Universität Basel

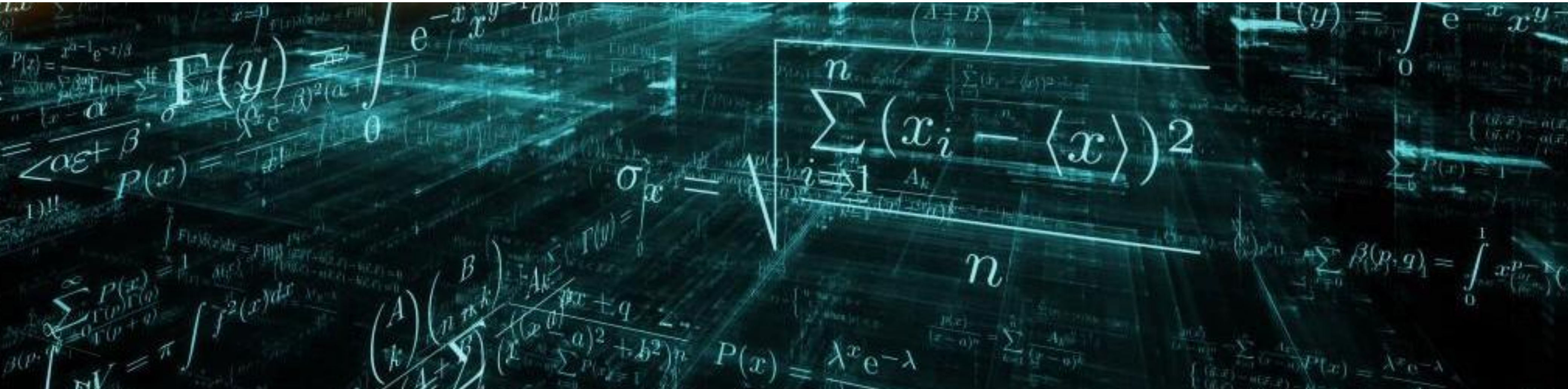
Computer Science / 15731-01 / 2025

Multimedia Retrieval

Chapter 9: Web Search

Dr. Roger Weber, roger.weber@gmail.com

9.1 Introduction
9.2 Additional Features
9.3 Page Rank
9.4 Hyperlinked-Induced Topic Search (HITS)
9.5 Literature and Links

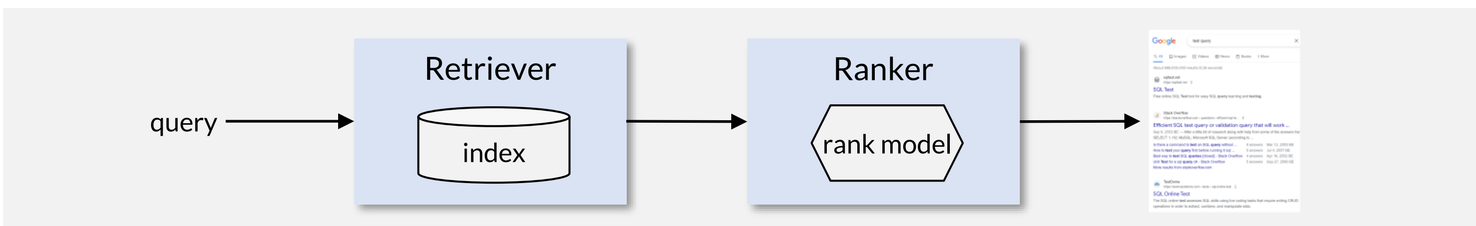


9.1 Introduction

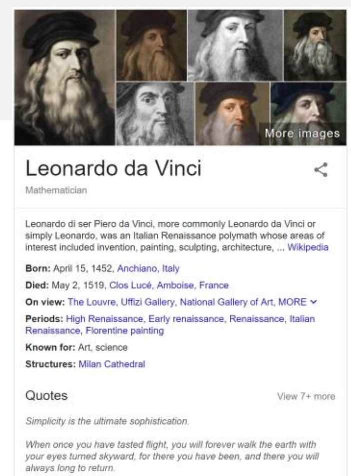
- The initial web search engines relied on traditional retrieval techniques, which were suitable for the early web stages. As web interest grew, it became clear that these methods were inadequate. Google was the first search engine to recognize web search as a distinct challenge and added improved ways to rank pages. It used a retriever-ranker structure with a basic Boolean search for finding web pages, similar to other engines. However, it ranked pages using newly developed features tailored to web users' unique interests and the rapidly evolving web structure.
- The web search experience has evolved over the years. Initially, users received a list of pages with ads in between. They had to refine queries and find answers within the documents. For example, a query like "Who is the US president?" would lead to the official US government site, but users had to extract the information themselves. This led them away from the search portal, which couldn't generate revenue through ads. To keep users on the site longer, search providers have enhanced the query experience in the following ways:
  - The initial enhancement introduced factual **knowledge cards** alongside or above search results. For fact-based queries, users received curated answers. The previous query would display a data card with the current US president, providing a convenient response for users. Likewise, product and brand queries showed ads and price summaries for common products. However, this was only possible for facts and products stored in the search provider's knowledge base.
  - The second improvement began with the introduction of **neural network-based reader models** as discussed in the previous chapter. A reader is a machine comprehension model that can pinpoint the most relevant passage on a page when given a natural language query. This enabled search engines to offer answers to nearly any natural language question, provided a search result page contained the relevant information. For example, when querying the US president, the search results now include a highlighted passage from one of them that contains the answer.
  - The third advancement came with the rise of transformer models and the power of large language models. As of the time of writing this, search engine providers are exploring **retrieval augmented generation (RAG)** along with **interactive chat** elements. As discussed in the previous chapter, RAG involves taking a few passages from search results, creating a prompt with context and the user query, and using a language model to generate a direct answer to the question. Users can also engage in chat to get more information and receive links to confirm the accuracy of the generated responses.

