majority of nodes are connected to the network by few in-coming and outgoing links.

Significant insight on the nature of the graph can be obtained by measuring the correlation between the degrees of adjacent vertexes [47]. This is also referred to as *assortative mixing*. Complex networks can be divided into three types based on the value of their mixing coefficient r: (i) *disassortative* if $r < 0$; (ii) *neutral* if $r \approx 0$; and (iii) *assortative* if $r > 0$. An alternative way to identify assortative or disassortative network is by using the average degree $E[k_{nn}(k)]$ of a neighboring vertex of a vertex with degree k [47]. As k increases, the expectation $E[k_{nn}(k)]$ increases for an assortative network and decreases for a disassortative one. In particular, a power-law equation $E[k_{nn}(k)] \approx k^{-\gamma}$ is satisfied, where γ is negative for an assortative network and positive for a disassortative one [49]. Social networks such as friendship networks are mostly assortative mixed, but technological and biological networks tend to be disassortative [62]. "Assortative mating" is a well-known social phenomenon that captures the likelihood that marriage partners will share common background characteristics, whether it is income, education, or social status. In online activity networks such as question-answering portals and newsgroups, the degree correlation provides information about user tendency to provide help. Such kind of networks are neutral or slightly disassortative: active users are prone to contribute without considering the expertise or the involvements of the users searching for help [63, 20].

Centrality and prestige. A key issue in social network analysis is the identification of the most important or prominent nodes. The measure of *centrality* captures whether a node is involved in a high number of ties regardless the directionality of the edges. Various definitions of centrality have been suggested. For instance, the *closeness centrality* is just the degree of a node eventually normalized by the number of all nodes V in the network. Two alternative measures of centrality are the *distance centrality* and the *betweenness centrality*. The closeness centrality $\mathcal{D}_c$ of a node u is the average distance of u to the rest of the nodes in the graph:

$$\mathcal{D}_c(u) = \frac{1}{|V|-1} \sum_{v \neq u} d(u, v),$$

where $d(u, v)$ is the shortest-path distance between u and v. Similarly, the betweenness centrality $\mathcal{B}_c$ of a node u is the average number of shortest paths that pass through u:

$$\mathcal{B}_c(u) = \sum_{s \neq u \neq t} \frac{\sigma_{st}(u)}{\sigma_{st}},$$

where $\sigma_{st}(u)$ is the number of shortest paths from the node s to the node t that pass through node u, and σ_{st} is the total number of shortest paths from s to t.

A different concept for identifying important nodes is the measure of *prestige*, which exclusively considers the capacity of the node to attract incoming links, and ignores the capacity of initiating any outgoing ties. The basic intuition behind the prestige definition is the idea that a link from node u to node v denotes endorsement. In its simplest form, the prestige of a node is defined to be its in-degree, but there are other alternative definitions of prestige [58]. This concept is also at the core of a number of link analysis algorithms, an issue which we will explore in the next section.

2.1 Link Analysis Ranking Algorithms

PageRank. Although we can view the existence of a link between two pages as an endorsement of authority from the former to the latter, the in-degree measure is a rather superficial way to examine page authoritativeness. This is because such a measure can easily be manipulated by creating spam pages which point to a particular target page in order to improve its authority. A smarter method of assigning authority score to a node is by using the *PageRank* algorithm [48], which uses the authoritative information of both the source and target page in an iterative way in order to determine the rank. The PageRank algorithm models the behavior of a "random surfer" on the Web graph. The surfer essentially browses the documents by following hyperlinks randomly. More specifically, the surfer starts from some node arbitrarily. At each step the surfer proceeds as follows:

- With probability α an outgoing hyperlink is selected randomly from the current document, and the surfer moves to the document pointed by the hyperlink.
- With probability $1 - \alpha$ the surfer jumps to a random page chosen according to some distribution. This distribution is typically chosen to be the uniform distribution.

The value $\mathrm{Rank}(i)$ of a node i (called the PageRank value of node i) is the fraction of time that the surfer spends at node i. Intuitively, $\mathrm{Rank}(i)$ is a measure of the importance of node i.

PageRank is expressed in matrix notation as follows. Let N be the number of nodes of the graph and let $n(j)$ be the out-degree of node j. We define the square matrix M as one in which the entry $M_{ij} = \frac{1}{n(j)}$ if there is a link from node j to node i. We define the square matrix $\left[\frac{1}{N}\right]$ of size $N \times N$ that has all entries equal to $\frac{1}{N}$. This matrix models the uniform distribution of jumping to a random node in the graph. The vector Rank stores the PageRank values that are computed for each node in the graph. A matrix M' is then derived by adding transition edges of probability $\frac{1-\alpha}{N}$ between every pair of nodes to include the case of jumping to a random node of the graph.

$$M' = \alpha M + (1-\alpha)\left[\frac{1}{N}\right]$$

Since the PageRank algorithm computes the stationary distribution of the random surfer, we have $M'\mathrm{Rank} = \mathrm{Rank}$. In other words, Rank is the principal eigenvector of the matrix M', and thus it can be computed by the power-iteration method [15].

The notion of PageRank has inspired a large body of research on designing improved algorithms for more efficient computation of PageRank [24, 54, 36, 42], and for providing alternative definitions that can be used to address specific issues in search, such as personalization [27], topic-specific search [12, 32], and spam detection [8, 31].

One disadvantage of the PageRank algorithm is that while it is superior to a simple indegree measure, it continues to be prone to adversarial manipulation. For instance, one of the methods that owners of spam pages use to boost the ranking of their pages is to create a large number of auxiliary pages and hyperlinks among them, called *link-farms*, which result in boosting the PageRank score of certain target spam pages [8].

HITS. The main intuition behind PageRank is that authoritative nodes are linked to by other authoritative nodes. The HITS algorithm, proposed by Jon Kleinberg [38], introduced a double-tier paradigm for measuring authority. In the HITS framework, every page can be thought of as having a hub and an authority identity. There is a mutually reinforcing relationship between the two: a good hub is a page that points to many good authorities, while a good authority is a page that is pointed to by many good hubs.

In order to quantify the quality of a page as a hub and as an authority, Kleinberg associated every page with a hub and an authority *score*, and he proposed the following iterative algorithm: Assuming n pages with hyperlinks among them, let $\boldsymbol{h}$ and $\boldsymbol{a}$ denote n-dimensional hub and authority score vectors. Let also W be an $n \times n$ matrix, whose (i, j)-th entry is 1 if page i points to page j and 0 otherwise. Initially, all scores are set to 1. The algorithm iteratively updates the hub and authority scores sequentially one after the other and vice-versa. For a node i, the authority score of node i is set to be the sum of the hub scores of the nodes that point to i, while the hub score of node i is the authority score of the nodes pointed by i. In matrix-vector terms this is equivalent to setting $\boldsymbol{h} = W\boldsymbol{a}$ and $\boldsymbol{a} = W^T\boldsymbol{h}$. A normalization step is then applied, so that the vectors $\boldsymbol{h}$ and $\boldsymbol{a}$ become unit vectors. The vectors $\boldsymbol{a}$ and $\boldsymbol{h}$ converge to the principal eigenvectors of the matrices W^TW and WW^T, respectively. The vectors $\boldsymbol{a}$ and $\boldsymbol{h}$ correspond to the right and left *singular vectors* of the matrix W.

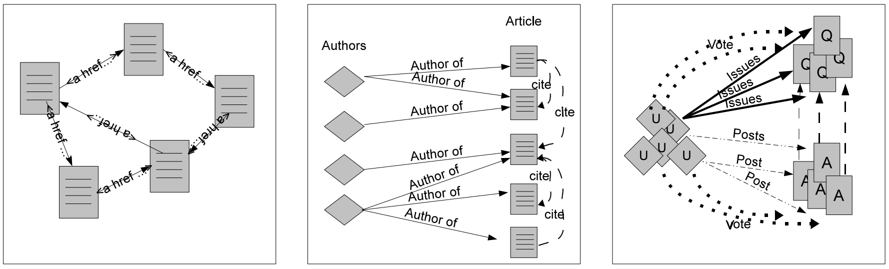
Given a user query, the HITS algorithm determines a set of relevant pages for which it computes the hub and authorities scores. Kleinberg's approach obtains such an initial set of pages by submitting the query to a text-based search engine. The pages returned by the search engine are considered as a root set, which is consequently expanded by adding other pages that either point to a page in the root set or are pointed by a page in the root set.

