4 Direction and Weights

xkcd webcomic by Randall Munroe (`xkcd.com/262/`). (XXX: Need to email to make sure we have permission per Creative Commons Attribution-NonCommercial 2.5 License.)

Many real-world networks are directed and weighted. Familiar examples include Wikipedia and the Web at large, where hyperlinks are weighted by click traffic; every app where we rate products and services, from books to movies and from drivers to songs; social networks stemming from Twitter, where friend/follower links are weighted by retweets, quotes, replies, and mentions; and even Facebook, where friend interactions are weighted by "likes" and reshares. Many of these networks are built on top of Internet and Web technologies. In this chapter you will become familiar with some of these networks and protocols.

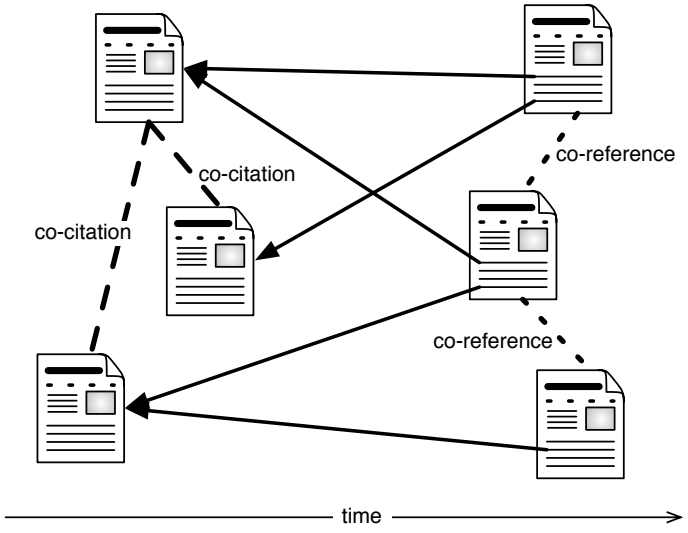


Fig. 4.1 A citation network. The citation links are shown as solid arrows. Undirected *co-citation* and *co-reference* links induced by the citation network are also shown as dashed lines. Note that in a citation network, links should always point backward in time: we cannot cite a paper that has not yet been published.

4.1 Directed Networks

In the networks that we have discussed until this point, the direction of the edges is not important. In social network, we often assume — even if it is not always true — that friendship is symmetric: if Alice is a friend of Bob's, then Bob is equally a friend of Alice's. In the Internet, packets travel in both directions between two routers, or two autonomous systems. Cars travel both ways on most roads, and for every flight from New York to Rome there is a flight from Rome to New York. In many networks these symmetries do not hold; a link has a particular direction, and may not be reciprocated. For instance, if Charlie follows Donna on Twitter, Donna may not follow Charlie back. As mentioned in Chapter 1, a *directed link* has a *source* node and a *target* node. The directed link is typically represented as an arrow pointing from source to target. We call e network with directed links a *directed network*. While in an undirected network each node has a degree, in a directed network each node has an *in-degree* (number of incoming links) and an *out-degree* (number of outgoing links).

The type of networks in which we most often observe directed links are communication and information networks. Familiar examples include email (see Figure 1.4) and Wikipedia (see Figure 1.5). Another notable example comes from science. Scientists always build upon what was done before. When they publish their findings,

they refer to related work in previous publications. The resulting *citation network* is a prototypical example of an information network. Nodes represent papers. A link from a paper to another, called *citation*, indicates some kind of relationship between the content of the to papers: a shared methodology, an alternative approach to solve a problem, a previous finding that was confirmed, improved, or possibly even contradicted. Figure 4.1 provides an illustration of citation networks.

4.2 The Web

We are all familiar with the Web, where a hyperlink leads from page **A** to page **B**, while page **B** may not have a link to page **A**. Interestingly, the idea of bi-directional links had existed for decades before the *World-Wide Web* was invented.

4.2.1 A Brief History of the Web

Bi-directional were hard to realize technically, because they required some type of centralized library to store the link information — search engines did not yet exist. In the early 1990s, Tim Berners-Lee introduced a simpler, directed hyperlink model that was easy to realize because anyone could link from a page to another without worrying about the target page; even if the target page did not exist, the user would just experience a broken link. Lots of people started writing Web pages using Berners-Lee's hypertext language for Web content, and hosting websites using his communication protocol for Web browsers and servers. The Web was born.

The link was key to the success of the Web. Each page would have a Web address called *URL* (Uniform Resource Locator) to make it easy to link from one page to another. For about 10 years, the Web grew as many organizations created websites to present information and sell products and services. But the Web was still mainly a network in which the producers of information were distinct from its consumers. Creating a website required skills that most people did not have. The introduction of online journals called "Web logs," or *blogs*, changed that. Blogs made it easier for people to create simple sites according to templates, and produce content in the form of journal (blog) entries, hosted by third-party providers. Each blog entry would have a URL to make it easy to link from one blog entry to another. Bloggers would link to each other's blogs. Blogs quickly became the fastest-growing segment of the Web, even competing with traditional media outlets. Most importantly, many people who had been mainly consumers of information became producers as well. This was one important aspect of the so-called *web 2.0* revolution.

Just as it became easier for people to share information through blogs, it also became easy to share photos, movies, and all sorts of other media through sites like Flickr and YouTube. People could also share links, by publishing their bookmarks on tagging sites and later on social networking sites. The link became ubiquitous and familiar. The natural next step was for people to have links, and this happened with

online social networks like Friendster, Orkut, MySpace, LinkedIn, and Facebook. To lower even mode the cost of creating nodes and links, the concept of *microblog* emerged, letting people post very short messages to be broadcast to their friends. This mix of social networks and blogs, introduced by Twitter and soon copied by Facebook, has been so popular that a significant portion of the Earth's population is now part of the Web.

From a network perspective, the concept of node has extended to represent anything with a URL, from pages to people, from sites to thoughts, from photos to songs, from movies to articles, and so on. Likewise, the concept of link has extended because any object can point to any other: Tweets link to blog entries, Wikipedia articles cite other articles and external pages, people connect with friends and favorite things, maps link to photos (and vice versa), and so on. The Web has thus grown to encompass almost every aspect of our lives.

4.2.2 How the Web Works

Let us go back to the basics of the Web. To better understand how this ubiquitous network works, and how we can gather data about it, we need to get an idea of its language and protocol. Pages are written in some version of *HTML* (HyperText Markup Language) with interactivity provided by scripting languages, such as Javascript, that can be interpreted by the browser. Details about these languages are outside the scope of this book, except for the important concept of hyperlinks between pages, that we have already discussed. HTML provides an easy way to encode a link to another page with a special *anchor tag* (`<a>`). For example, the simple code `<a href="http://npr.org/">news</a>` creates a link for the *anchor text* "news," which, when clicked by the user, will cause the browser to fetch the page at `npr.org`.

How the fetching or downloading of pages between a *client* (the browser) and a Web server works is specified by *HTTP*, the HyperText Transfer Protocol. Similar to other Internet client-server protocols, HTTP is actually pretty simple: it prescribes how the client requests for a page, and how the server responds. Figure 4.2 illustrates the protocol through an example. To first establish a connection with the server, the client must know the IP (Internet Protocol) address of the server. The URL typically specifies a server *hostname* and a *file path*. For example, in the URL `http://npr.org/` the server hostname is `npr.org` and the file path is `/`. Since in this case the path is a directory, the server will look for a default file name such as `index.html` in that directory. To get the IP address, the browser uses a service called DNS (Domain Name Service). This is a different protocol and we won't go into the details, but basically the browser asks a known DNS server to lookup some hostname (say `npr.org` and the DNS server returns the corresponding IP address (say `216.35.221.76`). IP address in hand, the browser connects to the server using another protocol, called TCP (Transmission Control Protocol), whose details do not concern us.

Once the connection is established, the `http://` part of the URL means that

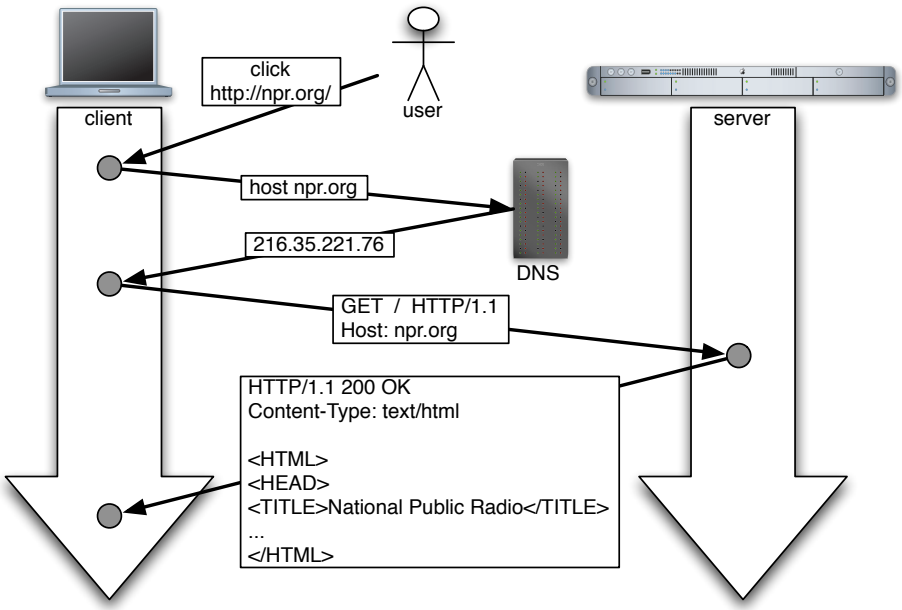


Fig. 4.2 An example of how client (browser) and server communicate through the HTTP protocol to follow a link and visit a Web page. The vertical arrows represent the direction of time.

the browser talks to the server using HTTP. To do this, the browser sends an HTTP request to the server and waits for an HTTP response. Request and response messages have a format consisting of a *header*, followed by an empty line, followed by an optional *body*. The request header can consist of just a couple of lines. The most common type of request is `GET`, which simply asks for a page (the path). In this case the request has no body, so as soon as the server receives the empty line, it responds. In other cases, such a `POST` request, the body contains additional content parameters. This can be used to send input from a form. In addition to the request type and path, the header typically contains one other line to specify the hostname. This may seem strange, as one would think that the server knows its own name! However, a single server may often host many websites. This is called *virtual hosting*. The hostname tells the server which of its virtual hosts is being requested. The response header can have several lines of information, such as the type of server, date, number of bytes returned, etc. The most important is the *response code*. For example, code 200 means "success" while 404 means "not found." The body of the response is the actual content of the resource requested, typically the HTML code of the page.

4.2.3 Web Crawlers

Any program that uses the HTTP protocol to request content from Web servers is a Web client. The browser is the familiar tool we use to navigate the Web, allowing us to virtually move from one node to the next in this huge network of sites and pages. *Web crawlers* are programs that automatically download Web pages. Since information in the Web is scattered across billions of pages served by millions of servers around the globe, crawlers are designed to collect information that can be analyzed and mined in a central location. The primary application of crawlers are search engines. The Web is a dynamic entity evolving at rapid rates, therefore search engines use crawlers to stay fresh and provide current information as pages and links are added, deleted, moved, and updated. A search engine takes the information collected by a crawler and creates a data structure (an *index*) that maps content (keywords and phrases) to the pages that contain it. This way, when a user submits a query, the search engine can quickly retrieve the pages that contain those keywords. Another task of the search engine is to determine how to rank results, so that users can find quality results among millions of hits. One of the key methods to do this exploits the network structure of the Web and is presented in Section 4.3.

Other uses of Web crawlers include business intelligence, whereby organizations monitor competitors and potential collaborators; digital libraries and bibliometrics systems to make scholarly work more accessible and assess its impact; webometrics tools, to evaluate influence of institutions from their online presence; and even malicious applications, for example the harvesting of email addresses by spammers, or the collection of personal information for phishing and identity theft. Crawlers are also employed for research purposes, such as to reconstruct the structure of the Web link graph. Because crawlers are so valuable to the study of information networks, let us get an idea of how they work.