- An overview of the different stages of web retrieval over the course of time:
  - In the 1990s, early search engines like AltaVista and Yahoo operated with smaller data sets compared to today's vast web. They primarily focused on building catalogs (Yahoo) and basic keyword-based searching (AltaVista). Due to the limited number of search results in that era, classical ranking functions were prevalent, with some tweaks, such as domain boosts, to score documents.

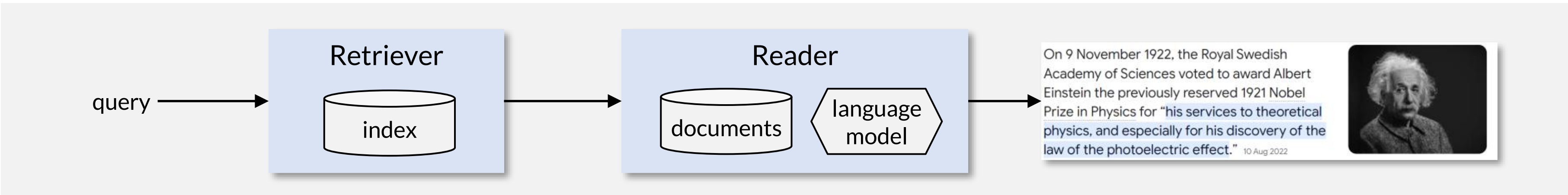
  - In 1998, Google originated from a Stanford research project and rapidly gained widespread usage with its innovative approach to web search. Being among the pioneers, Google understood the specific requirements of web users and optimized its ranking function for optimal results in common query types. It also introduced a distributed architecture, enabling more extensive crawling and simultaneous query handling compared to other search engines of the time. Most notably, it introduced PageRank a novel approach to re-rank results.



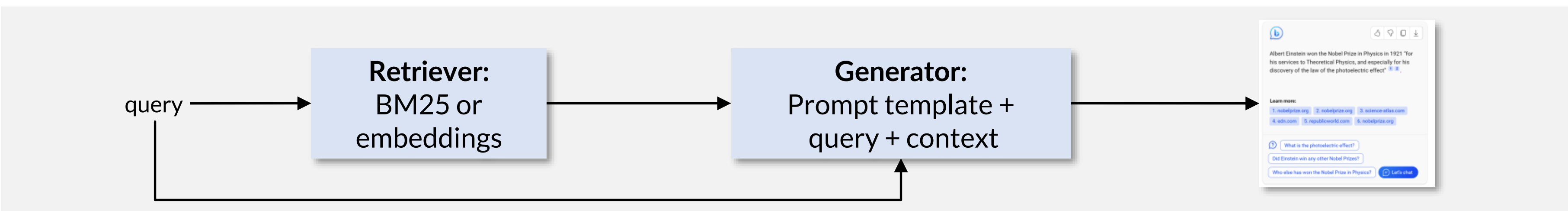
  - In 2012, Google launched the Knowledge Graph, a curated knowledge database containing facts from reliable sources. At its debut, it held 18 billion facts about 800 million entities. By 2023, it has expanded to include 800 billion facts on 8 billion entities. When a query aligns with an entity, a knowledge card appears alongside search results, providing users with verified information about that entity. This greatly benefited users who would otherwise need to sift through search results and linked documents to find their desired answers.



– Google has employed snippets since its beginning to provide contextual information from web pages in search results. Snippet generation is automated, but web authors can influence it using meta-information. In 2016, Rich Snippets were introduced initially for movies and recipes, later expanding to a wider array of queries. In 2020, Passage Ranking was introduced and went live in 2021. Unlike snippets, it uses a language model (BERT) to identify the most relevant passage in the top results for fact-based queries. This retriever-reader model aids users in quickly finding relevant content in lengthy documents. To enhance the web experience, most browsers now support the "scroll to text" feature, allowing users to jump to specific passages on a page from search results.