Kleinberg showed that additional information can be obtained by using more eigenvectors, in addition to the principal ones. Those additional eigenvectors correspond to clusters or distinct topics associated with the user query. One important characteristic of the HITS algorithm is that it computes page scores that depend on the user query: one particular page might be highly authoritative with respect to one query, but not such an important source of information with respect to another query. On the other hand, it is computationally expensive to compute eigenvectors for each query. This makes the algorithm computationally demanding. In contrast, the authority scores computed by the PageRank algorithm are not query-sensitive, and thus, they can be computed in a preprocessing stage.

3. Mining High-Quality Items

Online expertise-sharing communities have recently become extremely popular. The online media that allow the spread of this enormous amount of knowledge can take many different forms: users are sharing their knowledge in blogs, newsgroups, newsletters, forums, wikis, and question/answering portals. Those social-media environments can be represented as graphs with nodes of different types and with various types of relations among nodes. In the rest of the section we describe particular characteristics of the graphs arising in social-media environments, and their importance in driving the graph-mining process.

There are two main factors that differentiate social media from the traditional Web: (i) content-quality variance and (ii) interaction multiplicity. Differently from the traditional Web, in which the content is mediated by professional publishers, in social-media environments the content is provided by users. The massive contribution of users to the system leads to a high variance in the distribution of the quality of available content. With everyone able to create content and share any single opinion and thought, Thus the problem of determining items of high quality in an environment of excessive content is



(a) Single Item: Single Relation Model (b) Double Item: Double Relation Model (c) Multiple Items: Multiple Relation Model

Figure 15.1. Relation Models for Single Item, Double Item and Multiple Items

one of the most important issues to be solved. Furthermore, filtering out and ranking relevant items is more complex than in other domains.

The second aspect that must be considered is the wide variety of types of nodes, of relations among such nodes, and of interactions among users. For instance, the PageRank and HITS algorithms considers a simple graph model with one type of nodes (documents) and one type of edges (hyperlinks), see Figure 15.1(a).

On the other hand, social media are characterized by much more heterogeneous and rich structure, with a wide variety of user-to-document relation types and user-to-user interactions. In Figure 15.1(b) is shown the structure of a citation network as CiteSeer [21]. In this case, nodes can be of two types: `author` and `article`. Edges can also be of two types, `is-an-author-of` between a node of type `author` and a node of type `article`, and `cites` between two nodes of type `article`.

A more complex structure can be found in a question-answering portal, such as Yahoo! Answers [61], a graphical representation of which is shown in Figure 15.1(c). The main types of nodes are the following:

- `user`, representing the users registered with the system; they can act as askers or answerers, and can vote or comment questions and answers provided by other users,
- `question`, representing the questions asked by the users,
- `answer`, prepresenting the answers provided by the users.

Potential interesting research questions to ask for this type of application are the following: (i) find items of high-quality, (ii) predict which items will become successful in the future (assuming a dynamic environment), (iii) identify experts on a particular topic.

As in the case of other social-media applications, the variance of content quality in Yahoo! Answers is very high. According to Su et al. [56], the number of correct answers to specific questions varies from 17% to 45%, meanwhile the number of questions with at least one good answer is between 65% and 90%.

When a higher number of nodes and relations are involved, the features that can be exploited for developing successful ranking algorithms become notably more complex. Algorithms based on *single-item* models may still be profitably used, provided that the underlying multi-graphs can be projected on a single dimension. The results obtained at each projection provide a multifaceted set of features that can be profitably used for tuning automatic classifiers able to discern high-quality items, or to identify experts.

In the rest of this chapter we detail a methodology for mining multi-item multi-relation graphs for two particular study cases. In the first case we describe the methodology presented in [18] for predicting successful items in a co-citation network, while in the second case we report the work of Agichtein et al. [2] for determining high-quality items in a question-answering portal.

3.1 Prediction of Successful Items in a Co-citation Network

Predicting the impact that a book or an article might have on readers is of great interest for publishers and editors for the purpose of planning marketing campaigns or deciding the number of copies to print. This problem was addressed in [18], where the authors present a methodology to estimate the number of citations that an article will receive, which is one measure of impact in a scientific community. The data was extracted by the large collection of academic articles made publicly available by CiteSeer [21] through an Open Archives Initiative (OAI) interface.

The two main objects in bibliometric networks are authors and papers. A bibliographic network can be modeled by a graph $\mathcal{G} = (V_a \cup V_p, E_a \cup E_c)$, where (i) V_a represents the set of authors, (ii) V_p represents the set of the papers, (iii) $E_a \subseteq V_a \times V_p$ represents the edges that express which author has written which paper, and (iv) $E_c \subseteq V_p \times V_p$ represents the edges that express which paper cites which. To model the dynamics of the citation network, different snapshots can be considered, with $\mathcal{G}_t = (V_{t,a} \cup V_{t,p}, E_{a,t} \cup E_{t,c})$ representing the snapshot at time t. The set of edges $E_{a,t}$ and $E_{c,t}$ can also be represented by matrices $P_{a,t}$ and $P_{c,t}$ respectively.

One way to model the network is by assigning a dual role to each author: in one role, an author produces original content (i.e., as authorities in the Kleinberg model. In the other role, an author provides an implicit evaluation of other authors (i.e., as a hub) with the use of citations. Fujimura and Tanimoto [29] present an algorithm, called *EigenRumor*, for ranking object and users when they act in this dual role. In their framework, the authorship relation $P_{a,t}$ is called *information provisioning*, while the citation relation $P_{c,t}$ is called *infor-*

mation evaluation. One of the main advantages of the EigenRumor algorithm is that the relations implied by both information provisioning and information evaluation are used to address the problem of correctly ranking items produced by sources that have been proven to be authoritative, even if the items themselves have not still collected a high number of in-links. The EigenRumor algorithm has been proposed in order to overcome the problem of algorithms like PageRank, which tend to favor items that have been present in the network for a period of time long enough to accumulate many links.

For the task of predicting the number of citations of a paper, Castillo et al. [18] use supervised learning methods that rely on features extracted from the co-citation network. In particular, they propose to exploit features that determine popularity, and then to train a classifier. Three different types of features are extracted: (1) *a priori* author-based features, (2) *a priori* link-based features, and (3) *a posteriori* features.

- **A priori author-based features.** These features capture the popularity of previous papers of the same authors. At time t, the past publication history of a given author a can be expressed in terms of:

  (i) Total number of citations $\mathrm{C}_t(a)$ received by the author i from all the papers published before time t.

  (ii) Total number of papers $\mathrm{M}_t(a)$ published by the author a before time t
  $$\mathrm{M}_t(a) = |\,\{p|(a,p) \in E_a \wedge \mathrm{time}(p) < t\}\,|.$$

  (iii) Total number of coauthors $\mathrm{A}_t(a)$ for papers published before time t
  $$\mathrm{A}_t(a) = |\,\left\{a'|(a',p) \in E_a \wedge (a,p) \in E_a \wedge \mathrm{time}(p) < t \wedge a' \neq a\right\}\,|$$

  Given that one paper can have multiple authors, the previous three kinds of features are aggregated. For each, we consider the maximum, the average and the sum over all the co-authors of each paper.

- **A priori link-based features.** These features are based on the intuition that mutual reinforcement characterizes the relation between citing and cited authors: good authors are probably aware of the best previous articles written in a certain field, and hence they tend to cite the most relevant of them. As mentioned previously, the EigenRumor algorithm [29] can be used for ranking objects and users.