Crawlers are very complex software systems; Google founders Sergey Brin and Lawrence Page identified the Web crawler as the most sophisticated yet fragile component of a search engine. But the basic concept of a crawler is not difficult. In its simplest form, a crawler is just a breadth-first search algorithm (Section 2.3) running on the Web link graph. It starts from a set of *seed* pages (URLs) and then recursively extracts the links within them to fetch more pages, and so on. This simple description conceals technical challenges that we won't go into, such as network connections bottlenecks, page revisit scheduling, spider traps (when meaningless URLs are generated automatically by servers), canonical URLs (to decide whether two links point to the same page), robust parsing (interpreting the often-incorrect HTML syntax of pages), and the ethics of dealing with remote Web servers.

Figure 4.3 shows the flow of a basic crawler. The crawler maintains a queue of unvisited URLs called the *frontier*. The list is initialized with seed URLs, which are typically a set of high-quality pages, for example from a previous crawl. In each iteration of its main loop, the crawler picks the next URL from the frontier, fetches

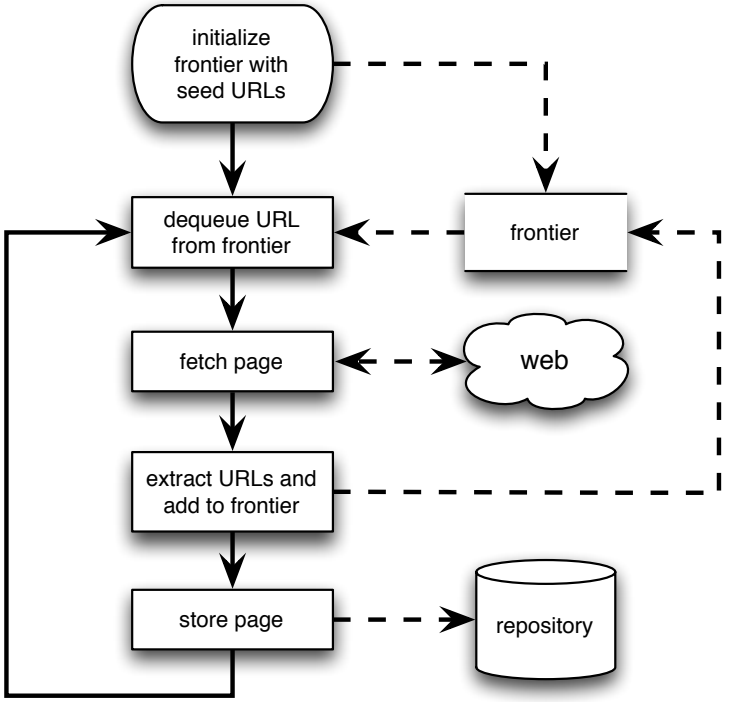


Fig. 4.3 Flow chart of a basic crawler. The main data operations are shown with dashed arrows.

the page corresponding to the URL through HTTP, parses the retrieved page to extract its URLs, adds newly discovered URLs to the frontier, and finally stores the page (and other extracted information, including index terms and network structure) in a disk repository. The crawler would stop when the frontier becomes empty, but this rarely happens in practice due to the high average out-degree of pages (on the order of ten or more links per page across the Web). The crawling process may be terminated when a certain number of pages have been crawled, or — as in the case of search engines — go on forever.

A typical crawler's frontier quickly becomes huge, containing many millions of unvisited links. Crawlers often employ heuristics in an attempt to prioritize links that are likely to lead to quality content. Because the frontier is commonly implemented as a first-in-first-out (FIFO) queue, before we visit any page at distance n from a seed, we visit all pages at distance $n-1$ or less. This is a good strategy because, as we shall see below, the farther away we go from a quality page, the lower the chances to find quality pages. Breadth-first crawlers seeded with large numbers of good pages can thus quickly discover many new good pages or revisit known pages to see if they have been updated since the last visit. Efficient crawlers employed by search engines use many tricks to optimize the crawling process, and

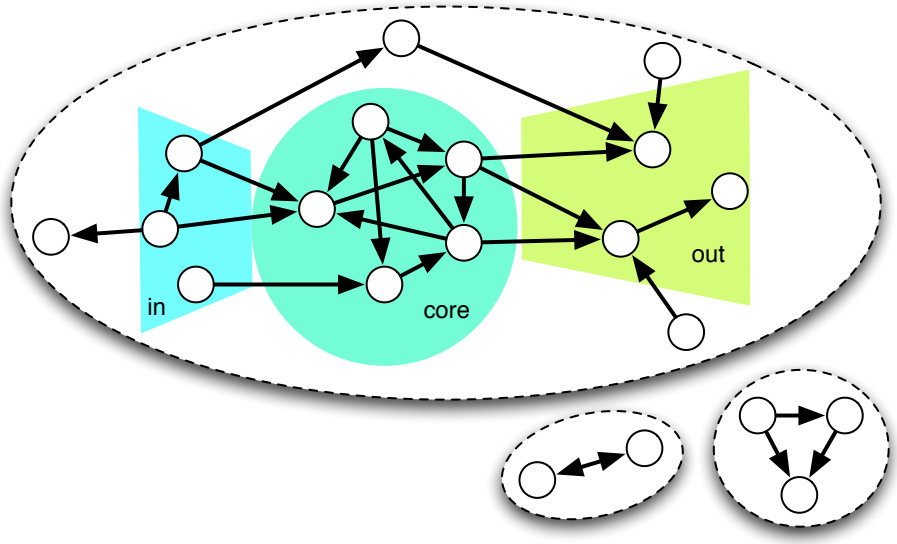


Fig. 4.4 The bow-tie structure of the Web graph. The giant component has a giant strongly connected component (sometimes referred to as "core"), an in-component ("in"), and an out-component ("out").

devote to this task clusters with thousands of machines working in parallel around the clock. This way they crawl and index millions of pages per day, and maintain search results fresh.

4.2.4 Web Structure

Large-scale crawls have revealed several interesting facts about the structure of the Web graph. There are various (weakly) connected components. Their sizes tend to have a skewed distribution, with the largest one dominating (with more than 90% of all pages) and a great many small ones. Within the giant component, we find the largest strongly connected component. Recall from Section 2.4 that the *in-component* and *out-component* of the giant strongly connected component are the sets of pages from which the component can be reached and that can be reached from the component, respectively. This is sometimes called *bow-tie* structure when referring to the Web (see Figure 4.4). The relative sizes of the giant strongly connected component, in-component, and out-component vary depending on the strategies used by Web crawlers to collect network data.

Several research teams have studied the degree distribution of the Web graph based on large crawls. The average in-degree (number of links to a page) is around 10–30 links, but the standard deviation is at least an order of magnitude larger, so that the heterogeneity parameter κ is large. Therefore the average is not a very

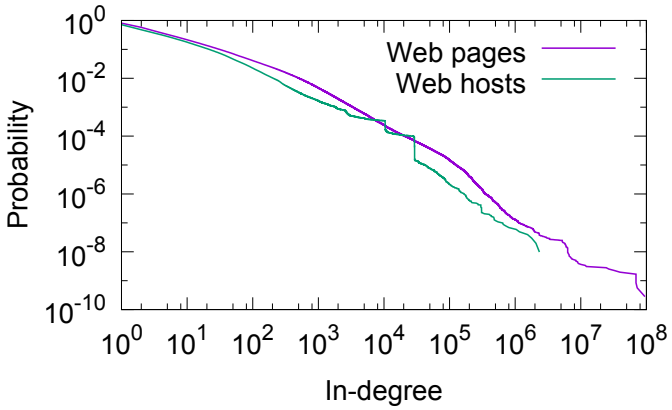


Fig. 4.5 Cumulative in-degree distribution of the Web graph. This network, based on a large crawl from August 2012, has $N = 3.6$ billion pages and $L = 129$ billion links. The distribution of in-degree is also shown for the *host graph*, in which nodes represent entire websites rather than individual pages, and a link indicates that there is at least one hyperlink between pages in two websites. In both cases we observe hub nodes with huge numbers of incoming links. (Network from `webdatacommons.org` based on crawl data from `commoncrawl.org`.)

meaningful measure. In fact, the distribution if in-degree has a heavy tail spanning several orders of magnitude, as shown in Figure 4.5. This reveals the presence of huge hub pages in the Web, which capture a disproportionate amount of links and traffic. This is a robust feature of the Web, and it has not changed since the Web was only a few years old and a few hundred million pages in size.

The distribution of out-degree is more difficult to analyze. Although crawlers find pages with thousands of outgoing links, the distribution does not span as many orders of magnitude as that of in-degree. More importantly, whereas a page with many incoming links is usually a signature of popularity, one containing too many links to other pages is typically a signature of spam behavior: so-called *link farms* created to boost a site's search ranking. Consider too that for efficiency reasons, crawlers often truncate the download of very long pages, so that measures of out-degree are unreliable.

Researchers have also used crawl data to study the average path length of the Web graph. Figure 4.6 plots the average path length as a function of network size for three strongly connected subnetworks obtained by crawling portions of the Web. We can clearly see that the average path length grows very slowly with the number of nodes: as the network size grows by several orders of magnitude, shortest paths get only a few steps longer on average. This *ultra-small world* structure is due to the presence of hub pages, as we discussed in Chapter 3.

These same features of the Web — heavy-tail distributions of component size

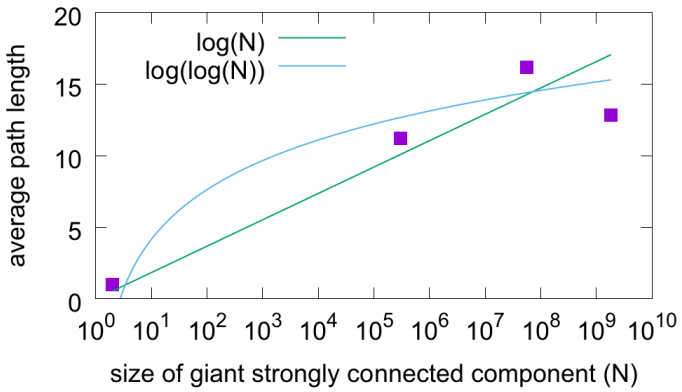


Fig. 4.6 Average path length versus size of Web graphs. The three to the right points correspond to measurements based on directed networks obtained by crawling the Web in 1999, 2000, and 2012. In each case we consider the subnetwork corresponding to the largest strongly connected component, whose size N is plotted on a logarithmic scale. The curves shows a slow-growing logarithmic function (small world) and an even slower-growing log-log function (ultra-small world).

and in-degree, ultra-small world signature — have also been observed in other information networks, notably blogs and the Wikipedia.

4.2.5 Topical Locality

In Section 2.1 we defined *homophily* as the tendency of similar nodes to be connected. The nodes of information networks, such as Web pages, Wikipedia articles, and research papers, are rich with content — text attributes that we can use to define and measure the similarity between two pages or documents. Based on the content, we can determine the topic of a page, or what it is about: two pages about sports are more similar to each other than a page about sports and one about music. Therefore we can express homophily in information network as the capability of guessing what a page is about by looking at the content of its neighbor pages. Conversely, two pages about related topics may link to each other or may have a short path connecting them. When this happens we say that the network has *topical locality*. The reason for topical locality is intuitive: when new pages, articles, or blog posts are created, the authors want to help their readers by linking to topically relevant information. As a result, links encode semantic information about the nodes.

To quantify topical locality, we can measure how likely it is that a target page within a given distance from a source page is about the same topic as the source page (Figure 4.7). We can then compare this with our expectation to run into a page about the same topic by chance, which depends on how general the topic is.