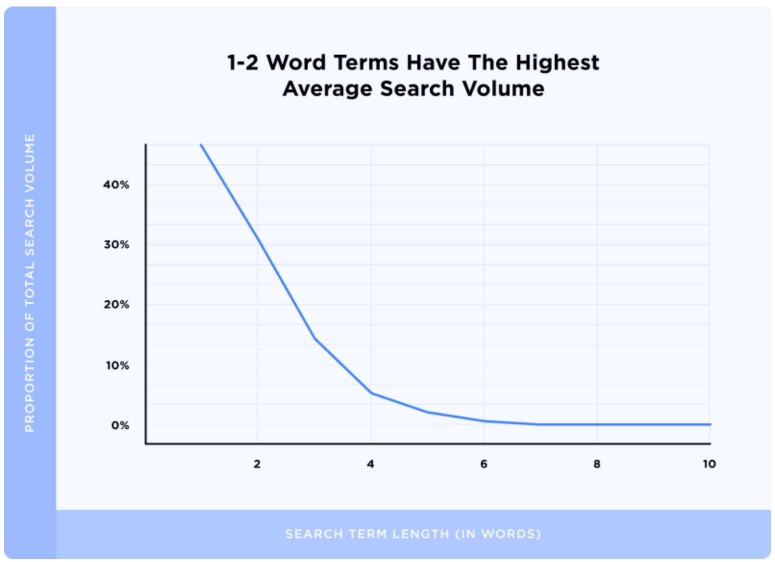


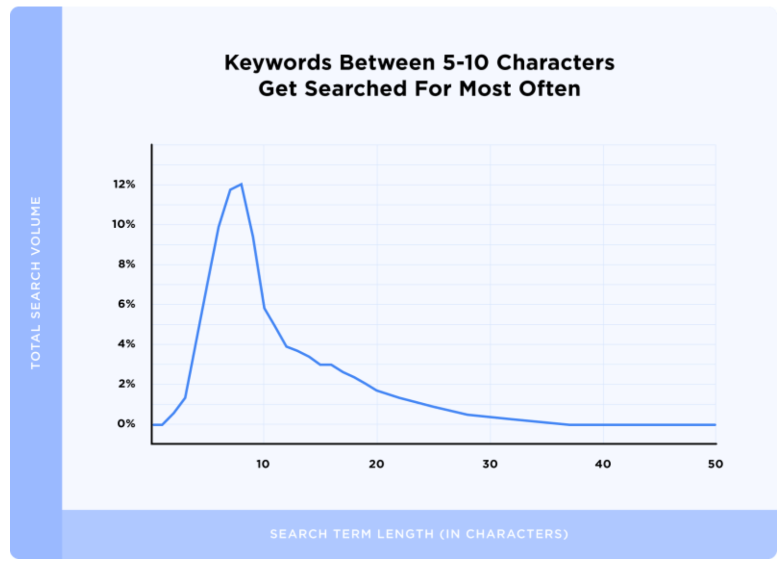
– In 2023, Microsoft, in a deepened partnership with OpenAI, introduced the next generation of web search on Bing. This innovation combines retrieval augmented generation (RAG) and interactive chat features. RAG utilizes a large language model to provide direct answers by generating responses from top search results and the user's query. The chat assistant offers source links for verification and allows users to ask follow-up questions in the same context. Google also launched Bard in 2023, a conversational AI chatbot based on Google's language model PaLM. Both providers face the challenge of the high cost of generating answers, and balancing the need for users to obtain good search results.



- Before we dive into the content, it is worth to review the differences between text retrieval and web search:
  - **Collection:** In classical retrieval settings, content is controlled and quality-assured. For example, online shop product catalogs undergo rigorous checks to maintain high quality and a positive user experience, ensuring data accuracy. Conversely, the web lacks such quality controls. While we may trust some sources more than others, the sheer scale of the web makes implementing quality controls or assessments impractical. Search engines must contend with a wide spectrum of quality, including poor grammar, (intentionally) incorrect information, as well as aggressive spamming and search result manipulations.
  - **Documents:** Classical retrieval, as seen in systems like Lucene, relies on a shared document structure for easy information extraction and structured or faceted search to refine results. The web lacks such a uniform structure. While many documents use common formats like PDF and HTML, extracting structural details beyond title, URI, and content proves challenging. Modern user experience methods, such as single-page applications, render large portions of the web inaccessible to search engines (referred to as the deep web), despite the information being publicly available. Some content can be accessed through deep links, but often requires the web crawler to locate a link to that page from another source.
  - **Queries:** Classical retrieval relies on specific query context. In a music database, queries typically relate to distinct object types like artists, producers, songwriters, albums, songs, or lyrics. In such narrow contexts, classical methods excel in providing relevant information. In contrast, a web search engine handles a wide array of query types, some comprising just 2-3 terms without any further context. Without structured data, search providers must not only locate relevant pages but also infer the implicit context of the query.
  - **Results:** Classical retrieval functions well on smaller scales but can also handle millions or billions of entries. It is easier to control result size, provide users with filters, facets, and sorting options for query refinement due to structured data and contextual boundaries. In contrast, a typical web search with 2-3 terms can yield billions of candidate documents, all containing the query terms, making it challenging to rank pages (as they all appear equally similar) or to provided faceted search. Thus, additional features are necessary to improve page ranking.
  - **Relationships:** The web lacks structure within documents but is a highly interconnected network with pages as nodes and links as edges. These links convey relationships between pages and identify influential sites ("go-to" destinations). This enables a web search engine to rank pages not just based on text but also using network topology. In contrast, classical retrieval contexts may have relationships among data items, but their structure is usually curated and provides minimal additional information. For example, relationship information (artist/song links for instance) in a music database may not reveal the popularity of songs.

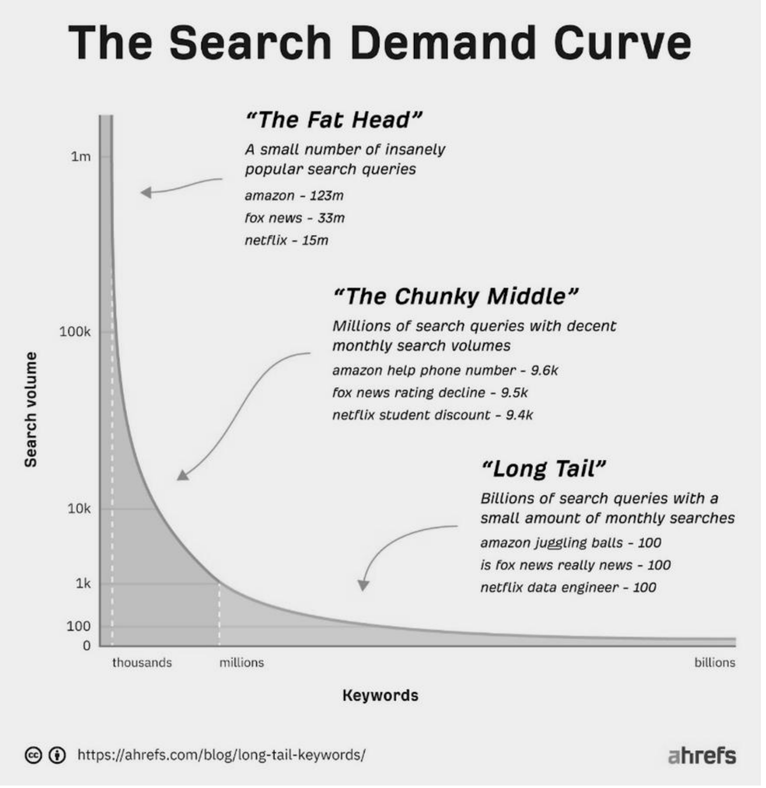
- What are the most common web query types? We can classify search queries into three categories:
  - **Navigational search queries:** users aim to reach a specific website or its particular section. Examples: "youtube," "amazon prime," "wikipedia," or "facebook login". While bookmarks were designed to address this specific need, many users rather employ a search engine to find the entry point for their favoured web application.
  - **Transactional search queries:** Users search for brands, products, hotels, restaurants, etc., with the intent to make a purchase or order goods and services. This covers a broad range of queries, including "new coffee machine" or "trip to Hawaii" where the transactional intent may not be immediately clear.
  - **Informational search queries:** Users seek news, events, sports, celebrities, or general information. Some queries are posed as direct questions: "who is the US president?", "how to make mac & cheese?", "where to find a lake for swimming", or "what is the weather tomorrow?".
- The following statistics, sourced from backlinko.com, ahrefs.com, and statista.com, have been published since 2020. While the numbers may have evolved, the fundamental distribution remains largely consistent. The initial statistics relate to query length, encompassing the number of terms and the character count (backlinko.com):