  The reputation score of a paper p is denoted by $\mathbf{r}(p)$. The authority and the hub values of the author a are denoted by $\mathbf{a}_t(a)$ and $\mathbf{h}_t(a)$ respectively. The EigenRumor algorithm is formalized as follows:

- $\mathbf{r} = P_{a,t}^T \mathbf{a}_t$ expresses the fact that good papers are likely to be written by good authors,
- $\mathbf{r} = P_{c,t}^T \mathbf{h}_t$ expresses the fact that good papers are likely to be cited by good authors,
- $\mathbf{a}_t = P_{a,t}\mathbf{r}$ expresses the fact that good authors usually write good papers,
- $\mathbf{h}_t = P_{c,t}\mathbf{r}$ expresses the fact that good authors usually cite good papers.

Combining the previous equations with a mixing parameter α, gives the following formula for the score vector:

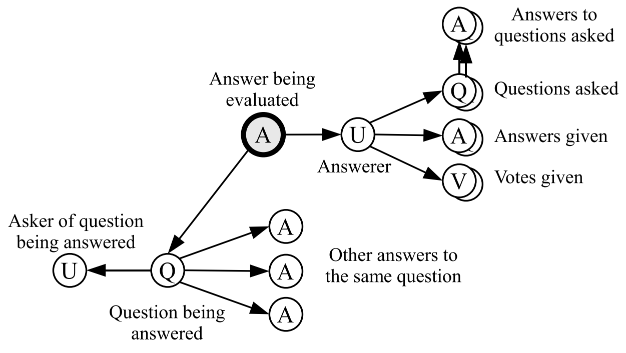
$$\mathbf{r} = \alpha P_{a,t}^T \mathbf{a}_t + (1-\alpha) P_{c,t}^T \mathbf{h}_t.$$

- **A posteriori features.** These features are simply used to count the number of citations of a paper at the end of a few time intervals that are much shorter than the target time for the prediction that has to be made.

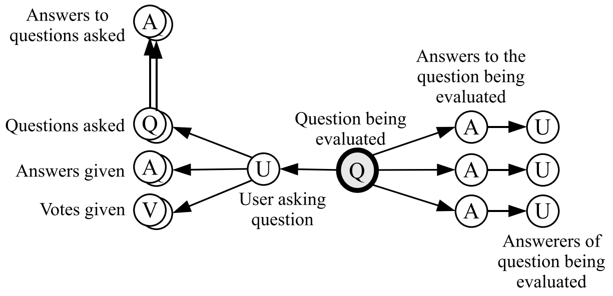
With respect to the case in which only *a posteriori* citations are used, *a priori* information about the authors helps in predicting the number of citations it will receive in the future. It is worth noting that a priori information about authors degrades quickly. When the features describing the reputation of an author are calculated at a certain time, and re-used without taking into account the last papers the author has published, the predictions tend to be much less accurate. These results are even more interesting if the reader considers that many other factors can be taken into consideration. For instance, the venue where the paper was published is related to the content of the paper itself.

3.2 Finding High-Quality Content in Question-Answering Portals

Yahoo! Answer is one of the largest question-answering portals, where users can issue question and find answers. Questions are the central elements. Each question has a life cycle. After it is “opened”, it can receive answers. When the person who has asked the question is satisfied by an answer or after the expiration of an automatic timer, the question is considered “closed”, and can not receive any other answers. However, the question and the answers can be voted on by other users. The question is “resolved” once a best answer is chosen. Because of its extremely rich set of user-document relations, Yahoo! Answers has recently been the subject of much research [1, 2, 11]. In [2], the authors focus on the task of finding high quality items in social networks and they use Yahoo! Answers as cases of study. The general approach is similar to the one used in the previous case for predicting successful items in co-citation networks, i.e., exploiting features that are correlated with quality in social media and then training a classifier to select and weight features for this task. In


Answers to questions asked
Answer being evaluated
Questions asked
Answers given
Answerer
Votes given
Asker of question being answered
Other answers to the same question
Question being answered

(a) Features for Inferring Answer Quality


Answers to questions asked
Answers to the question being evaluated
Questions asked
Question being evaluated
Answers given
Votes given
User asking question
Answerers of question being evaluated

(b) Features for Inferring Question Quality

Figure 15.2. Types of Features Available for Inferring the Quality of Questions and Answers

the remainder of this section, the features for quality classification are considered. As in the previous case, three different types of features are used: (1) intrinsic content quality features, (2) link-based (or relation-based) features, and (3) content usage statistics.

- **Intrinsic content quality features.** For text-based social media the intrinsic content quality is mainly related with the text quality. This can be measured using *lexical*, *syntactic* and *semantic* features.

  Lexical features include word length, word and phrase frequencies, and the average number of syllables in the words.

  All the word n-grams up to length 5 that appear in the documents more than 3 times are used as syntactic features.

  Semantic features try to capture (1) the *visual quality* of the text (i.e., ignored capitalization rules, excessive punctuation, spacing density,etc.), (2)*semantic complexity* (i.e., entropy of word length, readability measures [30, 43, 37], etc.) and (3) *grammaticality* (i.e., features that try to capture the correctness of grammatical forms, etc).

In the QA domain, additional features are required to explicitly model the relationship between the question and the answer. In [2] such a relation was modeled using the KL-divergence between the language models of the two texts, their non-stopword overlap, the ratio between their lengths, and other similar features.

- **Link-based features.** As mentioned earlier, Yahoo! Answers is characterized by nodes of multiple types (e.g., questions, answers and users) and interactions with different semantics (e.g., "answers", "votes for", "gives a star to", "gives a best answer"), that are modeled using a complex multiple-node multiple-relations graph. Traditional link-analysis algorithms, including HITS and PageRank, are proven to still be useful for quality classification whether applied to the projections obtained from the graph $\mathcal{G}$ considering one type of relation at the time.

  Answer features. In Figure 15.2(a), the relationship data related to a particular answer are shown. These relationships form a *tree*, in which the type "Answer" is the root. Two main subtrees start from the answer being evaluated: one related to the question `Q` being answered, and the other related to the user `U` contributing the answer.

  By following paths through the question subtree, it is also possible to derive features `QU` about the questioner, or features `QA` concerning the other answers to the same question. By following paths through the user subtree, we can derive features `UA` from the answers of the user, features `UQ` from questions of the user, features `UV` from the votes of the user, and features `UQA` from answers received to the user's questions.

  Question features. Figure 15.2(b) represents user relationships around a question. Again, there are two subtrees: one related to the asker of the question, and the other related to the answers received. The types of features on the answers subtree are: features `A` directly from the answers received and features `AU` from the answerers of the question being answered. The types of features on the user subtree are the same as the ones above for evaluating answers.

  Implicit user-user relations To apply link-analysis algorithms, it is necessary to consider the user-user graph. This is the graph $G = (V, E)$ in which the set of vertices V is composed of the set of users and the set $E = E_a \cup E_b \cup E_v \cup E_s \cup E_+ \cup E_-$ represents the relationships between users as follows:

  - E_a represents the answers: $(u, v) \in E_a$ if user u has answered at least one question asked by user v.

- E_b represents the best answers: $(u, v) \in E_b$ if user u has provided at least one best answer to a question asked by user v.
- E_v represents the votes for best answer: $(u, v) \in E_v$ if user u has voted for best answer at least one answer given by user v.
- E_s represents the stars given to questions: $(u, v) \in E_v$ if user u has given a star to at least one question asked by user v.
- E_+/E_- represents the thumbs up/down: $(u, v) \in E_+/E_-$ if user u has given a "thumbs up/down" to an answer by user v.

For each graph $G_x = (V, E_x)$, h_x is the vector of hub scores on the vertices V, a_x the vector of authority scores, and p_x the vector of PageRank scores. Moreover p'_x is the vector of PageRank scores in the transposed graph.

To classify these features in our framework, PageRank and authority scores are assumed to be related mostly to in-links, while the hub score deals mostly with out-links. For instance, let us consider h_b. It is the hub score in the "best answer" graph, in which an out-link from u to v means that u gave a best answer to user v. Then, h_b represents the answers of users, and is assigned to the record (UA) of the person answering the question.