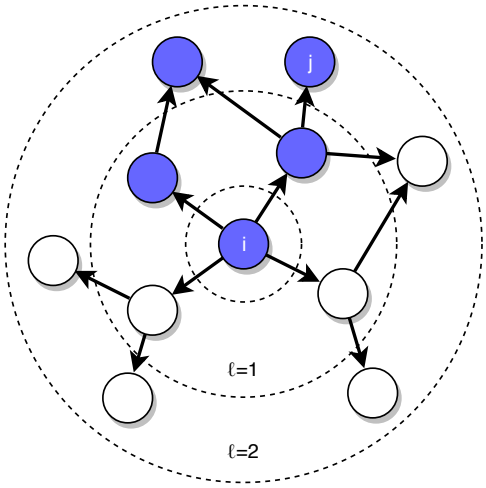


Fig. 4.7 Illustration of topical locality. Half of the pages at distance $\ell = 1$ from page i and one third of the pages at distance $\ell = 2$ are about the same topic (shown in blue).

Let us define $P(j,t)$ as the probability that page j is about topic t (blue in Figure 4.7). In the absence of any additional information, we expect a random page j to be about t with chance $P(t)$. This is the *generality* of t, or the fraction of all pages that are about t. If we know that some page i is about t and there is a path of ℓ links from i to j, our expectation becomes the conditional probability $P(t|\ell) = P(j,t|P(i,t) = 1, \ell_{i,j} = \ell)$. In Figure 4.7, $P(t|1) = 1/2$ and $P(t|2) = 1/3$. We can define the *ℓ-locality* of topic t as $\lambda_\ell(t) = \frac{P(t|\ell)}{P(t)}$. If $\lambda_\ell(t) > 1$ then the network neighborhood within distance ℓ provides information about the topic. Generalizing, we can define the *topical locality* at ℓ by averaging across a set of topics: $\lambda_\ell = \langle \lambda_\ell(t) \rangle_t$.

In practice, we can measure topical locality by performing a breadth-first crawl up to some distance from one or more seed pages about some topic. If a representative, large set of pages on a topic are available (e.g., Wikipedia articles in a given category), we can use a subset of those pages as seeds and count the fraction of the rest that are discovered by the crawl. Empirical data based on this methodology reveals that topical locality is strong on the Web, as illustrated in Figure 4.8: pages in the immediate link neighborhood of a seed page are much more likely to be topically related to the seed than random pages. Naturally, as we browse away from a page, we are less likely to encounter related pages; this phenomenon is called *topic drift*.

An alternative way to measure topical locality is to use text similarity as a proxy for topical relatedness. Text similarity is based on the co-occurrence of keywords in two pages or documents. The more keywords are shared by two pages, the stronger the evidence that the two pages are about the same topic. A commonly used text

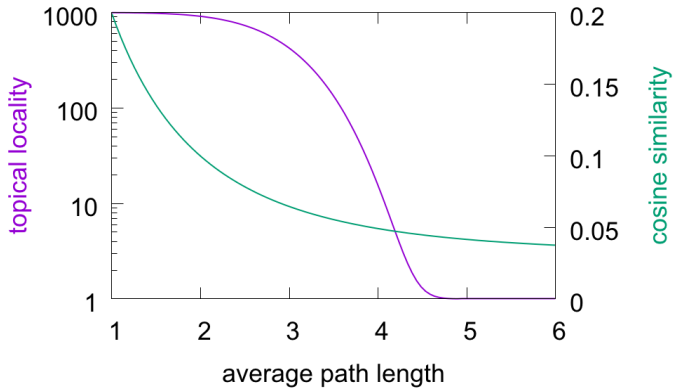


Fig. 4.8 Web topical locality, measured by breadth-first search crawls from 100 sets of seed pages for as many topics. We observe a strong topical locality: the likelihood to find a page about the same topic within one link is a thousand times higher than chance ($\lambda_{\ell=1} \approx 10^3$). As we crawl more than three links away, information about the initial topic is lost. The same topic drift is illustrated by the decay in average cosine similarity between seed and crawled pages.

similarity measure is the *cosine similarity*, described in Box 4.1. To quantify topical locality, one can measure the similarity between a crawled page and a seed page, and average across all seed pages and all pages in the crawl. Using this methodology, we can observe that neighbor pages tend to be more similar than distant pages (Figure 4.8).

Topical locality is a kind of relationship between the structure of information networks and the content of the nodes — what the network tells us about the content, and vice versa. If we start from "good" seed pages and we don't stray too far, we are likely to find other quality pages. This is one reason why search engine crawlers employ breadth-first search algorithms. Topical locality is also why it makes sense to "browse" the Web — if topical locality did not exist, would you ever click on a Web link?

There are other important hints about the content of pages that one can glean from the network structure. For example, the local network neighborhood of a page may be a signature of spam content, as when the only links to a page originate from nodes that have no incoming links themselves. Conversely, when a page has many incoming links from pages with high in-degree, that may be a clue about the quality or prestige of the page content; more on this in the next section.

Box 4.1 Cosine similarity

In information retrieval and text mining one often needs to measure the similarity between two documents, Web pages, blurbs of text, or tag clouds. Let us represent each document d as a high-dimensionality vector, with a dimension associated with each term in the vocabulary: $\vec{d} = \{w_{d,1}, \ldots, w_{d,n_t}\}$ where $w_{d,t}$ is the weight of term t in d, and n_t is the total number of terms. Deep learning techniques based on artificial neural networks lead to similar vector representations, except that each dimension corresponds to an abstract concept rather than a word or tag. There are various ways to calculate the weights. Typically they are proportional to the frequency with which a term appears in a document — a terms that occurs often is considered a good descriptor. One often removes "noise words" that are not meaningful, such as articles and conjunctions. It is also common to discount the weights of terms that occur in many documents, as they have low discriminating power. Then we can compute the similarity between two documents d_1 and d_2 by measuring the *cosine* between their vectors:

$$\cos(\vec{d_1}, \vec{d_2}) = \frac{\vec{d_1}}{||\vec{d_1}||} \cdot \frac{\vec{d_2}}{||\vec{d_2}||} = \frac{\sum_t w_{d_1,t} w_{d_2,t}}{\sqrt{\sum_t w_{d_1,t}^2}\sqrt{\sum_t w_{d_2,t}^2}}.$$

If the terms in d_1 are also present in d_2 and vice-versa, the cosine is close to one; if the two documents do not share any terms, the cosine is zero. Note that the cosine is normalized by the norm, or size of each vector:

$$||\vec{d}|| = \sqrt{\sum_t w_{d,t}^2}.$$

This way a longer document will not appear to be similar to many others just because it has many terms: its large norm would decrease the similarity.

4.3 PageRank

We have learned that the pages retrieved by a Web crawler are processed by a search engine to build an search index — a data structure that lists all of the pages containing any given word or phrase. So, when you submit a query, the search engine can quickly list all the matching pages. But there may be millions of Web pages matching a certain query, say "social network," and you have only time to look at a handful. *Ranking algorithms* are therefore an equally critical component of a search engine. If results were sorted solely according to similarity between page content and query, users would have to look at a lot of low-quality pages, and even spam.

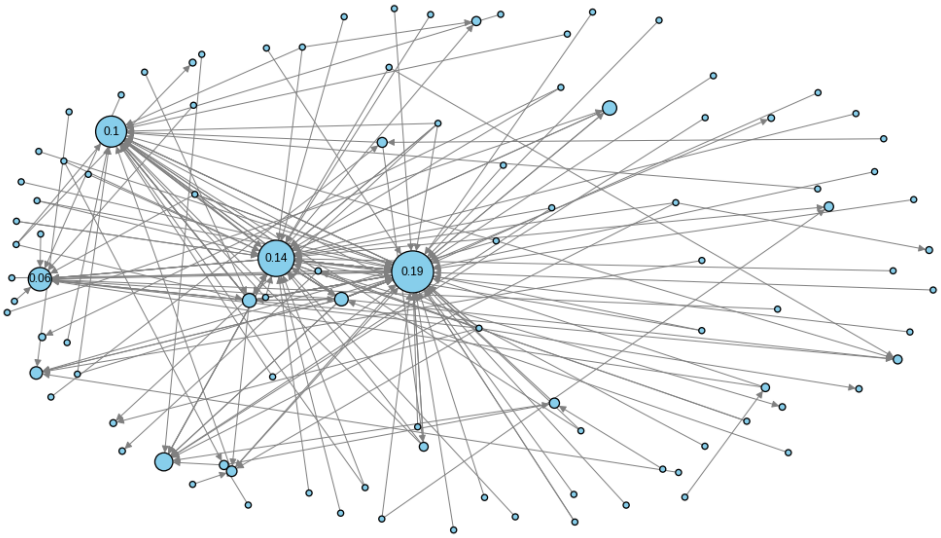


Fig. 4.9 PageRank in a directed network. The size of the nodes is proportional to their PageRank value, which is shown for a few of the nodes.

But if results are ranked also taking some page importance or prestige measure into account, then the top results are more likely to be relevant, interesting, and reliable. Search engines began employing network centrality measures as ranking criteria in 1998, when Sergey Brin and Larry Page introduced *PageRank* as a component of a new search engine they called *Google*.

PageRank is an algorithm, or procedure, to compute a centrality measure that aims to capture the prestige or importance of each node in a directed network. It is also the name we give to the centrality measure itself. So when applied to the Web, the algorithm assigns each page a PageRank value. The ranking algorithm of a search engine can then use this value, in combination with many other factors — such as the match between query and page text — to sort the results of a query. A page with high PageRank is considered prestigious or important, and is given a boost by the ranking algorithm: other things being equal, pages with larger PageRank are ranked higher. Consider as an example a spammer who copies the content of the Wikipedia article on "social network" into a blog page filled with ads. The original and plagiarized page might look very similar in terms of content. But the Wikipedia page has a much higher PageRank. So when you submit the query "social network" you will see the Wikipedia article among the top results, while the spammer's page will be buried down and you won't probably see it.

The intuition for PageRank comes from imagining people who surf the Web at random, that is, following random links from page to page. This process is known as a *random walk* (or random surf) on the Web graph. The random walk is a simple model of user browsing or searching behavior; not knowing where a desired piece of information may be, or which link will lead to it, we make the simplest assumption

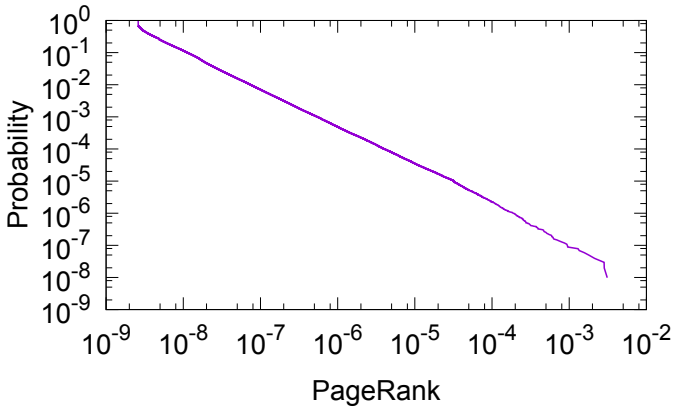


Fig. 4.10 Cumulative PageRank distribution of the Web host graph. PageRank values are normalized so that they sum up to one. (Network from the same 2012 crawl data used for Figure 4.5.)

that each link in a page has an equal chance to be clicked. We also wish to capture another user behavior, namely, that a user may at any time get tired of browsing and start a new browsing session. PageRank models this through occasional *jumps* from a current page to some other random page, irrespective of links. The random jump process is called *teleportation*.

By imagining many people carrying out these modified random walk processes (surfing plus jumping) for a long time, one can measure how often each page would be visited. The fraction of time we land on a page is what we call the *PageRank* of that page. Figure 4.9 illustrates the PageRank values in a directed network; nodes with many paths leading to them are visited more often by random surfers, and consequently have high PageRank. In practice, PageRank is computed in a more efficient way, explained in Box 4.2. Appendix B presents an interactive demonstration of PageRank.