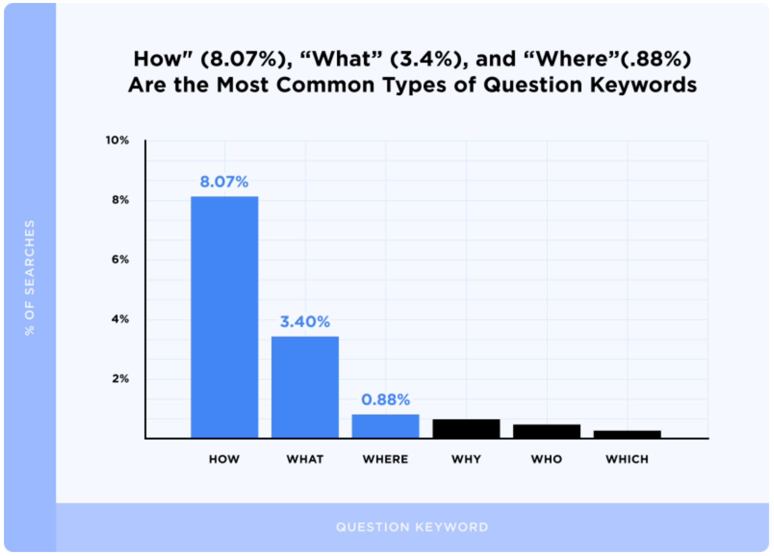




These are average values across different queries. Yet, as we will explore on the next page, queries are often repeated multiple times, and most frequently used queries in terms of search volume tend to be shorter in length.

- [ahrefs.com](#) examined query repetition frequency. As expected, the distribution adheres to Zipf's law, as displayed on the right: a small set of queries, mainly navigational ones, form the "Fat Head“, recurring millions of times. In the “Chunky Middle”, you find common queries appearing in thousands of searches, while the “Long Tail" includes all other queries that appear in fewer than 100 searches.
- According to [backlinko.com](#), 92% of unique query strings fall within the long tail. However, these long-tail queries account for a relatively small portion of the total search volume (3.3%). The majority of search demand is concentrated in a small percentage of high-volume terms. To illustrate, the top 500 most popular search terms constitute 8% of all search volume, and the top 2000 keywords are responsible for 12% of searches on both Google and Bing. The median search volume for a query is only 10 searches per month, meaning that 50% of queries are nearly unique for users, with minimal repetition over a month. These unique queries significantly influence users' choice of a search engine.
- Google handles trillions of searches annually, yet 15% of these queries (not search volume) are entirely new to Google. In other words, a significant portion of queries arises from news events, new products, or general interest in sports and people.
- According to [backlinko.com](#), 14% of queries are in question form. The most common question word is "how“, followed by "what“, "where“, and others (as shown on the right). However, this statistic does not encompass the intent of queries lacking a question word. For example, a name search could represent either a "who" (for people) or "what" (for products and services) question.





9.2 Additional Features

- In this section, we outline a few essential text related features for web retrieval. The following sections will delve into link information, expanding beyond web search to encompass a broader array of social media-related searches.
- Today, web search engines must handle over **40 billion pages, 60+ trillion unique URIs**, and an index size exceeding 100 PB. A standard query yields millions, or even billions, of results, yet users expect the top page or first link to be the most relevant one for their search context. However, for single or double-term queries and millions of candidate pages containing the terms, finding the most relevant ones based solely on term occurrences is not possible.
  - **Example query="ford":** does the user mean the car, the president, or a ford to cross the river?
  - **Example query ="uni basel":** which department or faculty is the user looking for? are they looking for courses, research, or maybe for a job?
  - **Example query="best pizza":** is the user asking for the best pizza type (funghi)? a pizza delivery service of that name? or a nearby restaurant with good pizzas?
  - **Example query="roger federer":** is the user looking for documents that contain "roger" and "federer"?
- A drawback of classical retrieval is its emphasis on term occurrences. Consider the query "roger federer" which many retrieval methods interpret as a search for documents containing the terms "roger" and "federer". While this approach may be suitable for classical retrieval, it falls short on the web, where pages may include these terms but not in close proximity (e.g., a marathon results page with numerous names).
- As an extension of the classical document descriptors, web retrieval considers:
  - proximity of query terms in documents (the closer the better)
  - term occurrence weighting based on tag
  - term boosting if it occurs in an anchor text of a link to the page
  - language and location awareness
  - penalties for low quality pages or block for entire sites (click baits, malicious content, link farms)

- **Proximity of query terms in documents:** In classical retrieval, we employed n-grams to expand our vocabulary with common term sequences aiding in expressing the closeness of query terms. However, n-grams are limited to a small set of sequences and can't adapt swiftly to new sequences arising from news or events. Given the importance of name and celebrity searches in web search, we require a method to factor in proximity when scoring at query time.
  - We enhance the postings in the index with position information, including the term's original position in the document's sequence. The posting then becomes a hit-list, as illustrated in this example:

    ```
    hitlist['white'] = [1, 13, 81, 109, 156, 195]
    hitlist['house'] = [2, 82, 112, 157, 189, 226]
    ```

    The hit-list replaces term frequency and lists all positions where the term appeared.
  - Let's consider a query with "white house". At query time, we create pairs of hits. Theoretically, we could generate the cross-product of all possible pairs, but it is adequate to only consider close matches (which may involve the same term in different pairs). Additionally, we aim to preserve the order of terms in the query. For the example mentioned, we might choose the following pairs:

    ```
    pairs = [(1,2), (81,82), (109, 112), (156, 157), (189,195)]
    ```

  - Next, we calculate the distance for each pair and create a histogram with n bins. Each bin corresponds to a given distance range. For example, the first bin includes pairs with a distance of 1 (adjacent). The second bin includes distances of 2 and 3, and so forth. We then count how often pair distances fall within each bin's range:

    ```
    pbins = [3,0,1,0,0,1,0,0,0,0]
    ```

  - Finally, we apply a scoring function to the bin distribution to enhance the document's ranking. A straightforward function involves a weighted sum over the bins, with higher weights assigned to closer proximity:

    ```
    weights = [89,55,34,21,13,8,5,3,2,1]
    score_proximity = ∑i pbins[i] * weights[i]
    ```

  - We can assume that more sophisticated functions are used to measure proximity, but the concept is the same.

- **Term occurrence weighting based on tag**: In classical retrieval, all term occurrences were considered equal. However, with HTML, we can distinguish the significance of an occurrence by its position in the document and the surrounding tags. For instance, a term in the title should carry more weight than one in a paragraph, as the author chose it to describe the page's content. Similarly, encounters of terms in `<h1>` or `<h2>` tags indicate the term's high importance.
  - Let's consider a page from the university of Basel for the term "university":

```
…<title> … university …</title>
…<h1> … university …</h1>
…<b> … university …</b>
…<p> … university …</p>
…<p> … university …</p>
…<p> … university …</p>
…<a> … university …</a>
…<b> … university …</b>
…<h1> … university …</h1>
```

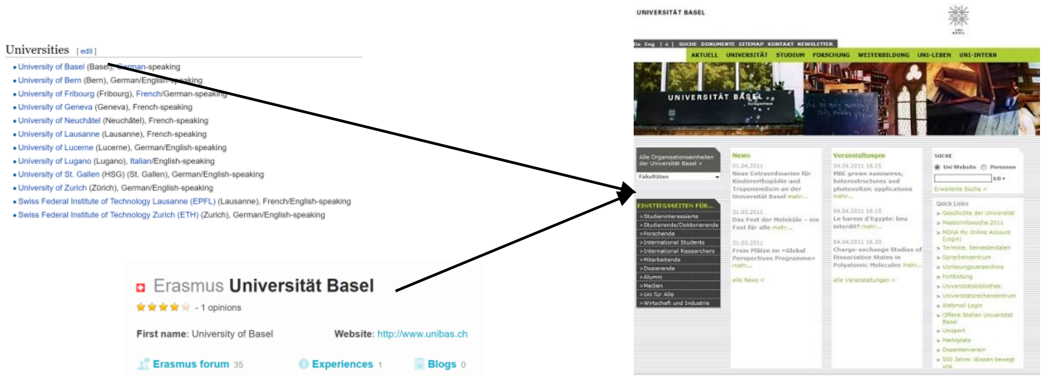
  The list on the right displays all instances of "university" along with the surrounding tags. In classical retrieval, we count 9 occurrences of the term, and with BM25, we utilize a saturation function on term frequencies to avoid excessive emphasis on frequent occurrences. In the web context, this saturation function is even more critical to safeguard against spammers attempting to manipulate scoring and rankings for specific target keywords.
  - In addition, we count each occurrence in tags separately. For the example above, we obtain:

```
terms = [(university,<title>,1),  (university,<h1>,2),   (university,<b>,2),
         (university,<p>,3),      (university,<a>, 1)]
```

  - The final term frequency is a weighted sum of each element in this list, employing a saturation function `sat_func(n)` on term occurrences to mitigate the impact of keyword spamming:

```
weights[tag → weight] = [<title> → 13, <h1> → 5, <p> → 1, <a> → 0.5]
tf[university] = Σterms[i,1]=university sat_func(terms[i,3]) * weights[terms[i,2]]
```

- **Term boosting if it occurs in an anchor text of a link to the page:** Anchor texts in a document typically provide precise descriptions of the linked page, although occasional generic terms like "click here" or "can be found here" may appear. With many thousands of links to a page, we will encounter numerous valuable keywords. While optimizing search results with term occurrences within a document is relatively straightforward, creating thousands or even millions of links (from various domains) to boost a term for a target page is much more challenging. At this stage, we are focusing on the terms used to describe the remote page, without considering the link as a relationship.
  – Let's consider a page from the university of Basel including two pages that refer to that page:

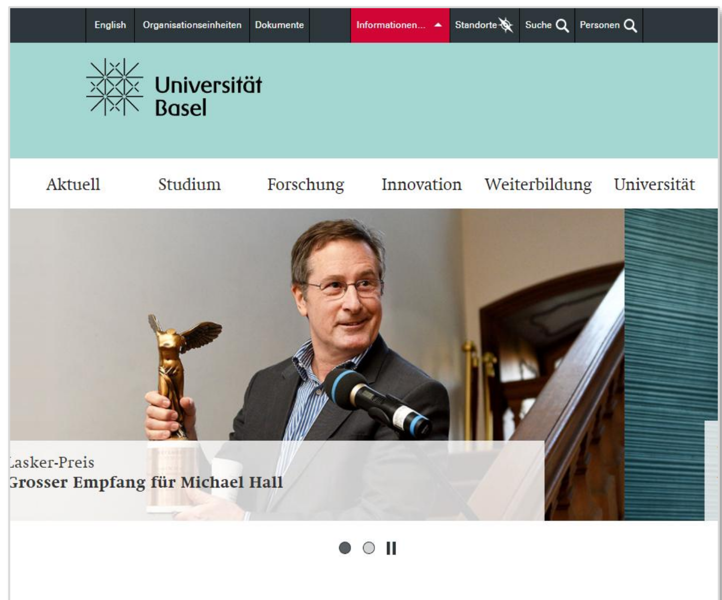


  First, how do we locate all links to a page for describing it with anchor keywords? During the crawling process, we extract link and anchor text, storing them in a dedicated database. Upon completion of the crawl process, we establish a separate index for all pages based on the anchor texts. We apply a saturation function to term occurrences, but one that accommodates larger numbers and may even grow indefinitely, such as a logarithmic function. This allows large numbers of pages to "vote" for a page and a keyword.
  – In the scoring process, we take this secondary index into account and derive a boost value for the document. As these occurrences are more challenging to manipulate than the terms within a document, we assign a greater weight to them. This does not imply blind trust in all anchor texts being accurate, but rather, we assume that there are sufficient reliable descriptions, often necessitating thousands or millions of anchor occurrences before a page receives a substantial boost.
  – A "Google bomb" is an attempt to manipulate this secondary score by orchestrating the inclusion of the same keyword in millions of links. In response, most search engines impose restrictions on the number of anchors per domain, with each domain having a maximum vote on how well a term aligns with a target page.

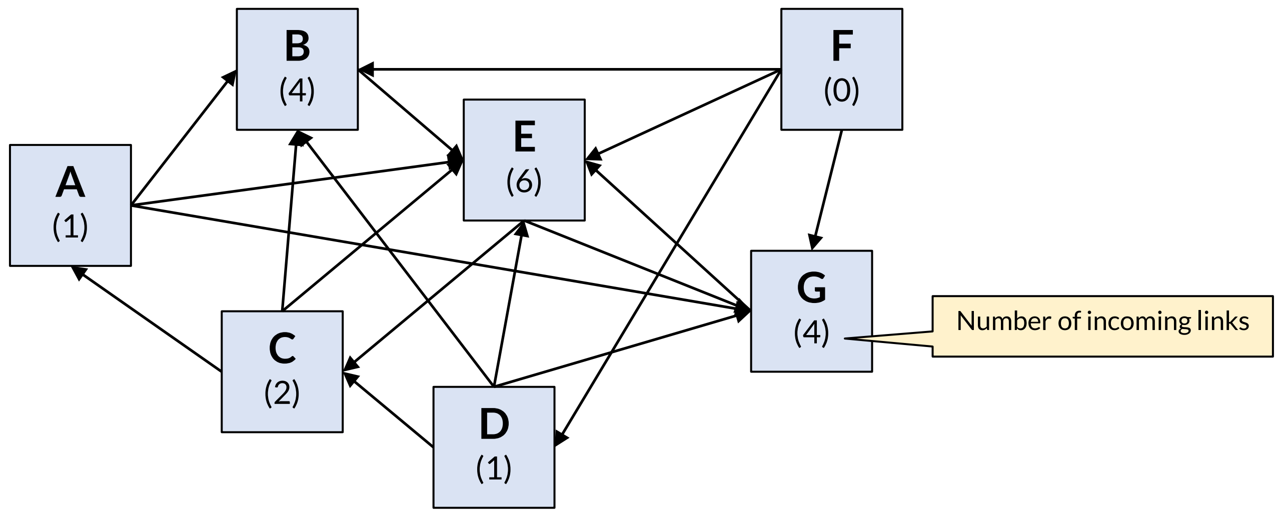
- **Language and location awareness**: In 2014, Google launched the Pigeon update, which modified its scoring function to prioritize local search results. It considered the user's location and the distance to businesses. Initially, the goal was to enhance results for queries like "where to eat pizzas?" as users sought nearby restaurants based on their current location, especially on mobile devices. Over time, this concept expanded to include products, places, and general searches, as long as the query didn't specify a particular location.
  - First, the search engine needs to collect (estimated) geolocation data for each page or website. For businesses like restaurants, owners often manage this information themselves to ensure accuracy on platforms like Google Maps. Alternatively, the crawler can assign an approximate location using the IP address and a managed database.
  - Second, the search engine considers either the exact geolocation from the user's device or an estimated location based on their IP address. However, this relies on users willingly sharing their current location and not using third-party tools to alter it, like VPNs. If the query explicitly mentions a location, such as the name of a city or country, that information takes precedence. For instance, a query like "best pizza in Paris?" produces different results than "best pizza?". The query analyzer identifies geolocation details through named entities and maps each entity to an accurate location. This applies not only to city names but also to landmarks, as seen in queries like "best pizza Niagara Falls?" or "best pizza Golden Gate Bridge?"
  - Finally, the search engine must determine the relevance of geolocation data for each query, considering the query's context and type. For instance, queries like "best pizza" receive a higher emphasis on location information, while queries for "Albert Einstein" prioritize global results but may still include local sites that host exhibitions related to Albert Einstein's work with a boosted local ranking.
- **Penalties for low quality pages or block for entire sites**: Search engine providers also monitor the internet to identify potentially harmful, illegal, or malicious content. They employ various techniques to detect such sites and impose penalties (e.g., for thin content with excessive ads or keyword spamming) or may even block sites entirely from search results (e.g., link farms, doorway pages, and illegal content). More recently, this includes maintaining global information about site trustworthiness. Trustworthiness can be determined from factors like grammatical accuracy, keyword usage, and known ownership of the page. Registered corporations, for example, receive higher trust scores compared to anonymous blogs. This practice has impacted nearly 12% of Google queries. However, it can make it challenging for newer websites to compete with established ones for top rankings.