- **Content usage statistics.** Usage statistics such as the number of clicks on the item and time spent on the item have been shown useful in the context of identifying high quality web search results. These are complementary to link-analysis based methods. Intuitively, usage statistics measures are useful for social media content, but require different interpretation from the previously studied settings.

  In the QA settings, it is possible to exploit the rich set of metadata available for each question. This includes temporal statistics, e.g., how long ago the question was posted, which allows us to give a better interpretation to the number of views of a question. Also, given that clickthrough counts on a question are heavily influenced by the topical and genre category, we also use derived statistics. These statistics include the expected number of views for a given category, the deviation from the expected number of views, and other second-order statistics designed to normalize the values for each item type. For example, one of the features is computed as the click frequency normalized by subtracting the expected click frequency for that category, divided by the standard deviation of click frequency for the category.

The conclusion of Agichtein et al. [2] from analyzing the above features, is that many of the features are complementary and their combination enhances the robustness of the classifier. Even though the analysis was based on a particular question-answering system, the ideas and the insights are applicable to other social media settings, and to other emerging domains centered around user contributed-content.

4. Mining Query Logs

A query log contains information about the interaction of users with search engines. This information can be characterized in terms of the queries that users make, the results returned by the search engines, and the documents that users click in the search results. The wealth of explicit and implicit information contained in the query logs can be a valuable source of knowledge for a large number of applications. Examples of such applications include the following:

- (i) analyzing the interests of users and their searching behavior,
- (ii) finding semantic relations between queries (which terms are similar to each other or which one is a specialization of another) allowing to build taxonomies that are much richer than any human-built taxonomy,
- (iii) improving the results provided by search engines by analysis of the documents clicked by users and understanding the user information needs,
- (iv) fixing spelling errors and suggesting related queries,
- (v) improving advertising algorithms and helping advertisers select bidding keywords.

As a result of the wide range of applications which work with query-logs, considerable research has recently been performed in this area. Many of these papers discuss related problems such as analyzing query logs and on addressing various data-mining problems which work off the properties of the query-logs. On the other hand, query logs contain sensitive information about users and search-engine companies are not willing to release such data in order to protect the privacy of their users. Many papers have demonstrated the security breaches that may occur as a result of the release of query-log data even after anonymization operations have been applied and the data appears to be secure [34, 35, 41]. Nevertheless, some query log data that have been carefully anonymized have been released to the research community [22], and researchers are working actively on the problem of anonymizing query logs without destroying the utility of the released data. Recent advances on the anonymization problem are discussed in Korolova et al. [39]. Because of the wide range of knowledge embedded in query logs, this area is a central problem for the entire research community, and is not restricted to researchers working on problems related to search engines. Because of the natural ability to construct graph representations of query-log data, the graph mining area is particularly related to problems associated with query-log mining. In the next sections, we discuss graph representations of query log data, and consequently we present techniques for mining and analyzing the resulting graph structures.

4.1 Description of Query Logs

Query log. A typical query log $\mathcal{L}$ is a set of records $\langle q_i, u_i, t_i, V_i, C_i \rangle$, where q_i is the submitted query, u_i is an anonymized identifier for the user who submitted the query, t_i is a timestamp, V_i is the set of documents returned as results to the query, and C_i is the set of documents clicked by the user. We denote by Q, U, and D the set of queries, users, and documents, respectively. Thus, we have $q_i \in Q$, $u_i \in U$, and $C_i \subseteq V_i \subseteq D$.

Sessions. A *user query session*, or just *session*, is defined as the sequence of queries of one particular user within a specific time limit. More formally, if t_θ is a timeout threshold, a user query session S is a *maximal* ordered sequence

$$S = \big\langle \, \langle q_{i_1}, u_{i_1}, t_{i_1} \rangle, \ldots, \langle q_{i_k}, u_{i_k}, t_{i_k} \rangle \, \big\rangle,$$

where $u_{i_1} = \cdots = u_{i_k} = u \in U$, $t_{i_1} \leq \cdots \leq t_{i_k}$, and $t_{i_{j+1}} - t_{i_j} \leq t_\theta$, for all $j = 1, 2, \ldots, k-1$. The typical timeout threshold used for splitting sessions in query log analysis is $t_\theta = 30$ minutes [13, 19, 50, 57].

Supersessions. The temporally ordered sequence of all the queries of a user in the query log is called a *supersession*. Thus, a supersession is a sequence of sessions in which consecutive sessions are separated by time periods larger than t_θ.

Chains. A chain is a topically coherent sequence of queries of one user. Radlinski and Joachims [53] defined a chain as *"a sequence of queries with a similar information need"*. For instance, a query chain may contain the following sequence of queries [33]: "`brake pads`"; "`auto repair`"; "`auto body shop`"; "`batteries`"; "`car batteries`"; "`buy car battery online`". Clearly, all of these queries are closely related to the concept of car-repair. The concept of chain is also referred to in the literature with the terms *mission* [33] and *logical session* [3]. Unlike the straightforward definition of a session, chains involve relating queries based on an *analysis* of the user information need. This is a very complex problem, since it is based on an analysis of the information need, rather than in a crisp way, as in the case of a session. We do not try to give a formal definition of chains here, since this is beyond the scope of the chapter.

4.2 Query Log Graphs

Query graphs. In a recent paper about extracting semantic relations from query logs, Baeza-Yates and Tiberi define a graph structure derived from the query log. This takes into account not only the queries of the users, but also the actions of the users (clicked documents) after submitting their queries [4]. The analysis of the resulting graph captures different aspects of user behavior and topic distributions of what people search in the web. The graph representation introduced in [4] allows us to infer interesting semantic relationships among queries. This can be used in many applications.

The basic idea in [4] is to start from a weighted query-click bipartite graph, which is defined as the graph that has all distinct queries and all distinct documents as two partitions. We define an edge (q, u) between query q and document d, if a user who has submitted query q has clicked on document d. Obviously, d has to be in the result set of query q. The bipartite graph that has queries and documents as two partitions is also called the *click graph* [23]. Baeza-Yates and Tiberi define the *url cover* $\mathrm{uc}(q)$ of a query q to be the set of neighbor documents of q in the click graph. The weight $w(q, d)$ of the edge (q, d) is defined to be the fraction of the clicks from q to d. Therefore, we have $\sum_{d \in \mathrm{uc}(q)} w(q, d) = 1$. The url cover $\mathrm{uc}(q)$ can be viewed as a vector representation for the query q, and we can then define the similarity between two queries q_1 and q_2 to be the *cosine similarity* of their corresponding url-cover vectors. This is denoted by $\cos(\mathrm{uc}(q_1), \mathrm{uc}(q_2))$. The next step in [4] is to define a graph G_q among queries, where the weight between two queries q_1 and q_2 is defined by their similarity value $\cos(\mathrm{uc}(q_1), \mathrm{uc}(q_2))$.

Using the url cover of the queries, Baeza-Yates and Tiberi define the following semantic relationship among queries:

- **Identical cover:** $\mathrm{uc}(q_1) = \mathrm{uc}(q_2)$. Those are undirected edges in the graph G_q, which are denoted as *red* edges or edges of type I. These imply that the two queries q_1 and q_2 are equivalent in practice.

- **Strict complete cover:** $\mathrm{uc}(q_1) \subset \mathrm{uc}(q_2)$. Those are directed edges, which are denoted as *green* edges or edges of type II. These imply that q_1 is more specific than q_2.

- **Partial complete cover:** $\mathrm{uc}(q_1) \cap \mathrm{uc}(q_2) \neq \emptyset$ and none of the previous two conditions are fulfilled. These are denoted as *black* edges or edges of type III. They are the most common edges and exist due to multi-topic documents or related queries, among other reasons.

The authors of [4] also define relaxed versions of the above concepts. In particular, they define α-red edges and α-green edges, when equality and inclusion hold with a slackness factor of α.