It turns out that Web in-degree provides in many cases a good approximation of PageRank: if a page or website has many incoming links, it tends to have more paths leading to it and therefore more visibility. In fact, the distribution of PageRank is quite similar to the distribution of in-degree on the Web (Figure 4.10). Then, why not just use in-degree for ranking? To answer this question, consider that not all paths are equal. Paths from pages that are themselves visited often provide a larger boost. In other words, the importance of a page is affected by the importance of the pages that link to it — would you rather have a link to your homepage from your friend's blog, or from the front page of the *New York Times?* This is an important way in which PageRank differs from in-degree. Among two pages with the same in-degree, the one linked by pages with higher PageRank wins the game.

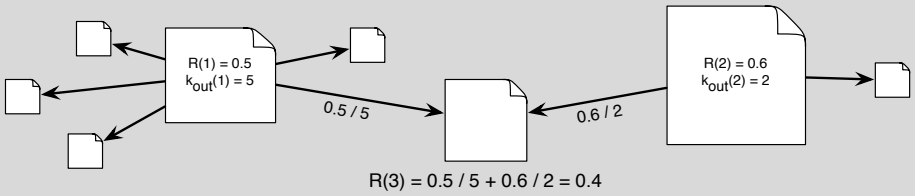
Of course, people play this game. The difference between a high and low PageRank may mean appearing on the first page of search results, or not, and that in

Box 4.2 **PageRank**

PageRank is usually computed from the hyperlink structure of a Web graph with an iterative approach called *power method*. The idea is to compute how PageRank flows from page to page. The PageRank R of each node is initialized to some value (say $R_0 = 1/N$, so that all values sum up to one). At each step, each node's value is refined, until the process converges, that is, none of the PageRank values change from one step to the next. We assume random jumps occur with probability expressed by a parameter α, called *teleportation factor*, typically set at a small value $\alpha \approx 0.15$. The probability $1-\alpha$ or *damping factor* is instead associated with the random walk process. According to the PageRank model, at every step, with probability α a user jumps to a node selected at random among all pages; with probability $1-\alpha$ the user continues browsing, following a random link from the current page. Therefore the PageRank of node i at time t is expressed by:

$$R_t(i) = \frac{\alpha}{N} + (1-\alpha) \sum_{j \in \text{pred}(i)} \frac{R_{t-1}(j)}{k_{out}(j)}. \tag{4.1}$$

The first term describes teleportation to i, which is one of N possible targets of a jump. The second term describes how one can traverse one of the links to i during the random walk: the sum is over the set of predecessors of i, i.e., the pages that link to i. Each of these pages, j, has $k_{out}(j)$ outgoing links. The PageRank of j is spread equally along these outgoing links, one of which leads to i. The figure illustrates one step in the spreading process described by the second term of the equation ($\alpha = 0$).



Note the *recursive* definition: the PageRank of a page depends on that of its neighbors. PageRank is a conserved quantity ($\sum_i R(i) = 1$), as it flows from one page to its neighbors through links. For $\alpha > 0$ it turns out that, because teleportation virtually connects all nodes, PageRank is guaranteed to converge, and it does so relatively quickly — in less than 100 steps or so, even in very large networks. Therefore the power method is a much more efficient way to compute PageRank, compared to simulating modified random walks.

turn may mean the survival or failure of a business. Many businesses thrive entirely thanks to their good search rankings. Therefore it should come as no surprise that there is an entire *search engine optimization* (SEO) industry to help websites improve their search rankings. Most SEO firms employ methods sanctioned by search engines, such as adopting descriptive page text and improving website navigability. However, less scrupulous SEO agents may employ methods of which search engines disapprove. These are often referred to as "spamdexing," or spamming the search index. One frequent spamdexing approach is that of creating *link farms*, large sets of fictitious websites that link to each other and to a target website. Such a clique structure is designed to trick PageRank-like algorithms and boost the target page's ranking. Search engines employ sophisticated network algorithms to combat link farms and other spamdexing attacks. When they detect this kind of abuse, they may remove a website from the search index.

NetworkX provides a function that runs the PageRank algorithm on a given directed network and returns a dictionary with the PageRank values of the nodes:

```
PR_dict = pagerank(D)    # D is a DiGraph
```

4.4 Weighted Networks

So far we have focused on unweighted networks, in which links have a binary nature: either two nodes are linked, or not. However, connections between real-world entities are rarely so black-an-white. Very often links have attributes that allow us to compare two connections and determine which is stronger. In Chapter 1 we have seen some examples of weighted networks: the Twitter retweet network, where two account may retweet each other any number of times; email network, where users can send any number of messages to each other; the Internet, where the data that travels on a physical link between two routers is measured by numbers of packets or mega-bits; brain networks, where synapses between neurons transmit electrical firing signals at different rates; and food webs, where we can characterize the biomass of one prey species consumed by a predator species. All of these networks are both weighted and directed.

Even when we think of a network as unweighted, that is often just a reflection of our attempt to simplify how we model and analyze the underlying real-world relationships. Take the Facebook network, for example: we typically think of a friendship link as a binary relationship. However, not all friendships are equal; two close friends may have many common contacts, may like and comment on each other's posts many times, and may reshare and tag each other's photos. The same is not true for two remote acquaintances. The unweighted Facebook friend network is just a very simplified model of actual relationships. But make no mistake: platforms like Facebook monitor your every action and are well aware of the strength of each of your links. Other social networks are similar; the movie co-star network, for

instance, can have weighted links based on the number of movies in which two stars have acted together. In Chapter 5 we will see that the strength of social ties has long played an important role in the study of social networks.

Information and transportation networks provide more examples of weighted, directed networks: in the Web and Wikipedia, some links are clicked much more often than others. And in air traffic networks, some connections carry more flights and passengers than others.

In all of these networks, we use link *weights* to represent important measures such as messages, bits, likes, clicks, and passengers. Recall from Chapter 1 that the degree, in-degree, and out-degree centrality measures are extended to the *strength*, *in-strength*, and *out-strength* in weighted networks. So when we talk about *strength centrality* we refer to a property of nodes, not links.

4.5 Information and Misinformation Spread

Let us delve a bit deeper into the features of weighted directed network, using *information diffusion networks* as a case study. In such a network, nodes represent people and links represent pieces of information — ideas, concepts, news, or behaviors that are passed from person to person. A transmissible unit of information is called a *meme* — images with captions are only one kind of internet memes. Twitter provides us with data that is ideal to observe how images, movies, links, phrases, hashtags, and other memes spread online. Each of these memes is uniquely identified by a text string: a URL for a Web link or media entity, or a label prefixed by the hash mark (`#`) for a hashtag that expresses a concept or topic. A tweet may carry multiple memes. For example the message *"Hoosiers are the best #GOIU iuhoosiers.com"* contains the hashtag `#GOIU` and the link `https://iuhoosiers.com/`.

Using data from Twitter, we can build various kinds of diffusion networks capturing different ways in which a meme can spread: via retweets, quoted tweets, mentions, and replies.[1] For example, if Alice mentions Bob in a tweet, Bob is likely to see the tweet and therefore we can assume that the tweet has spread from Alice to Bob. Similarly, if Alice replies to Bob then the message has been delivered from Bob to Alice. For simplicity, let us focus on retweets. If Alice follows Bob, then Alice may retweet a message from Bob and in so doing spread a meme contained in the tweet to all of her followers. A *retweet cascade* is a directed tree that captures how a meme propagates from its originator to all the users who are eventually exposed to it. However, as illustrated in Figure 4.11, it is not straightforward to reconstruct a cascade tree from Twitter data. Still, we can easily observe all the users who are exposed to the meme. They are all connected to the origin, forming a star net-

[1] You can explore interactive diffusion networks from Twitter using a tool from the Observatory on Social Media at `osome.iuni.iu.edu/tools/networks/`. You can also generate animation movies showing how these networks unfold over time, using another tool at `osome.iuni.iu.edu/tools/movies/`.

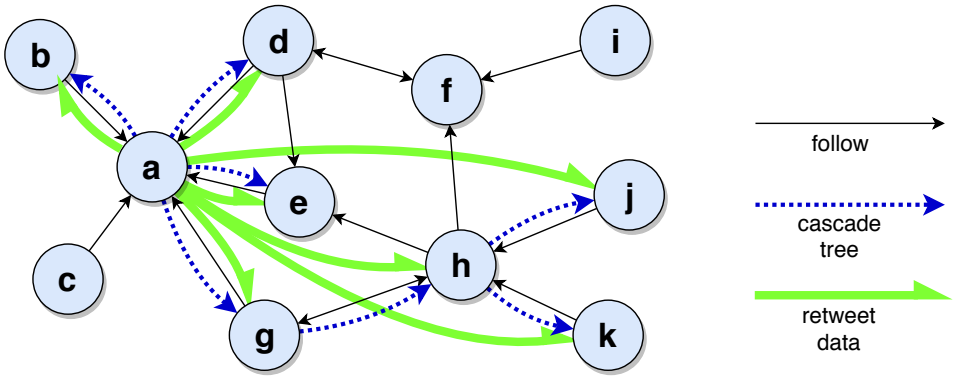


Fig. 4.11 Illustration of follower and retweet networks on Twitter. Users typically see, and sometimes retweet, the tweets of accounts they follow. In this example, **b** follows **a** and also retweets a message from **a**. **c** also follows **a**, but does not retweet **a**. A user may also retweet a message that they see because it was retweeted by someone they follow. Here **h** does not follow **a**, but retweets a message that originated from **a** and that was retweeted by **g**, who is followed by **h**. By tracing these retweet chains, one could in theory reconstruct the *retweet cascade tree* with **a** (the origin of the tweet) as root node. However, Twitter does not provide data about retweet cascades. Instead, each retweet points to the origin of the tweet. Therefore the cascade tree becomes a star network with the same root node.

work. If the underlying follower network in known, it is possible to reconstruct an approximation of the cascade tree.

The same meme can generate many cascade trees (stars). For example, multiple users can share the same link to a news article, or tweet using the same hashtag. By aggregating all of these star trees, we obtain a *forest* (set of trees) that we call the diffusion network.

In building a diffusion network, one must first formulate the meme, or memes, whose spread we want to analyze. We might be interested in a single hashtag, say `#elections` or `#soccer`; or in all links to articles from a news source or a set of news sources. For instance, when investigating online misinformation, we may wish to track stories from low-credibility sources that routinely spread fabricated news, hoaxes, conspiracy theories, hyper-partisan content, click bait, or junk science. The diffusion network in Figure 1.3 was obtained by aggregating many cascade forests of popular hashtags associated with political conversations during the 2010 US midterm election.