9.3 Page Rank

- In this section and the next, we explore how web topology and link information can be leveraged to enhance web page rankings. Google pioneered this approach to assess the importance of domains and sites in relation to others.
  - Let's examine the two pages shown at the bottom. They are both potential results for a query like "uni basel". On the left, we have a page about a course, and on the right, we see the (former) main page of the unibas.ch domain. Even without considering link information, we can use term occurrences to rank these pages. Anchor text occurrences, in particular, may assist in distinguishing between the two pages. We can assume that there are more links with anchors containing the query terms for the right-hand page.
  - In addition, we can consider the web's topology. A basic approach would be to count the number of links pointing to each of these two pages and boost the one with a higher count. However, this simplistic approach does not consider the quality of the links. For example, a link from a Wikipedia article should carry more weight than a link from an anonymous blog. Using PageRank, we can obtain a global ranking for all pages, which can be applied to each query to enhance the page's ranking. Nevertheless, we need to carefully balance this boost with other ranking factors to prevent that the same pages dominate the rankings, regardless of the search terms used.



- PageRank was developed by Larry Page, a Google co-founder, during his Stanford research time. He aimed to establish a global ranking system, known as PageRank, for all web pages, assessing their perceived importance to users. This metric relies solely on link data and operates independently of specific queries, serving as a page-level boost rather than affecting keyword-based scoring.



  - A simple approach focuses solely on incoming links. In the given example, E would rank highest due to its six incoming links. B and G would come next with four links each, followed by C with two links, D with one link, and finally F with no incoming links. This aligns with our intuitive perception, as E seems to be the central node.
  - Now, let's examine B and G. They both have four incoming links, placing them in a tie for second place. However, upon closer inspection, we notice that G is referenced by E, while B is referenced by less significant pages. This means that G seems more important than B. We want to factor in the source quality of the link in our importance metric. In other words, merely considering incoming links is not sufficient; we aim to assign a weight to the link based on the importance of its source.
  - It is important to note that using the number of incoming links as a ranking measure is not very robust. A webpage author can easily generate thousands of incoming links to manipulate the page's ranking, a practice known as link farming. Hence, we seek a method that is robust against manipulations and external search result optimizations.

- PageRank builds on a random surfer model, where a user explores the web by clicking on links. In this model, the user can also randomly switch to another page, for instance from their bookmarks. This model operates under the following assumptions:
  - While on a page, users can perform two actions: 1) with a probability α, they will follow one of the links on the page. 2) with a probability of $1-\alpha$, they will enter a random URL, typically from their bookmarks.
  - For 1), the user navigates by randomly selecting a link (all links have an equal chance of being chosen).
  - For 2), if a page has no outbound links (making it a sink), the user selects a random URL.
- PageRank can be viewed as a Markov chain with pages as states and links as transitions. PageRank for a page represents the likelihood of a random surfer visiting that page. Higher PageRank implies greater popularity, indicating more frequent visits by web users, making the page a strong contender for top search results.
  - To compute a single page's probability, we examine two events: 1) reaching the page through an incoming link, and 2) a random switch to the page. We denote a link from page q to p as $q \rightarrow p$ and use $L(q)$ to represent the number of outgoing links from q. The set of all pages $\mathbb{P}$ contains $N = |\mathbb{P}|$ elements. PageRank, considering the probability α that a user follows a link, can be expressed as:

$$PR(p) = \frac{1-\alpha}{N} + \alpha \cdot \sum_{q \rightarrow p} \frac{PR(q)}{L(q)} \qquad \forall p \in \mathbb{P}$$

  - **Interpretation:** Achieving a high PageRank depends on the number of incoming links, but each link's importance is determined by the PageRank of the linking page. When a page has multiple outgoing links, its PageRank is distributed equally among the linked pages. On the other hand, it favors older pages that are well connected while new pages, even very good ones, lack the number of links necessary to obtain high ranks.

- **Evaluation:** The PageRank equation on the previous page establishes an implicit system of equations that we can solve iteratively. Let $\boldsymbol{r} \in \mathbb{R}^N$ be the vector containing the PageRanks of all documents in $\mathbb{P}$. The connections between pages are represented by a adjacency matrix $\mathbf{M} \in \mathbb{R}^{N \times N}$ given as follows:

$$M_{i,j} = \begin{cases} \dfrac{1}{L(p_j)} & \text{if } p_j \rightarrow p_i \\ \dfrac{1}{N} & \text{if } p_j \text{ has no outgoing links} \\ 0 & \text{otherwise} \end{cases}$$

With this matrix $\mathbf{M}$, we can rewrite the previous PageRank formula as follows:

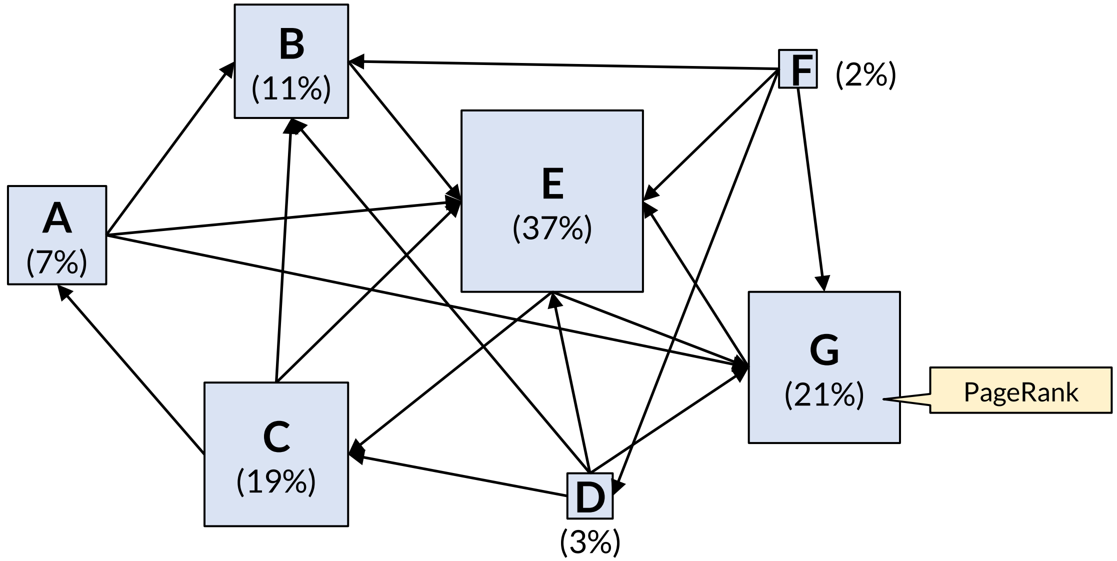
$$\boldsymbol{r} = \frac{1-\alpha}{N} \cdot \mathbf{1} + \alpha \cdot \mathbf{M}\boldsymbol{r} \qquad \text{with } \mathbf{1} \text{ being a column vector of length } N \text{ with ones in all components}$$