The resulting graph is very rich and may lead to many interesting applications. The mining tasks can be guided both by the semantic relationships of the edges as well as the graph structure. Baeza-Yates and Tiberi demonstrate an application of finding multi-topic documents. The idea is that edges with low weight are most likely caused by multi-topic documents e.g., e-commerce sites to which many different queries may lead. Thus, low-weight edges are considered as voters for the documents shared by the two corresponding queries. Documents are sorted according to the number of votes they received: the more votes a document gets, the more multitopical it is. Then the multi-topic documents may be removed from the graph (on a basis of a threshold value) and a new graph of better quality can be computed.

As Baeza-Yates and Tiberi point out, the analysis described in their paper is only the tip of the iceberg, and the potential number of applications of query graphs is huge. For instance, in addition to the graph defined in [4], Baeza-Yates [3] identifies five different types of graphs whose nodes are queries, and an edge between two queries implies that: (i) the queries contain the same word(s) (*word graph*), (ii) the queries belong to the same session (*session graph*), (iii) users clicked on the same urls in the list of their results (*url cover graph*), (iv) there is a link between the two clicked urls (*url link graph*) (v) there are l common terms in the content of the two urls (*link graph*).

Random walks on the click graph. The idea of representing the query log information as a bipartite graph between queries and documents (where the edges are weighted according to the user clicks) has been extensively used in the literature. Craswell and Szummer [23] study a random-walk model on the click graph, and they suggest using the resulting probability distribution of the model for ranking documents to queries. As mentioned in [23], query-document pairs can be considered as "soft" (positive) relevance judgments. These are however are noisy and sparse. The noise is due to the fact that users judge from short summaries and might not click on relevant documents. The sparsity problem is due to the fact that the users may not click on relevant documents. When a large number of documents are relevant, users may click on only a small fraction of them. The random-walk model can be used to reduce the amount of noise and it also alleviates the sparseness problem. One of the main benefits of the approach in [23] is that relevant documents to a query can be ranked highly even if no previous user has clicked on them for that query.

The click-graph can be used in many applications. Some of the applications discussed by Craswell and Szummer in [23] are the following:

- **Query-to-document search.** The problem is to rank relevant documents for a given ad-hoc query. The click graph is used to find documents of high quality and relevant documents for a query. Such documents may not necessarily be easy to determine using pure content-based analysis.

- **Query-to-query suggestion.** Given a query of a user, we want to find other queries that the user might be interested in. The role of the click-graph is determine other relevant queries in the "proximity" of the input query. Examples of finding such related queries can be found in [9, 59].

- **Document-to-query annotation.** The idea is that a query can be used as a concise description of the documents that the users click for that query, and thus queries can be used to represent documents. Studies have shown that the use of such a representation can improve web search [60]. It can be used for other web mining applications [51].

- **Document-to-document relevance feedback.** For this application, the task is to find relevant documents for a given target document, and are also relevant for a user.

The random walk on the click graph models a user who issues queries, clicks on documents according to the edge weights of the graph. These documents inspire the user to issue new queries, which in turn lead to new documents and so on. More formally, we define $\mathcal{G} = (Q \cup D, E)$ is the click graph, with Q and D being the set of queries and documents. We define E being the set of edges, the weight C_{jk} of an edge (j, k) is the number of clicks in the query log between nodes j and k. The weights are then normalized to represent the transition probabilities at the t-th step of the walk. The transition probabilities are defined as follows:

$$\Pr_{t+1|t}[k \mid j] = \begin{cases} (1-s)\frac{C_{jk}}{\sum_i C_{ji}}, & \text{if } k \neq j, \\ s, & \text{if } k = j. \end{cases}$$

In other words, a self-loop is added at each node. The random walk is performed by traversing the nodes of the click graph according to the probabilities $\Pr_{t+1|t}[k \mid j]$.

Let $\mathbf{A}$ be the adjacency-matrix of the graph, whose (j, k)-th entry is $\Pr_{t+1|t}[k \mid j]$. Then, if $\mathbf{q}_j$ is a unit vector with an entry equal to 1 at the j-th position and all other entries equal to 0, the probability of a transition from node j to node k in t steps is $\Pr_{t|0}[k \mid j] = [\mathbf{q}_j \mathbf{A}^t]_k$. The notation $[\mathbf{u}]_i$ refers to the i-th entry of vector $\mathbf{u}$. The random-walk models that are typically used in the literature, such as PageRank and much more, consider *forward walks*, and exploit the property that the resulting vector of visiting probabilities $[\mathbf{q}\mathbf{A}^t]$ converges to a fixed distribution. This is the stationary distribution of the random walk, as $t \to \infty$, and is independent of the vector of initial probabilities $\mathbf{q}$. The value $[\mathbf{q}\mathbf{A}^t]_k$, i.e., the value of the stationary distribution at the k-th node, is usually interpreted as the importance of node k in the random walk, and it is used as the score for ranking node k.

Craswell and Szummer consider the idea of running the random walk *backwards*. Essentially the question is which is the probability that the walk started at node k given that after t steps is at node j. Bayes' law gives

$\mathrm{Pr}_{0|t}[k \mid j] \propto \mathrm{Pr}_{t|0}[j \mid k]\,\mathrm{Pr}_0[k]$, where $\mathrm{Pr}_0[k]$ is a *prior* of starting at node k and it is usually set to the uniform distribution, i.e., $\mathrm{Pr}_0[k] = 1/N$. To see the difference between forward and backward random walk, notice that since the stationary distribution of the forward walk is independent from the initial distribution, the limiting distribution of the backward random walk is uniform. Nevertheless, according to Craswell and Szummer, running the walk backwards for a small number of steps (before convergence) gives meaningful differentiation among the nodes in the graph. The experiments in [23] confirm that for ad-hoc search in image databases, the backward walk gives superior precision results than the forward random walk.

Random surfer and random querier. While the classic PageRank algorithm simulates a *random surfer* on the web, the random-walk on the click graph simulates the behavior of a *random querier*: moving between queries and documents according to the clicks of the query log. Poblete et al. [52] observe that searching and surfing the web are the two most common actions of web users, and they suggest building a model that combines these two activities by means of a random walk on a *unified* graph: the union of the hyperlink graph with the click graph.

The random walk on the unified graph is described as follows: At each step, the user selects to move at a random query or a random document with probability $1-\alpha$. With probability α, the user makes a step, which can be one of two types:

- with probability $1-\beta$ the user follows a link in the hyperlink graph,
- with probability β the user follows a link in the click graph.

The authors in [52] point out that combining the two graphs is beneficial, because the two graph structures are complementary and each of them can be used to alleviate the shortcomings of the other. For example, using clicks is a way to take into account user feedback, and this improves the robustness of the hyperlink graph to the degrading effects of link-spam. On the other hand, considering hyperlinks and browsing patterns increases the density and the connectivity of the click graph, and the model takes into account pages that users might visit *after* issuing particular queries.

The query-flow graph. We will now change the focus of the discussion to a different type of graphs extracted from query logs. In all our previous discussions, the graphs do not take into account the notion of *time*. In other words, the timestamp information from the query logs is completely ignored. However, if one wants to reason about the querying patterns of users, and the ways that user submit queries in order to achieve more complex information retrieval goals, one has to include the temporal aspect in the analysis of query logs.

In order to capture the querying behavior of users, Boldi et al. [13] define the concept of the *query-flow graph*. This is related to the discussion about sessions and chains at the beginning of this section. The query-flow graph G_{qf} is then defined to be directed graph $G_{\mathrm{qf}} = (V, E, w)$ where:

- the set of nodes is $V = Q \cup \{s, t\}$, i.e., the distinct set of queries Q submitted to the search engine and two special nodes s and t, representing a *starting state* and a *terminal state*. These can be interpreted as the begin and end of a chain;
- $E \subseteq V \times V$ is the set of directed edges;
- $w : E \to (0, 1]$ is a weighting function that assigns to every pair of queries $(q, q') \in E$ a weight $w(q, q')$ representing the probability that q and q' are part of the same chain.