Once we build a diffusion network, we can observe several features. One is the community structure — is the spread uniform across the network, or concentrated within dense and segregated clusters of nodes? For example, Figure 1.3 shows a polarized structure with two mostly segregated communities: conservatives and progressives, each retweeting almost exclusively members of the same group. These communities are sometimes called *echo-chambers* because a user is mostly ex-

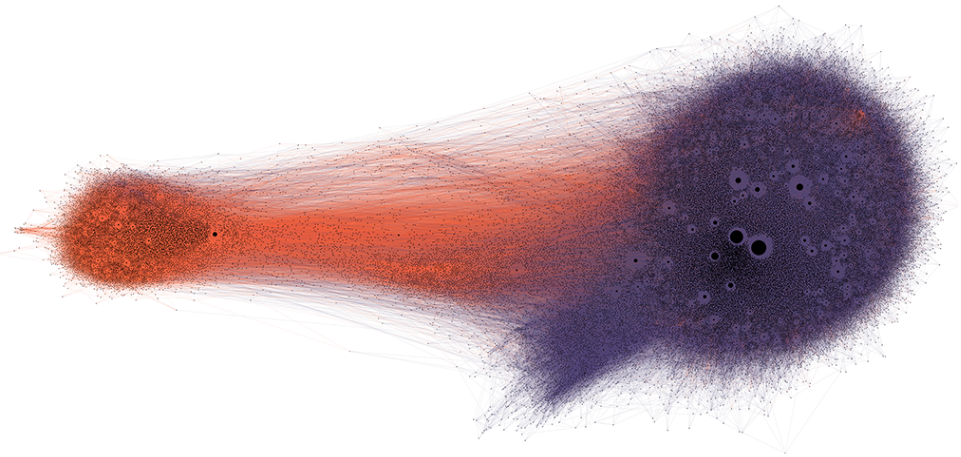


Fig. 4.12 Retweet subnetwork for articles from low-credibility (purple links) and fact-checking (orange links) sources, during the run-up to the 2016 US election. Each of the $N = 52452$ nodes represents a Twitter account with retweet links from/to at least $k = 5$ other nodes in the same subnetwork.

posed to opinions reinforcing their own. Figure 4.12 shows another example of echo-chamber, made of people who are vulnerable to political misinformation. Observe that the users who spread the misinformation share very few articles from fact-checking sources. In Chapter 6 we will learn how to detect such communities in networks.

Another property of interest is the *virality* of a meme. The simplest way to quantify it is to measure the size of the diffusion network, i.e., the number of users exposed to the meme. However, given a large network, its structure is also revealing. For instance, a star network where many followers of a celebrity retweet a meme may reflect the popularity of the celebrity more than that of the meme. On the other hand, assuming we can reconstruct cascade trees, a deep network with long chains of retweets may indicate wider appeal of the message. As illustrated in Figure 4.13, misinformation is often more viral than actual news reports.

Diffusion networks can provide us with insight about patterns of information production and consumption. In a meme's weighted and directed retweet network, the weight of a link from Alice to Bob indicates how many times Bob has propagated the meme originating from Alice's posts. We can think of Alice as a producer and Bob as a consumer of information about the meme. Extending this analysis to the entire network, we can use the out-strength and in-strength of a node to measure the user's propensity to produce and consume information, respectively — to be retweeted by others or to retweet others. A user may play both roles, of course. Therefore we can look at the ratio between out-strength and in-strength to classify a user: if the ratio is much larger than one, the user is mainly a producer; conversely a ratio below one indicates a consumer.

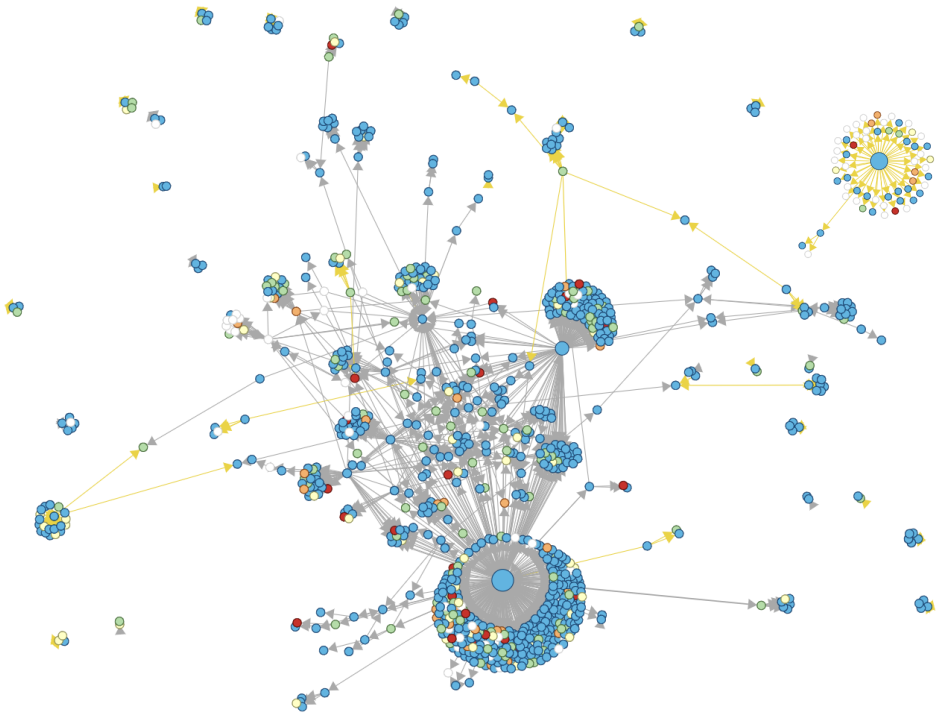


Fig. 4.13 Retweet network for two articles about the White Helmets, a volunteer rescue organization acting during the civil war in Syria. The White Helmets have been the target of a disinformation campaign, with false claims of terrorist ties and other conspiracy theories. The gray and yellow links depict the diffusion of one of these false claims and a fact-checking article, respectively. We can easily observe that the spread of the misinformation is more viral. Node size if proportional to out-strength and node color captures the probability that an account is automated: blue nodes are likely human and red nodes are likely bots. Image from Hoaxy, a tool that visualizes the spread of misinformation on Twitter (`hoaxy.iuni.iu.edu`).

Whether we focus on one meme or aggregate across all messages, a high value of out-strength can also be used as an indicator of *influence*, in the sense that messages from the user get retweeted a lot. Figs. 4.13 and 4.14 both illustrate influential nodes in diffusion networks by drawing nodes with size proportional to their out-strength. Using out-strength as a proxy for influence is in contrast with the in-degree of the follower network (i.e., the number of followers), which measures popularity but not necessarily influence; one could have many followers who do not retweet them. By comparing these two quantities — number of retweets vs. number of followers — we can get a better sense of one's influence given their popularity.

Given the huge power of social media to inform, persuade and influence all of us, it should come as no surprise that increasing resources are being devoted to

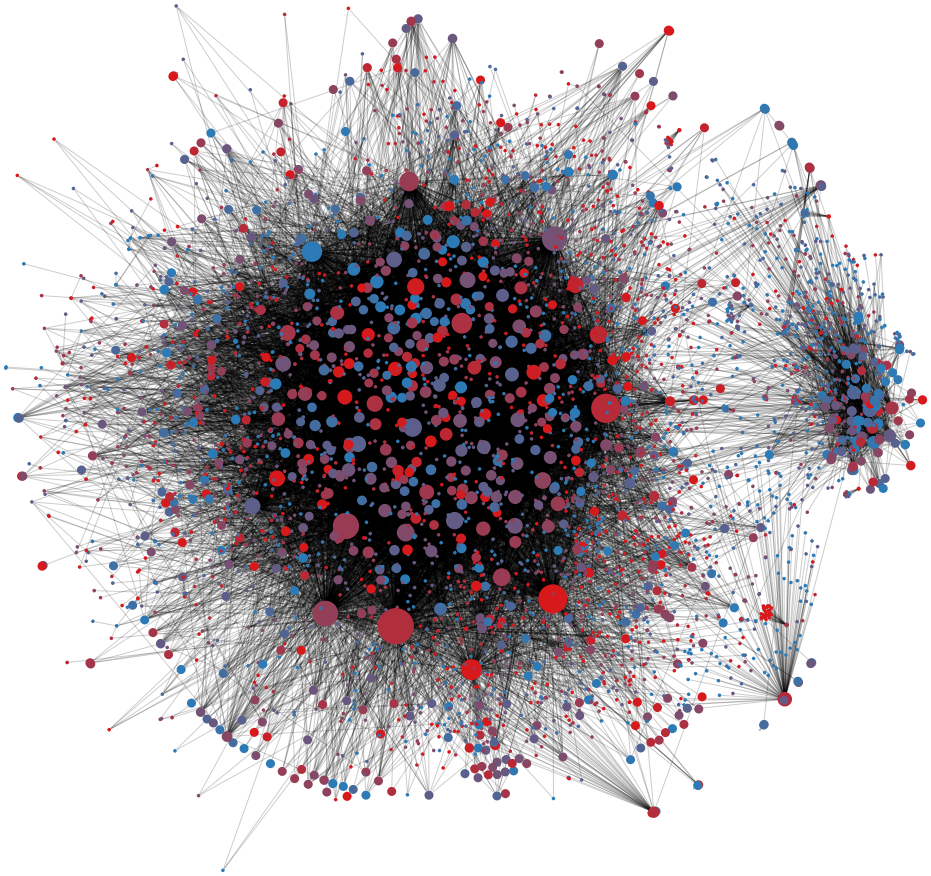


Fig. 4.14 Diffusion network for the #SB277 hashtag, about a California law on vaccinations. Node size represents account influence (out-strength) and node color represents bot scores: red nodes are likely bots, blue nodes likely humans.

manipulate these platforms. It is now easy and cheap to buy fake followers to boost an account's perceived popularity on Twitter. This amounts to adding nodes and links to increase a node's in-degree in the follower network, similarly to how one might create fake websites and links to boost a site's PageRank. Diffusion networks are also manipulated through *social bots*, deceptive fake accounts that impersonate users. Bots can be used to generate fake tweets and create the appearance of grass-roots campaigns, or *astroturf*. In this way they can trick both human users and ranking algorithms, effectively hijacking public attention. Bots can also be used to retweet some messages, amplifying their perceived engagement and boosting their spread. Figs. 4.13 and 4.14 illustrate how bots are used to amplify the spread of misinformation and manipulate public debates. In fact, we see in Figure 4.14 that bots can gain significant influence. Figure 4.15 illustrates another form of network manipulation leveraging fake replies and mentions. In this way one can target in-

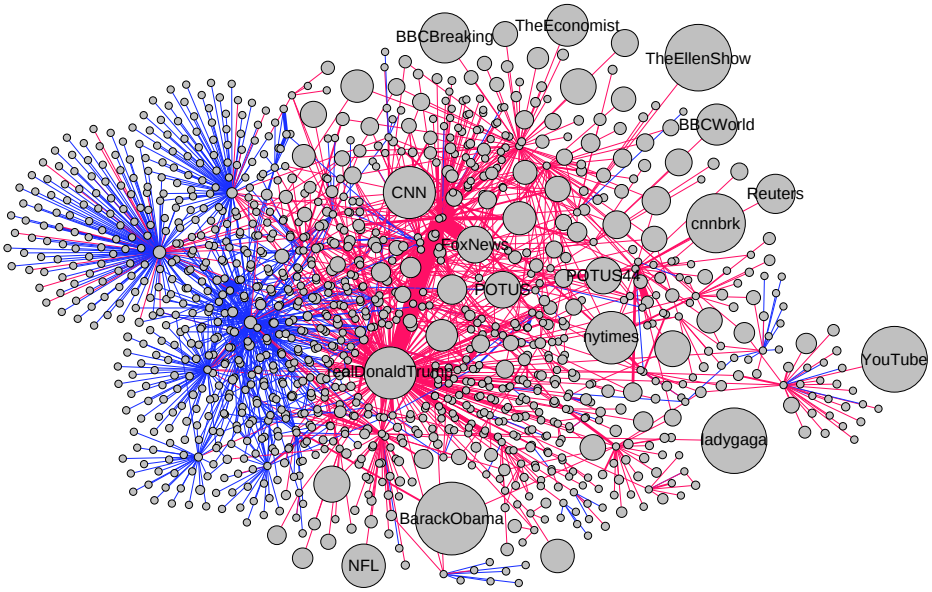


Fig. 4.15 A portion of the diffusion network for a fake news report that claimed massive voter fraud by illegal aliens in the 2016 US election. Despite presenting no evidence and being debunked by fact-checkers, the article was shared over 18 thousand times on Twitter. In this visualization, node size represents the number of followers of an account; links illustrate how the article spread by retweets or quoted tweets (blue), and by replies or mentions (red); and the width of an edge represents its weight — number of retweets, quotes, replies, and mentions between two accounts. The red link with maximum weight originated from a bot systematically tweeting the misinformation in replies to messages mentioning the US President.

fluential and popular users, such as reporters and politicians, and expose them to misinformation in the hope that they will spread it to their many followers. Yet another misinformation diffusion network manipulated by influential bots can be seen in Figure 7.1 (Chapter 7).[2] While our examples are focused on Twitter, other social media platforms like Facebook, Instagram, and WhatsApp are also exploited for spreading misinformation.