Boldi et al. suggest a machine learning method for building the query-flow graph. First, given a query log $\mathcal{L}$, it is assumed that it has been split into a set of sessions $\mathcal{S} = \{S_1, \ldots, S_m\}$. Two queries $q, q' \in Q$ are *tentatively* connected with an edge if there is at least one session in $\mathcal{S}$ in which q and q' are consecutive. Then, for the tentative edges, the weights $w(q, q')$ are learned using a machine learning algorithm. If the weight of an edge is estimated to be 0, then the edge is removed. The features used to learn the weights $w(q, q')$ include *textual features* (such as the cosine similarity, the Jaccard coefficient, and size of intersection between the queries q and q', computed on on sets of stemmed words and on character-level 3-grams), *session features* (such as the number of sessions in which the pair (q, q') appears, the average session length, the average number of clicks in the sessions, the average position of the queries in the sessions, etc.), and *time-related features* (such as the average time difference between q and q' in the sessions in which (q, q') appears). Several of those features have been used in the literature for the problem of segmenting a user session into logical sessions [33]. For learning the weights $w(q, q')$, Boldi et al. use a rule-based model and 5 000 labeled pairs of queries as training data. Boldi et al. argue that the query-flow graph is a useful construct that models user querying patterns and can be used in many applications. One such application is that of query recommendations.

Another interesting application of the query-flow graph is segmenting and assembling chains in user sessions. In this particular application, one complication is that there is not necessarily some timeout constraint in the case of chains. Therefore, as an example, all the queries of a user who is interested in planning a trip to a far-away destination and web searches for tickets, hotels, and other tourist information over a period of several weeks should be grouped in the same chain. Additionally, for the queries composing a chain, it is not required to be consecutive. Following the previous example, the user who is planning the far-away trip may search for tickets in one day, then make some other queries related to a newly released movie, and then return to trip planning the next day by searching for a hotel. Thus, a session may contain queries from many chains. Conversely, a chain may contain queries from many sessions.

In [13] the problem of finding chains in query logs is modeled as an *Assymetric Traveling Salesman Problem* (ATSP) on the query-flow graph. The formal definition of the chain-finding problem is the following: Let $S = \langle q_1, q_2, \ldots, q_k \rangle$ be the supersession of one particular user. We assume that a query-flow graph has been built by processing a query log that includes S. Then, we define a *chain cover* of S to be a partition of the set $\{1, \ldots, k\}$ into subsets $C_1, \ldots, C_h$. Each set $C_u = \{i_1^u < \cdots < i_{\ell_u}^u\}$ can be thought of as a chain $C_u = \langle s, q_{i_1^u}, \ldots, q_{i_{\ell_u}^u}, t \rangle$, which is associated with probability

$$\Pr[C_u] = \Pr[s, q_{i_1^u}] \Pr[q_{i_1^u}, q_{i_2^u}] \ldots \Pr[q_{i_{\ell_u - 1}^u}, q_{i_{\ell_u}^u}] \Pr[q_{i_{\ell_u}^u}, t],$$

We would like to find a chain cover maximizing $\Pr[C_1] \ldots \Pr[C_h]$.

The chain-finding problem is then divided into two subproblems: *session reordering* and *session breaking*. The session reordering problem is to ensure that all the queries belonging to the same search session are consecutive. Then, the session breaking problem is much easier as it only needs to deal with non-intertwined chains.

The session reordering problem is formulated as an instance of the ATSP: Given the query-flow graph G_{qf} with edge weights $w(q, q')$, and given the session $S = \langle q_1, q_2, \ldots q_k \rangle$, consider the subgraph of G_{qf} induced by S. This is defined as the induced subgraph $G_S = (V, E, h)$ with nodes $V = \{s, q_1, \ldots, q_k, t\}$, edges E, and edge weights h defined as $h(q_i, q_j) = -\log \max\{w(q_i, q_j), w(q_i, t) w(s, q_j)\}$. The maximum of the previous expression is taken over the options of splitting and not splitting a chain. For more details about the edge weights of G_S, see [13]. An optimal ordering is a permutation π of $\langle 1, 2, \ldots k \rangle$ that maximizes the expression

$$\prod_{i=1}^{k-1} w(q_{\pi(i)}, q_{\pi(i+1)}).$$

This problem is equivalent to that of finding a Hamiltonian path of minimum weight in this graph.

Session breaking is an easier task, once the session has been re-ordered. It corresponds to the determination of a series of cut-off points in the re-ordered session. One way of achieving this is by determining a threshold η in a validation dataset, and then deciding to break a reordered session whenever $w(q_{\pi(i)}, q_{\pi(i+1)}) < \eta$.

4.3 Query Recommendations

As the next topic of graph mining for web applications and query-log analysis, we discuss the problem of *query recommendations*. Even though the problem statement does not involve graphs, many approaches in the literature work by exploring the graph structures induced from query logs. Examples of such graphs were discussed in the previous section.

The application of query recommendation takes place when search engines offer not only document results but also *alternative queries* in response to the queries they receive from their users. The purpose of those query recommendations is to help users locate information more effectively. Indeed, it has been observed over the past years that users are looking for information for which they do not have sufficient knowledge [10], and thus they may not be able to specify their information needs precisely. The recommendations provided by search engines are typically queries similar to the original one, and they are obtained by analyzing the query logs.

Many of the algorithms for making query recommendations are based on defining similarity measures among queries, and then recommending the most popular queries in the query log among the similar ones to a given query. For computing query similarity, Wen et al. [59] suggest using distance functions based on (i) the keywords or phrases of the query, (ii) string matching of keywords, (iii) the common clicked documents, and (iv) the distance of the clicked documents in some pre-defined hierarchy. Another similarity measure based on common clicked documents was proposed by Beeferman et al. [9]. Baeza-Yates et al. [5] argue that the distance measures proposed by the previous methods have practical limitations, because two related queries may output different documents in their answer sets. To overcome these limitations, they propose to represent queries as term-weighted vectors obtained by aggregating the term-weighted vectors of their clicked documents. Association rule mining has also been used to discover related queries in [28]. The query log is viewed as a set of transactions, where each transaction represents a session in which a single user submits a sequence of related queries in a time interval.

Next we review some of the query recommendation methods that are based on graph structures.

Hitting time. Mei et al. [44] propose a query recommendation method, which is based on the proximity of the queries on the *click graph*. Recall that the click graph is the bipartite graph that has queries and documents as two partitions, and the weight of an edge $w(q, u)$ indicates the number of times that document d has been clicked when query q was submitted. The main idea is based on the concept of structural proximity of specific nodes. When the user submits a query, the corresponding node is located in the click graph, and other recommendations are queries that are located in the *proximity* of the query node.

For a meaningful notion of distance between nodes in the click graph, Mei et al. suggest to use the notion of *hitting time*. The hitting time from a node u to a node v in a graph G is the expected number of steps taken when v is visited for a first time in a random walk starting from u. Hitting time captures not only nodes that are connected by short paths in the graph but also nodes that are connected by many paths. Therefore, it is a *robust* distance measure between graph nodes.

In addition, Mei et al. [44] propose an adaptation of their method that can provide personalized query suggestions. The idea is to adjust the weights of the edges of the click graph so that they can better model the preferences of the user for whom we want to provide a recommendation. Mei et al. observe that models for personalized web search provide estimates of a probability that a user clicks on a certain document. Thus, any personalized algorithm for web search can be combined with their hitting-time method in order to provide personalized recommendations.

Topical query decomposition. A different aspect of query recommendation is addressed by Bonchi et al. [14], who try to overcome a common limitation of many query recommendation algorithms. This limitation is that many of the recommendations are very similar to each other. Instead Bonchi et al. formulate a new problem, which they call *topical query decomposition*. In this new framework, the goal is to find a set of queries that cover different aspects of the original query. The intuition is that such a set of diverse queries can be more useful in cases when the query is too short (and thus imprecise and ambiguous), and it is hard to receive good recommendations based on the query-content only.

The problem statement of topical query decomposition is based again on the click graph. In particular, let q be a query and $D(q)$ be the result set of q, i.e., the neighbor nodes of q in the click graph. We denote with $\mathcal{Q}(q)$ the maximal set of queries p_i, where for each p_i, the set $D(p_i)$ has at least one document in common with the documents returned by q. In other words, we have

$$\mathcal{Q}(q) = \{p_i | \langle p_i, D(p_i)\rangle \in \mathcal{L} \wedge D(p_i) \cap D(q) \neq \emptyset\}.$$

The goal is to compute a *cover*, i.e., selecting a sub-collection $\mathcal{C} \subseteq \mathcal{Q}(q_i)$ such that it covers almost all of $D(q_i)$. As stated before, the queries in $\mathcal{C}$ should represent coherent, conceptually well-separated set of documents: they should have small overlap, and they should not cover too many documents outside $D(q_i)$.

Bonchi et al. propose two different algorithms for the problem of topical query decomposition. The first algorithm is a top-down approach, based on set covering. Starting from the queries in $Q(q)$, this approach tries to handle the problem as a special instance of the weighted set covering problem. The weight of each query in the cover is given by its internal topical coherence, the fraction of documents in $D(q)$, the number of documents it retrieves that are not in $D(q)$, as well as its overlap with other queries in the solution. The second algorithm is a bottom-up approach, based on clustering. Starting with the documents in $D(q)$, this approach tries to build clusters of documents which are compact in the topics space. Since the resulting clusters are not necessarily document sets associated with queries existing in the query log, a second phase is needed. In this phase, the clusters found in the first phase are "matched" to the sets that correspond to queries in the query log.

Query recommendations based on the query-flow graph. Boldi et al. [13] investigate the alternative approach of finding query recommendations using the query-flow graph instead of the click graph. A random walk approach is used in the this case, as in the approach of Mei et al. [44]. However, in this case, the recommended queries are selected on the basis of a PageRank measure instead of hitting time. We also allow *teleportation* (or jumps) to specific nodes during the random walks in order to bias the walk towards these nodes. In particular, given the query q, the method computes the PageRank values of a random walk on the query-flow graph where the teleportation is always at the node of the graph that corresponds to query q. In this way, queries that are close to q in the graph are favored to be selected as recommendations. The advantage of using the query-flow graph instead of the click graph is that the method favors as recommendations for q queries q' that follow q in actual user sessions. Thus, it is likely that q' are natural continuations of q in an information seeking task performed by users.

Boldi et al. [13] explore various alternatives to that of using random walk on the query-flow graph for the query recommendation problem. One interesting idea is to use normalized PageRank. Here, if $s_q(q')$ is the PageRank score for query q' on a random walk with teleportation to the original query q, instead of using the pure random-walk score $s_q(q')$, they consider the ratio $\hat{s}_q(q') = s_q(q')/r(q')$ where $r(q')$ is the absolute random-walk score of q' (i.e., the one computed using a uniform teleportation vector). The intuition behind this normalization is to avoid recommending very popular queries (like "ebay") that may easily get high PageRank scores even though they are not related with the original query. The experiments in [13] showed that in most cases $\hat{s}_q(q')$ produces rankings that are more reasonable, but sometimes tend to boost by too much the scores with low absolute value $r(q')$. To use a bigger denominator, they also tried dividing with $\sqrt{r(q')}$, which corresponds to the geometric mean between $s_q(q')$ and $\hat{s}_q(q')$.

Another interesting variant of the query-recommendation framework of Boldi et al. is providing recommendations that depend not only on the last query input by the user, but on some of the last queries in the user's history. This approach may help to alleviate the data sparsity problem. This is because the current query may be rare, but among the previous queries there might be queries for which we have enough information in the query flow graph. Basing the recommendation on the user's query history may also help to solve ambiguous queries, as we have more informative suggestions based on what the user is doing during the current session. To take the recent queries of the user into account, one has to modify the random walk, in order to perform the teleportation into the set of last queries, instead of only the one last query. For more details on the method and various examples of recommendations see [13].

Using both the click graph and session data. Finally, we discuss the query-recommendation approach of Cao et al. [17], which uses both the click graph and session data. As in the previous case of Boldi et al., the algorithm of Cao et al. has the advantage that it provides recommendations that are based on the few last queries of the user. The proposed algorithm has two steps. In the first step, the algorithm uses the click-graph in order to clusters all the queries of the query log. In particular, two queries are represented by the vector of neighbor documents in the click graph, and then the queries are clustered based on the Euclidean distance of their representation vectors. A simple greedy clustering algorithm is proposed that can scale to very large query-log data. In the second step, user sessions are processed and each query is represented by the cluster center that was assigned to during the first clustering step. The intuition of representing queries by their cluster center is to address the problem that two queries might have the same search intent. Thus, the authors in [17] prefer to work with "query concepts" rather than individual queries. Then *frequent sequential patterns* are mined from the user sessions. For each frequent sequence of query concepts $c_s = c_1 \ldots c_l$, the concept c_l is used as a candidate concept for the sequence $c'_s = c_1 \ldots c_{l-1}$. A ranked list of candidate concepts c for c'_s is then built based on the occurrences of the concepts c following c'_s in the same session; the more occurrences c has, the higher c is ranked. In practice, it is only needed to keep the representative queries of the top-k (e.g., $k = 5$) candidate concepts. These representative queries are called the *candidate recommendations* for the sequence c'_s and can be used for query recommendation, when c'_s is observed online.

5. Conclusions

In this chapter we reviewed elements of mining graphs in the context of web applications. We focused on graphs arising in social networks, social media, and query logs. We discussed modeling issues and we presented specific problems in those areas, such as estimating the reputation and the popularity of items in a network, mining query logs, and performing query recommendations. Understanding the structure of graphs appearing in those applications, modeling the complex interactions between entities, and designing algorithms for leveraging the latent knowledge introduces new challenges in the field of graph mining. Classic graph-mining algorithms such as those involving random walks can provide a starting point. However, they often need to be extended and adapted in order to capture the requirements and complexities of the data models and the applications at hand.

References

[1] Lada A. Adamic, Jun Zhang, Eytan Bakshy, and Mark S. Ackerman. Knowledge sharing and yahoo answers: everyone knows something. In *Proceedings of the 17th international conference on World Wide Web (WWW)*, 2008.

[2] Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. Finding high quality content in social media, with an application to community-based question answering. In *Proceedings of ACM WSDM*, pages 183–194, Stanford, CA, USA, February 2008. ACM Press.

[3] Ricardo Baeza-Yates. Graphs from search engine queries. In *Theory and Practice of Computer Science (SOFSEM)*, 2007.

[4] Ricardo Baeza-Yates and Alessandro Tiberi. Extracting semantic relations from query logs. In *Proceedings of the 13th ACM international conference on Knowledge discovery and data mining (KDD)*, 2007.

[5] Ricardo A. Baeza-Yates, Carlos A. Hurtado, and Marcelo Mendoza. Query recommendation using query logs in search engines. In *Current Trends in Database Technology – EDBT Workshops*, 2004.

[6] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[7] Albert-Laszlo Barabasi. *Linked: How Everything Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life*. Plume Books, April 2002.

[8] L. Becchetti, C. Castillo, D. Donato, R. Baeza-Yates, and S. Leonardi. Link analysis for web spam detection. *ACM Transactions on the Web (TWEB)*, 2(1):1–42, February 2008.

[9] Doug Beeferman and Adam Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the 6th ACM international conference on Knowledge discovery and data mining (KDD)*, 2000.

[10] Nicholas J. Belkin. The human element: helping people find what they don't know. *Communications of the ACM*, 43(8), 2000.

[11] Jiang Bian, Yandong Liu, Ding Zhou, Eugene Agichtein, and Hongyuan Zha. Learning to recognize reliable users and content in social media with coupled mutual reinforcement. In *Proceedings of the 18th international conference on World Wide Web (WWW)*, 2009.

[12] P. Boldi, R. Posenato, M. Santini, and S. Vigna. Traps and pitfalls of topic-biased pagerank. In *Proceedings of the 4th International Workshop on Algorithms and Models for the Web-Graph (WAW)*, 2008.

[13] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, and Sebastiano Vigna. The query-flow graph: model and applications. In *Proceeding of the 17th ACM conference on Information and knowledge management (CIKM)*, 2008.