[2] You can explore the role of bots in interactive diffusion networks from Twitter using the Hoaxy tool from the Observatory on Social Media at `hoaxy.iuni.iu.edu`.

4.6 Weight Heterogeneity

In a weighted network, the weights of the links may carry important information about the process or relationships modeled by the network. Links with different weights may represent very different associations. To explore this difference, consider another class of networks that are both directed and weighted: those that capture various kinds of traffic. Traffic networks include air and other transportation networks, where weights represent passengers or flights between airports, or cars between intersections; the Internet, where weights denote packets or bits of data between routers; and Wikipedia, with weights representing clicks between articles. Let us focus on the case of Web traffic, similar to Wikipedia but extended to all websites.

4.6.1 Web Traffic

Data about Web traffic can be captured by browsers (or browser toolbars or extensions) that record click data and transmit it to a collection server. Alternatively, an Internet Service Provider may *sniff* packets that carry HTTP/HTTPS requests, which include the target host and page as well as the source URL, called *referer*. Both of these collection methods are somewhat biased; in the former case we only observe traffic generated by users of instrumented browsers, in the latter we can only see packets that pass through the ISP routers. Nevertheless, both methods enable very large collections of Web traffic data. One can count clicks between individual pages, or, as we do next, consider the aggregate traffic at the level of entire websites identified by their hostnames, such as `en.wikipedia.org`, `google.com`, and `www.indiana.edu`.

Inspecting Web traffic networks allows us to study the distribution of website traffic (total number of clicks to a site), expressed by node in-strength; and of link traffic (total number of clicks on a hyperlink), expressed by link weight. The heavy tails in Figure 4.16 reveal that both of these distributions are extremely heterogeneous. Most websites receive very few clicks, while some receive massive traffic. Similarly, many links are hardly ever clicked while others are clicked all the time.

Recall from Section 4.3 that the idea of PageRank was to simulate Web users who browse the Web. By comparing the ranking of network nodes according to their in-strength with the ranking produced by PageRank, we can ask whether PageRank is able to predict traffic from the structure of the Web link graph. In other words, does the random surfer model capture the aggregate browsing patterns of actual Web users? As it turns out, the answer is no: despite similar heavy-tailed distributions (see Figs. 4.10 and 4.16), the rank correlation between PageRank and traffic is quite weak. Therefore, some of the simplifying assumption of the PageRank model must be violated by our Web browsing behavior.

To get an idea of which ingredients of the random surfer are least realistic, let

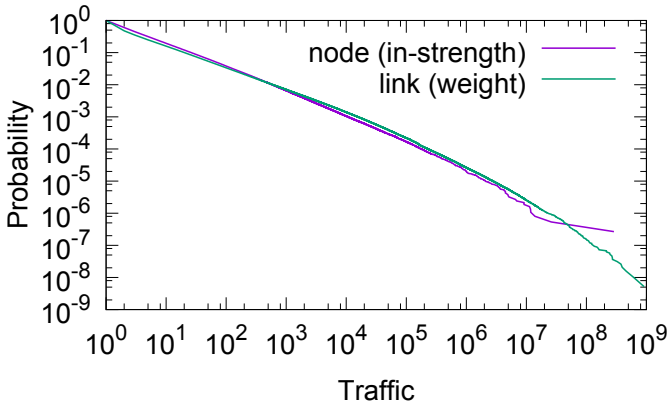


Fig. 4.16 Distributions of node in-strength (website traffic) and link weight in a Web traffic network. Almost one billion clicks, representing the combined Web browsing activities of about 100,000 anonymous users, were collected at Indiana University between 2006 and 2007. The data only includes HTTP requests made via a common browser for actual Web pages. The resulting network has about $N = 4$ million sites and $L = 11$ million weighted directed links.

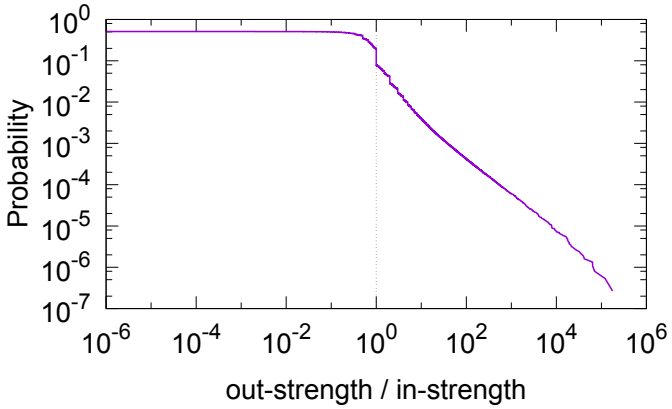


Fig. 4.17 Distributions of the ratio between node out-strength and in-strength in the Web traffic network described in Figure 4.16. Ratios $s_{out}/s_{in} \ll 1$ indicate sites where browsing sessions are more likely to terminate, while $s_{out}/s_{in} \gg 1$ for starting points.

us consider the ratio between out-strength and in-strength of nodes. Except for the start and end nodes of browsing sessions, the ratio must be one because the flow of traffic that goes into a node equals the flow that comes out of the node. According to PageRank, teleportation does not favor any nodes; each node is equally likely to be the target of a random jump (where browsing begins) and equally likely to be the source of a random jump (where browsing ends). Therefore, even with teleportation, the PageRank model produces a ratio between out-strength and in-strength very

close to one for all nodes. We would then expect a narrow distribution peaked around one. But Figure 4.17 paints a very different picture, with huge fluctuations spanning many orders of magnitude: some nodes are much more likely to be starting points and others are mush more likely to be ending points of browsing sessions. This is not surprising; we tend to start surfing from a few familiar, bookmarked sites. On the other hand, most sites are not very interesting, and therefore more likely to be places where we stop and jump away. We conclude that random teleportation is an unrealistic ingredient of the PageRank model.

4.6.2 Co-Occurrence Networks

We have already discussed many examples of weighted networks in this chapter: information, communication, transportation, biological, and even social networks often have weighted links. But there are yet other other ways in which weighted networks can arise in the presence of relationships between more than one type of entity.

The simplest case is a directed network, in which each link has a source and a target node. Imagine moving all source nodes to one side and all target nodes to the other (nodes that are both sources and targets can be duplicated and appear on both sides). To use a concrete case, consider a citation network (Figure 4.1). A citation links is a relationship connecting two different types of entities, the citing paper (source) and the cited paper (target). We can now construct a new network among papers in each of these groups. Two papers are *co-cited* if there are one or more papers that cite both of them. Their co-citation count is the number of papers citing both. Similarly, two papers are *co-referenced* if there are one or more papers cited by both of them. Their co-reference count is the number of papers cited by both. *Co-citation* and *co-reference* networks are undirected, weighted networks where the links are weighted by co-citation and co-reference counts, respectively. They are often used to find related sets of publications.

There are many situations in which relationships between two distinct types of entities are naturally represented by *bipartite networks* — where each link connects two nodes of different types. One example that we discussed in Chapter 1 is the relationship between actors/actresses and movies in which they starred. As shown in Figure 1.2, we can create from this bipartite network a network between actors/actresses who have co-starred in movies. The links in the co-star network can be weighted by the number of movies in which two people have co-starred. Generating weighted networks from bipartite networks in this way, called *projection*, is common practice, and the resulting weighted networks are called *co-occurrence networks* because links represent two entities of one type that "occur" together in association with one or more entities of another type. Other common examples of co-occurrence networks include students who take the same classes, products such as movies and books purchased by the same customers, and pages co-liked by Facebook users. Each time you "like" or share something on a social media platform, you create a link between you and the object (Figure 4.18(a)). These links

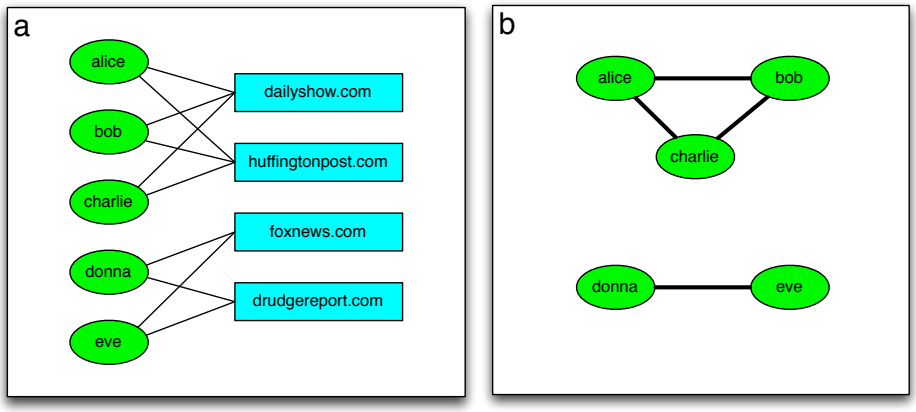


Fig. 4.18 (a) A bipartite network induced by "like" relationships. (b) A user co-occurrence network derived from projecting the "like" network onto user nodes.

are shared with friends, but also aggregated by the platform across millions of people, yielding massive co-occurrence networks (Figure 4.18(b)) that can be used for recommendation and targeted advertising.

A bipartite network can of course have weighted links. Rating systems have weights that represent, say, how much one enjoys a movie or a mobile app. Another source of weighted bipartite networks is *social tagging*: a person annotates a resource (identified by a URL) with one or more labels, or *tags*. Sharing sites like `Flickr.com` and `YouTube.com` popularized social tagging for images, movies, and other media. The elementary construct of the social tagging representation is the *triple* (u, r, t) where user u tags resource r with tag t. A resource can be a media object, scientific publication, website, news article, and so on. Tagging can be implicit in social media. For example, many Twitter users link to news articles or blog entries while also labeling their posts with hashtags. Triples can be extracted from such tweets — one for each pair of link and hashtag, with the author of the post as user. Figure 4.19(a) illustrates a set of triples. When aggregated across many users, such a set is called a *folksonomy*, because it is a taxonomy that emerges from many people. Folksonomies can be useful for searching or suggesting websites.

From a folksonomy, we can extract a bipartite network network by projecting the triples onto two of the node types. The resulting links connect only nodes of one type (say, tags) to nodes of another type (say, resources). We can therefore think of these links as directed, as shown in Figure 4.19(b). The links can also have weights representing, say, how many users have annotated a particular resource with a particular tag (Figure 4.19(b)). This way, instead of losing the information about the users, we have encoded it into a measure of links reliability.