[14] Francesco Bonchi, Carlos Castillo, Debora Donato, and Aristides Gionis. Topical query decomposition. In *Proceedings of the 14th ACM international conference on Knowledge discovery and data mining (KDD)*, 2008.

[15] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engines. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[16] Guido Caldarelli. *Scale-Free Networks*. Oxford University Press, 2007.

[17] 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 *Proceeding of the 14th ACM international conference on Knowledge discovery and data mining (KDD)*, 2008.

[18] Carlos Castillo, Debora Donato, and Aristides Gionis. Estimating the number of citations of a paper using author reputation. In *String Processing and Information Retrieval Symposium (SPIRE)*, 2007.

[19] L. Catledge and J. Pitkow. Characterizing browsing behaviors on the world wide web. *Computer Networks and ISDN Systems*, 6, 1995.

[20] Hyunwoo Chun, Haewoon Kwak, Young H. Eom, Yong Y. Ahn, Sue Moon, and Hawoong Jeong. Comparison of online social relations in volume vs interaction: a case study of cyworld. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement (IMC)*, 2008.

[21] CiteSeer, http://citeseer.com.

[22] Nick Craswell, Rosie Jones, Georges Dupret, and Evelyne Viegas, editors. *Workshop on Web Search Click Data (WSCD), held in conjunction with WSDM*, Barcelona, Spain, 2009.

[23] Nick Craswell and Martin Szummer. Random walks on the click graph. In *Proceedings of the 30th annual international ACM conference on Research and development in information retrieval (SIGIR)*, 2007.

[24] G. M. Del Corso, A. Gulli, and F. Romani. Fast pagerank computation via a sparse linear system. *Internet Mathematics*, 2(3), 2005.

[25] Alex Fabrikant, Elias Koutsoupias, and Christos Papadimitriou. Heuristically optimized trade-offs: A new paradigm for power laws in the internet. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP)*, 2002.

[26] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *Proceedings of the annual ACM conference on Data Communication (SIGCOMM)*, 1999.

[27] D. Fogaras, B. Rácz, K. Csalogány, and T. Sarlós. Towards scaling fully personalized pageRank: algorithms, lower bounds, and experiments. *Internet Mathematics*, 2(3):333–358, 2005.

[28] Bruno M. Fonseca, Paulo Braz Golgher, Edleno Silva de Moura, Bruno Pôssas, and Nivio Ziviani. Discovering search engine related queries using association rules. *Journal of Web Engineering*, 2(4), 2004.

[29] Ko Fujimura and Naoto Tanimoto. The eigenrumor algorithm for calculating contributions in cyberspace communities. *Trusting Agents for Trusting Electronic Societies*, pages 59–74, 2005.

[30] Robert Gunning. *The technique of clear writing*. McGraw-Hill, 1952.

[31] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating Web spam with TrustRank. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, pages 576–587, Toronto, Canada, August 2004. Morgan Kaufmann.

[32] T.H. Haveliwala. Topic-sensitive pagerank. In *Proceedings of the eleventh International World Wide Web Conference (WWW)*, Honolulu, Hawaii, 2002.

[33] Rosie Jones and Kristina L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *Proceedings of the 16th ACM conference on Conference on information and knowledge management (CIKM)*, 2008.

[34] Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. I know what you did last summer: query logs and user privacy. In *Proceeding of the 16th ACM conference on Information and knowledge management (CIKM)*, 2007.

[35] Rosie Jones, Ravi Kumar, Bo Pang, and Andrew Tomkins. Vanity fair: privacy in querylog bundles. In *Proceeding of the 17th ACM conference on Information and knowledge management (CIKM)*, 2008.

[36] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Exploiting the block structure of the web for computing pagerank. Technical report, Stanford University, 2003.

[37] J. Peter Kincaid, Robert P. Fishburn, Richard L. Rogers, and Brad S. Chissom. Derivation of new readability formulas for navy enlisted personnel. Technical Report Research Branch Report 8-75, Millington, Tenn, Naval Air Station, 1975.

[38] Jon Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of ACM*, 46(5), 1999.

[39] Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alexandros Ntoulas. Releasing search queries and clicks privately. In *Proceedings of the 18th international conference on World Wide Web (WWW)*, 2009.

[40] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS)*, 2000.

[41] Ravi Kumar, Jasmine Novak, Bo Pang, and Andrew Tomkins. On anonymizing query logs via token-based hashing. In *Proceedings of the 16th international conference on World Wide Web (WWW)*, 2007.

[42] A.N. Langville and C.D. Meyer. Updating pagerank with iterative aggregation. In *Proceedings of the 13th International World Wide Web Conference on Alternate track papers & posters (WWW)*, New York, NY, USA, 2004.

[43] G. Harry McLaughlin. SMOG grading: A new readability formula. *Journal of Reading*, 12(8):639–646, 1969.

[44] Qiaozhu Mei, Dengyong Zhou, and Kenneth Church. Query suggestion using hitting time. In *Proceeding of the 17th ACM conference on Information and knowledge management (CIKM)*, 2008.

[45] Michael Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, 1(2), 2003.

[46] M. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 2005.

[47] M. E. J. Newman and Juyong Park. Why social networks are different from other types of networks. *Physical Review E*, 68(3):036122, Sep 2003.

[48] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: bringing order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[49] Romualdo Pastor-Satorras, Alexei Vazquez, and Alessandro Vespignani. Dynamical and correlation properties of the internet. *Physical Review Letters*, 87(25):258701, Nov 2001.

[50] Benjamin Piwowarski and Hugo Zaragoza. Predictive user click models based on click-through history. In *Proceedings of the 16th ACM conference on Conference on information and knowledge management (CIKM)*, 2007.

[51] Barbara Poblete and Ricardo Baeza-Yates. A content and structure website mining model. In *Proceedings of the 15th international conference on World Wide Web (WWW)*, 2006.

[52] Barbara Poblete, Carlos Castillo, and Aristides Gionis. Dr. searcher and mr. browser: a unified hyperlink-click graph. In *Proceeding of the 17th*

ACM conference on Information and knowledge management (CIKM), 2008.

[53] Filip Radlinski and Thorsten Joachims. Query chains: learning to rank from implicit feedback. In *Proceeding of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005.

[54] Atish Das Sarma, Sreenivas Gollapudi, and Rina Panigrahy. Estimating pagerank on graph streams. In *Proceedings of the 27th ACM Symposium on Principles of Database Systems (PODS)*, 2008.

[55] Stephan H. Strogatz. Exploring complex networks. *Nature*, 410(6825):268–276, March 2001.

[56] Qi Su, Dmitry Pavlov, Jyh-Herng Chow, and Wendell C. Baker. Internet-scale collection of human-reviewed data. In *Proceedings of the 16th international conference on World Wide Web (WWW)*, pages 231–240, New York, NY, USA, 2007. ACM Press.

[57] Jaime Teevan, Eytan Adar, Rosie Jones, and Michael A. S. Potts. Information re-retrieval: repeat queries in yahoo's logs. In *Proceedings of the 30th annual international ACM conference on Research and development in information retrieval (SIGIR)*, 2007.

[58] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[59] Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. Clustering user queries of a search engine. In *Proceedings of the 10th international conference on World Wide Web (WWW)*, 2001.

[60] Gui-Rong Xue, Hua-Jun Zeng, Zheng Chen, Yong Yu, Wei-Ying Ma, WenSi Xi, and WeiGuo Fan. Optimizing web search using web click-through data. In *Proceedings of the 13th ACM international conference on Information and knowledge management (CIKM)*, 2004.

[61] Yahoo! Answers, http://answers.yahoo.com.

[62] Soon-Hyung Yook, Filippo Radicchi, and Hildegard Meyer-Ortmanns. Self-similar scale-free networks and disassortativity, Jul 2005.

[63] Jun Zhang, Mark S. Ackerman, and Lada Adamic. Expertise networks in online communities: structure and algorithms. In *Proceedings of the 16th international conference on World Wide Web (WWW)*, 2007.