From a bipartite network, as discussed earlier, we can create co-occurrence networks by further projecting one type of nodes based on shared neighbors of the other type. For example, in Figure 4.19(c) we project onto tags to obtain a resource net-

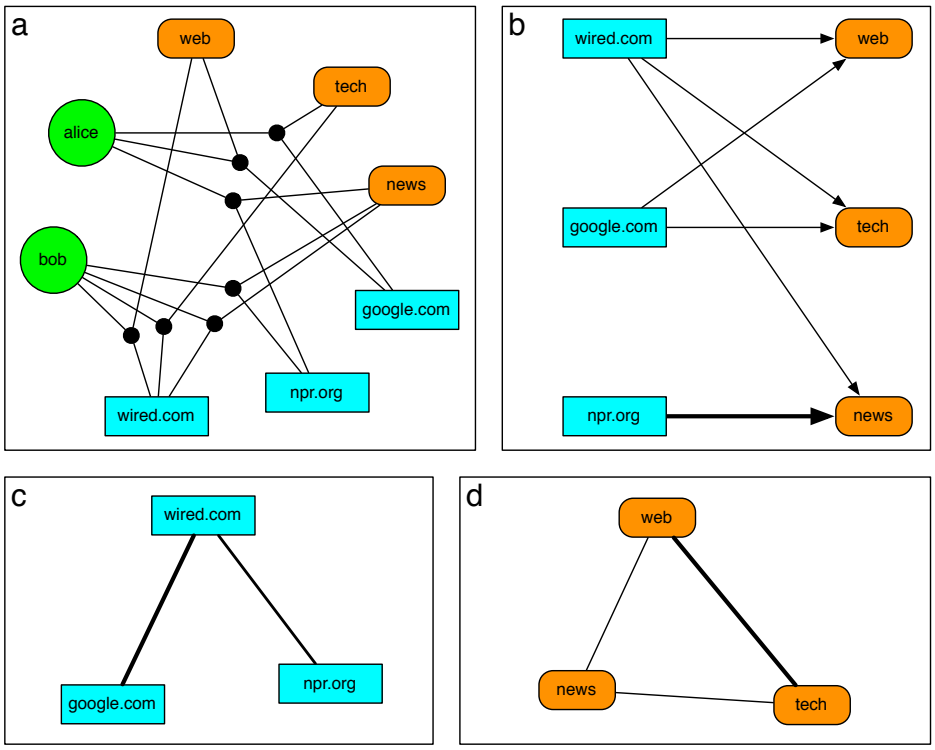


Fig. 4.19 Example of a folksonomy and derived bipartite and co-occurrence networks. (a) Two users (`alice` and `bob`) annotate three resources (`npr.org`, `wired.com`, `google.com`) using three tags (`news`, `web`, `tech`), resulting in seven triples. (b) Projecting the folksonomy onto resources and tags, we obtain a bipartite network. Edge weights correspond to numbers of triples, or numbers of users. The link from `npr.org` to `news` has a larger weight because both users agree on that annotation. (c) A tag co-occurrence network. The link between `web` and `tech` has a larger weight because of the similarity in their resources: the two tags co-occur with two resources, `wired.com` and `google.com`. (d) A resource co-occurrence network. The resources `wired.com` and `google.com` are more similar because they co-occur with two tags, `web` and `tech`.

work and in Figure 4.19(d) we project onto resources to build a tag network.[3] The direction of the links is lost in these co-occurrence networks, but we can preserve weight information by comparing how two tags are connected to the resources.

One way is to represent, say, a tag as a vector of resources $\vec{t} = \{w_{t,1}, \ldots, w_{t,n_r}\}$

[3] You can explore interactive hashtag co-occurrence networks from Twitter using a tool from the Observatory on Social Media at `osome.iuni.iu.edu/tools/networks/`. You can also generate animation movies showing how these networks unfold over time, using another tool at `osome.iuni.iu.edu/tools/movies/`.

where $w_{t,r}$ is the number of people tagging resource r with tag t, and n_r is the total number of resources. So each element of the tag vector is a weight representing the association of a resource with the tag: the weights of the links in Figure 4.19(b). Then we can compute the *cosine similarity* between two tag vectors (see Box 4.1) and use it as a co-occurrence link weight. If the two tags are used to annotate resources in similar ways, the weight is high; if they never co-occur, the weight is zero and the tag nodes are not linked.

4.6.3 Link Filtering

Dense networks are hard to visualize and study: they look like "hairballs" and many links are not significant. For these reasons, it is often helpful to prune links in a weighted network. This is especially needed in co-occurrence networks, were many links with low weight may result from noise. As an example, consider a network of documents or web pages where links are defined by text similarity — shared keywords from the textual content of the nodes. A pair of unrelated or weakly related documents are likely to be linked because they share a few generic keywords. In cases like this, we need a method to filter such links and obtain a sparser network with only the meaningful connections.

The simplest approach to prune a network is to remove all links with weight below some threshold. This method works well in many settings. However, there are also many cases in which filtering based on a global threshold does not work. To see why, consider a network with a heavy-tail distribution of link weights — a pretty common scenario in co-occurrence, traffic, and other weighted networks (recall Figure 4.16). The weights are so heterogeneous that it is impossible to find a good threshold: we are likely to preserve some links that are not significant, and/or disconnect many nodes with low strength. For such a node, low-weight links may be significant, even though links with the same weight may be insignificant for a node with much higher strength.

To get around this problem, we need to use different thresholds for different nodes. One way to accomplish this would be to define a threshold relative to the degree or strength of each node: we might keep only the 10% of links with largest weight for each node, or the links with largest weight that account for 80% of a node's strength. But even so, we cannot be sure that we are retaining all of the significant links, or whether we are retaining some that are not significant. A more principled approach is to find the *network backbone*, i.e., detect the links that carry a disproportionate fraction of each node's strength. These are the most significant links to be preserved. Box 4.3 provides details about this method. Figure 4.20 illustrates a backbone network extracted from a dense co-occurrence network.

Box 4.3 **Network backbone**

In networks with broad distributions of link weights, using a global threshold to prune links is inappropriate. Instead, we can use the weight fluctuations for each node to identify the links to be preserved — those that carry most of the weight. Given a node i with degree k_i and strength s_i, let us evaluate a link against a null model in which the weights are produced by a random partition of s_i among the k_i links. The probability that a link weight w_{ij} is compatible with the null model can be expressed by:

$$p_{ij} = \left(1 - \frac{w_{ij}}{s_i}\right)^{k_i - 1}. \tag{4.2}$$

The rule is to preserve the link if $p_{ij} < \alpha$, where α is a parameter that represents the desired significance level. Lower values of α lead to sparser networks, as fewer links are preserved. Since a link is connected to two nodes, we can obtain two values for p_{ij} by plugging the strength and degree of either node into Eq. 4.2. We can then use the larger or smaller of these values, depending on whether we wish to prune more or less aggressively. This link filtering procedure extracts a *network backbone*, which preserves the essential structure and global properties of the network.

4.7 Summary

Information networks, such as Wikipedia and the Web in general, have directed links. So do communication networks, including email and the Internet; transportation networks, such as air flights; and some social media, notably Twitter. Often there is data about traffic flowing between nodes, making these networks weighted as well. Network weights can be used to represent clicks, messages, packets, travelers, retweet counts, and so on. We explored several features of directed and weighted networks by focusing on several cases.

1 The Web forms a huge information network, with a virtually infinite number of pages as nodes, connected by hyperlinks. Browsers use the HTTP protocol to navigate links and download page content. This content is usually expressed in the HTML language, that specifies how rich content — text and embedded media — is presented.
2 We can study the structure of the Web graph, where each node represents a page or a website, using data collected by Web crawlers — programs that browse the Web automatically and allow us to reconstruct large samples of the network. The Web has a heavy-tail distribution of in-degree and ultra-

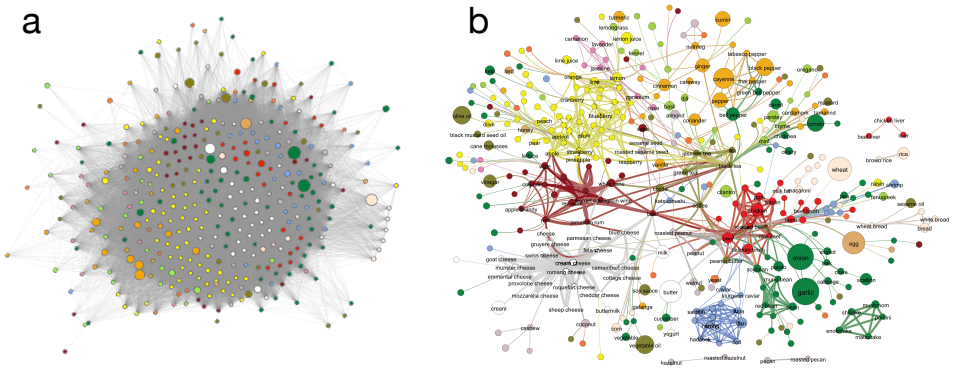


Fig. 4.20 A flavor network: each node denotes an ingredient, its color indicates a food category, and its size reflects the ingredient prevalence in recipes. Two ingredients are connected if they share flavor compounds, link width representing the number of shared compounds. (a) Full network. (b) Backbone network showing only the significant links identified by the backbone method with $\alpha = 0.04$. Images adapted from Ahn et al. (2011) under CC BY 4.0 license.

short paths facilitated by hugely popular hub pages. It also has a dominant connected component, with a giant strongly connected component within.

3 We can represent documents, such as Web pages, as high-dimensional vectors of words, and use the cosine between these vectors to measure text similarity among pages. In this way we can study topical locality, the relationship between network connections and page content. Because authors tend to link related pages, the Web has a clustered structure such that pages within a short distance of each other in the network are likely to be similar and semantically related.

4 PageRank is a node centrality measure based on a random-walk model of Web surfing, modified with random jumps. While people in reality do not browse the Web in such a random manner, PageRank can be applied to measure prestige of Web pages — or of nodes in any directed network. PageRank is especially important for its role in ranking search engine results.

5 Information diffusion networks arise when we share content on social media, for example by retweeting links, images and hashtags. The resulting cascade networks allow us to track the spread of news, ideas, beliefs, and even misinformation. The size and structure of these networks help us identify viral concepts. Using node out-strength and in-strength, we can characterize the roles of people who produce and consume information. Nodes with high strength, especially relative to their degree, flag accounts that are more active or influential. Social bots can be used to manipulate these networks.

6 Weighted networks often result from bipartite networks. The weight of a link

between two nodes **a** and **b** of the same type is a measure of how many nodes of another type are associated with both **a** and **b**. Co-citation/co-reference networks, product recommendation networks, and similarity networks based on shared words or tags are all examples of such co-occurrence networks.

7 Weighted networks, such as those obtained from traffic and co-occurrence data, can be very dense. Therefore it is often necessary to prune the network by filtering out the low-weight links. However, weighted networks typically have heavy-tailed weight distributions. In these cases, using a global weight threshold isolates the majority of the nodes. By determining a local weight threshold to identify the statistically significant links for each node, we can extract the backbone of a heterogeneous weighted network.

4.8 Further Readings

You can read about the vision, design, and history of the Web in a book coauthored by its inventor (Berners-Lee and Fischetti, 2000). To learn more about how search engines work, consider the textbooks on information retrieval by Baeza-Yates and Ribeiro-Neto (2011) or Manning et al. (2008). Liu (2011) goes into how to mine data from Web link, content, and usage networks; its chapter 8 focuses on Web crawlers.

Albert et al. (1999) first analyzed the average path length of the Web in 1999, based on a crawl of Notre Dame University websites. The Web was thought to contain about a billion pages back then, so the authors extrapolated from a logarithmic fit between average path length and (sub)network size to estimate that the Web had a diameter of 19 links. The following year, Broder et al. (2000) reported on the first systematic study of the Web structure. They measured the average path length on a much larger Web crawl with about $N = 10^8$ pages, in rough agreement with the earlier prediction. A more recent analysis of a much larger Web crawl by Meusel et al. (2015) suggests that the average path length scales sub-logarithmically (Figure 4.6).

Barabási and Albert (1999) reported the first measurement of the heavy-tail Web page in-degree distribution as fitting a power law with exponent $\gamma \approx 2.1$. Broder et al. (2000) later confirmed this from a larger crawl. Broder et al. (2000) also analyzed the bow-tie structure of the directed Web graph. Serrano et al. (2007) showed that the relatives sizes of the largest strongly connected component and in/out-components depend on the particular crawler used to reconstruct the Web graph.

Davison (2000) measured topical locality on the Web by comparing the content of pairs of pages selected at random, linked by a common predecessor (siblings), and connected by a hyperlink (in the same or different websites). Menczer (2004) extended this analysis by performing a breadth-first crawl to consider how content

and semantic similarity decays for pages within a certain distance of each other (Figure 4.8).

In the landmark paper that introduced the Google search engine, Brin and Page (1998) described PageRank and how it is used to rank search results. It turns out that the same measure of centrality had been proposed 50 years earlier by Seeley (1949) as a way to gauge the importance of a person in a social network. A related authority measure, based on a bipartite representation of the Web graph, was proposed by Kleinberg (1999). Fortunato et al. (2007) showed that the in-degree is a good local approximation of PageRank, and the two measures have the same broad distribution. For a review of the math behind PageRank, see Gleich (2015).

Dawkins (2016) introduced the concept of *meme* to mean a unit of information, belief, or behavior that can be transmitted from person to person. This was a precursor of the images, hashtags, and links that now spread on social media. Goel et al. (2015) proposed a definition of structural virality for memes and a method to reconstruct retweet cascade networks on Twitter. Studying these diffusion networks, Cha et al. (2010) showed that having a high degree (many followers) is not the only factor that affects a node's influence.

A strong polarization of communication networks on Twitter was observed by analyzing the diffusion networks of political hashtags, with segregated conservative vs. progressive communities (Conover et al., 2011b). This makes it easy to infer the political leaning of Twitter users (Conover et al., 2011a). Similarly, Shao et al. (2018a) find segregated communities sharing misinformation vs. fact-checking articles. A social network community with homogeneous opinions, isolated from different views, is sometimes referred to as an echo chamber (Sunstein, 2001) or filter bubble (Pariser, 2011).

Ratkiewicz et al. (2011) observed the earliest instances of fake news websites producing misinformation to be spread via social media. The factors that affect the viral spread of misinformation in social networks are the subject of extensive investigation (Lazer et al., 2018); they include novelty (Vosoughi et al., 2018) and amplification by social bots (Ferrara et al., 2016; Shao et al., 2018b).

Meiss et al. (2008) collected massive data about Web clicks to reconstruct a large Web traffic network, revealing the limitations of PageRank as a model of Web surfing. Better models account for bookmarking of popular starting nodes, backtracking (or browser tabs), and topical locality (Meiss et al., 2010). Meiss et al. (2008) also showed that the distribution of links weights has a heavy tail. Serrano et al. (2009) introduced the method to extract the backbone of networks with such heterogeneous weights.

Exercises

4.1 Go to `scholar.google.com` and search for publications on a topic that interests you. Pick two papers from the list of search results.

1 What is the in-degree of each of the two papers in the citation network?
2 What is the out-degree of each of the two papers in the citation network? (Hint: the papers must be available for access or download to answer this question.)
3 Download the two papers and analyze the lists of references. Calculate the co-reference (also known as *bibliographic coupling*) between the two papers.
4 For each of the two papers, look at the lists of other papers citing them (click on "Cited by..."). Calculate the co-citation between the two papers. (Hint: this can be tedious if you select two paper with too many citations.)

4.2 Go to the Wikipedia article on "network science" (`en.wikipedia.org/wiki/Network_science`).

1 What is the out-degree of this page in the Wikipedia network? (Hint: the "See also" section has links to other Wikipedia articles; it may not be present, meaning $k_{out} = 0$.)
2 Visit the successor nodes of the "network science" node in the Wikipedia network and report on how many of the outgoing links from this article are reciprocal.
3 Build the ego network of the "network science" node and find the largest strongly connected component.
4 What is the node in the "network science" ego network with the maximum out-degree?
5 What is the node in the "network science" ego network with the maximum in-degree?

4.3 Consider the Wikipedia "network science" ego network constructed in the previous problem. Select a handful of nodes in this network. Represent each of these nodes as a list of categories — these are found at the bottom of each Wikipedia article; an example is "Network theory." For each pair of nodes, calculate the cosine similarity between the category vectors. (Hint: a list is a vector where the weight of each category is one, and the weight of any category not in the list is zero.)

1 Which two articles from your sample are most similar to each other? What is the value of the cosine similarity?
2 Which two articles from your sample are least similar to each other? What is the value of the cosine similarity?

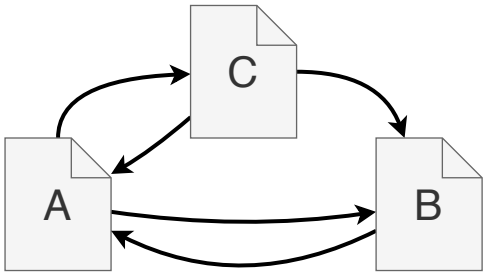


Fig. 4.21 A directed network representing a small website with three pages and their hyperlinks.

3 Do your measurements provide evidence of topical locality? Why or why not?

4.4 Consider the small network in Figure 4.21. Initialize the PageRank of each page with the value $R_0 = 1/3$. Apply Eq. 4.1 without teleportation ($\alpha = 0$) to calculate the values of PageRank in the next iteration ($t = 1$). Continue to update the values until convergence — assume the values have converged when there is no change in the third decimal digit of each node's PageRank. (Hint: be sure to use the values from the previous iteration when computing the new values; for example use the initial values ($t = 0$) when calculating the $t = 1$ values.) After how many iterations do the values converge? What are the final values of PageRank?

4.5 Repeat the previous exercise using the network in Figure 4.21, but this time use a teleportation parameter $\alpha = 0.2$. What is t at convergence and what are the PageRank values?

4.6 Go to the PageRank demo page at `carl.cs.indiana.edu/fil/cgi-bin/pagerank/` and insert a bunch of nodes and links. The demo calculates the PageRank and you can use it to measure the popularity of each node. Observe how the values change with the addition of new nodes and links.

1 This exercise is more fun if you do it in a large study group: each participant can use their own laptop, use an anonymous pseudonym in place of their name, enter their favorite colors, and link to their friends. Who is the most popular? What can you do to increase your PageRank? (Hint: consider alliances and be ruthless!)
2 This exercise can be conducted as a class exercise, with extra credit assigned every day to the top three students over a period of a few days.
3 The search feature of the demo works like a super-simplified search engine. Search for some colors. How does similarity between your query and a node's text content affect the ranking of the node? How does PageRank affect the ranking?

4.7 Download the Wikipedia dataset. Use NetworkX to load it as a directed network (DiGraph), then run the PageRank algorithm to compute the PageRank of each article.

1 What are the top 10 articles by PageRank?
2 Compare the top 10 articles by PageRank with the top 10 articles by in-degree? Are they the same? Why or why not?

4.8 Go through the Chapter 4 Tutorial on the book's GitHub repository.[4]

4.9 Test the friendship paradox (discussed in Chapter 3) on Twitter. Refer to the Chapter 4 Tutorial for usage of the Twitter API. Since Twitter is a directed network, we can formulate the friendship paradox in terms of in-degree/out-degree of a node's successors/predecessors. One version is to ask: Do your friends (the people you follow) have more followers than you, on average? In this case we want to measure the in-degree of your successor nodes in the follower network. If you do not use Twitter, you can answer this question using someone else's Twitter handle, such as `@clayadavis`.

1 Assuming `user` is the response from a query to `users/show.json` about `@clayadavis`, which of the following will give you the number of people followed by `@clayadavis`?

   a. `user['friends']['count']`
   b. `user['friends_count']`
   c. `user['followers']['count']`
   d. `user['followers_count']`

2 Verify if the friendship paradox holds in your case by calculating the average number of followers of your friends, and comparing it with your number of followers. If the account has more than the maximum number of friends that can be returned by the Twitter API with a single request (200 at the time of this writing), you will need more than one API call to obtain the full list of friends. Refer to the tutorial for using a cursor to get multiple pages of results. What is the average number of followers amongst all of your friends?
3 Mathematically, the friendship paradox makes a statement about averages: "On average, your friends have more followers than you do." A different statement would be "Most of your friends have more followers than you do." This second statement is about the median, not the average, and is in fact a stronger statement (less likely to be true). Is it true here? Do most of your friends have more followers than you do? To answer this question, measure the median number of followers amongst all of your friends.
4 Amongst your friends, what is the screen name of the user with the most followers?

4.10 Build the retweet network for `#RepealThe19th`, a controversial hashtag used during the 2016 US presidential campaign to advocate for the repeal of the 19th amendment to the US Constitution, which grants women the right to

[4] `github.com/CambridgeUniversityPress/IntroductoryNetworkScience`

vote). The data for this exercise is provided in the datasets directory of the book's GitHub repository. The file named `repealthe19th.jsonl.gz` includes 23,343 tweets containing the hashtag. After parsing the file, you should verify that you have this many tweets; a deviation from this number would flag parsing errors that may affect your answers. Refer to the Chapter 4 Tutorial for how to use these tweets to create a retweet network in order to answer the following questions. Keep the following in mind: (i) Link direction follows information flow: if Alice retweets Bob, there is a link from Bob to Alice. (ii) Remove self-loops; you can do so after creating the network, or modify your network creation code to not add them at all. This will definitely affect some answers.

1 How many nodes are in the retweet network?
2 How many links are in the retweet network?
3 What is the screen name of the node with highest out-strength in the network? What is their out-strength?
4 What is the screen name of the node with the second-highest out-strength in the network?
5 What is the screen name of the node with highest in-strength in the network? What is their in-strength?
6 Describe what the out-strength and in-strength values of these accounts tell you about their online behavior.
7 What is the ID of the most retweeted tweet? Note that, given a tweet ID, you can see the tweet by visiting the following URL in your browser: `https://twitter.com/statuses/<tweet-id>` replacing `<tweet-id>` with the numeric tweet ID.
8 How many nodes in the retweet network have zero out-strength?
9 Which of the following best describes what it means for a node to have zero out-strength in this network? Each of the statements applies only to the sample of tweets we used to build this network.

   a. The user produced no tweets
   b. The user hasn't retweeted anyone else
   c. The user hasn't been retweeted by anyone
   d. The user has no followers
   e. The user does not follow any other users

10 Which of the following best describes the connectivity of this network?

   a. Strongly connected
   b. Weakly connected
   c. Connected
   d. Unconnected

11 How many nodes are in this network's largest weakly-connected component?

4.11 You have a NetworkX graph called `G`, and you are about to draw it with the following command: `nx.draw(G, node_size=node_size_array)`. Which of the following is a correct way to obtain `node_size_array` so that the nodes are sized according to their degree?

a. `node_sizes = [G[n] for n in G]`
b. `node_sizes = [G.degree() for n in G]`
c. `node_sizes = [G.degree(n) for n in G]`
d. `node_sizes = [G.degree(n) for n in G.nodes_iter()]`
e. `node_sizes = [d for d in G.degree().values()]`

4.12 Build a retweet network for a hashtag that represents a topic in which you are interested. Refer to the Chapter 4 Tutorial for usage of the Twitter API. Use the Twitter search API to retrieve recent tweets about the hashtag. Make sure you have `'result_type': 'recent'` in the search parameters. Extract at least 1,000 tweets matching your search query; if there aren't that many tweets, start over and search for something else. Since this exceeds the maximum number of tweets that can be returned by the Twitter API with a single search request (100 at the time of this writing), you will need to use pagination. Finally, build the retweet network from these tweets.

1 Draw the retweet network. To make it informative, follow these guidelines: (i) The nodes should be sized in proportion to their out-degree. (Hint: see the previous exercise.) (ii) The links should be sized according to the number of tweets between the two users. (iii) Singleton nodes and self-loops should be removed. (iv) Use your judgement whether to show only the largest connected component or all components.
2 What is the screen name of the most retweeted user? Also report the hashtag you used.
3 In a few sentences, report on some interesting observation about this retweet network.

4.13 Consider a retweet network from one of the previous exercises, where link weights represent retweet counts. Prune the network by removing links with weights below a threshold ω.

- Recall the definition of density for a directed network from Chapter 1. Draw a plot showing how the network density (on the y-axis) decreases as a function of the weight threshold ω (on the x-axis).
- How much does the density decreases when you apply a threshold of $\omega = 5$ retweets?
- What is the value of ω such that the density is reduced to approximately half its initial value?