We now can describe the iterative process to solve this equation system. Let $\boldsymbol{r}^{(t)}$ denote the PageRank values of pages after the t-th iteration:

1. Initialization: $\boldsymbol{r}^{(0)} = \frac{1}{N}$, $\alpha = 0.85$
2. Iteration:
   - $\boldsymbol{r}^{(t+1)} = \frac{1-\alpha}{N} \cdot \mathbf{1} + \alpha \cdot \mathbf{M}\boldsymbol{r}^{(t)}$
   - stop if $\left| r^{(t+1)} - r^{(t)} \right| < \epsilon$

Due to the sparse adjacency matrix, the iteration converges rapidly and is scalable to huge document sets. In their original study, Brin and Page reported 52 iterations for a network with 322 million links and 45 iterations for 161 million links. They found that the number of iterations is linear in relation to the logarithm of the number of edges, denoted as n. Compressed representations are employed to reduce memory consumption of the sparse matrix.

- **Example from before:** Let's use the PageRank formula on the graph below. The node sizes now represent the PageRank, and the values indicate the probabilities of a random surfer visiting the page:

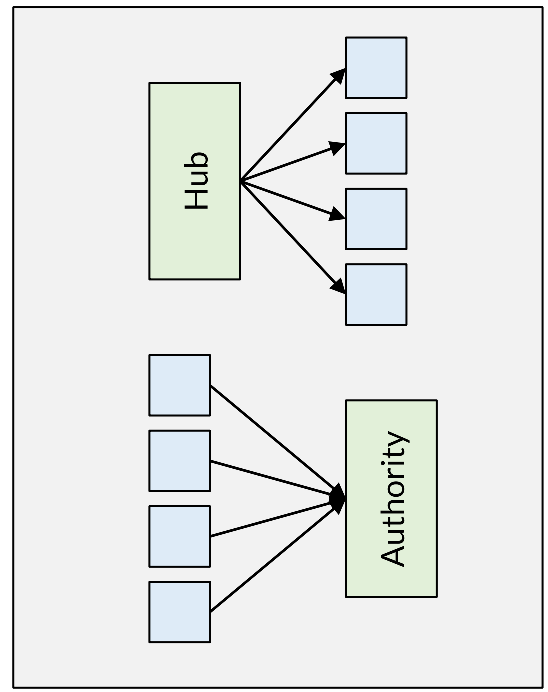


  - **Discussion**: E is still the center of the network but G and C are now more important than B. Even though B has 4 incoming links, two of them come from the least important pages D and F. On the other side, E has only two outgoing links and hence both C and G receive about 43% (with $\alpha = 0.85$) of the PageRank of E.

- **Usage for ranking:** PageRank serves as an absolute measure of a page's importance, irrespective of the specific query. It is calculated once following a comprehensive crawl and applies to all queries. While PageRank is a crucial metric, it is just one of several criteria. Overemphasizing PageRank would lead to repetitive search results. Terms and proximity are equally significant, but PageRank contributes to promoting pages that are more likely to be visited by users, thus pushing them tothe top of search results.

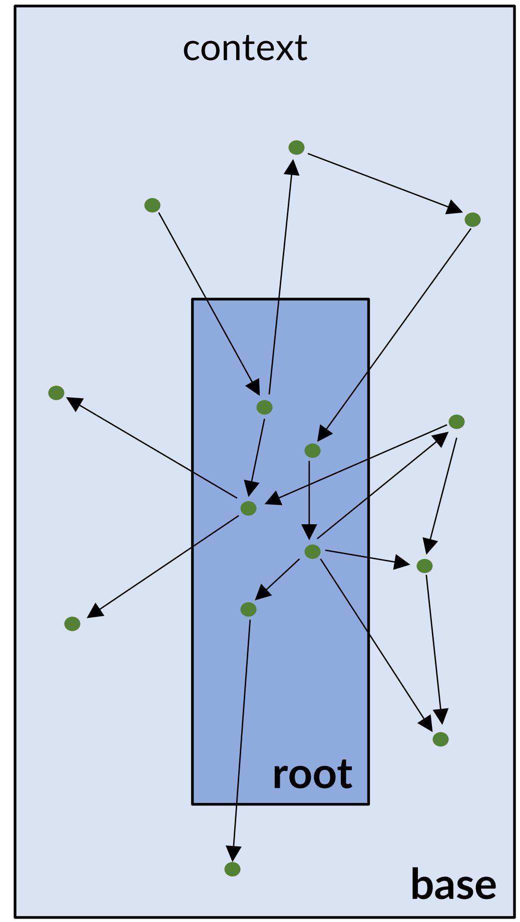
9.4 Hyperlinked-Induced Topic Search (HITS)

- There are different ways how to interpret link information during feature extraction. A common observation is that there are two prototypes of web pages:
  - **Authorities** are web pages acknowledged by the community as the definitive sources for specific topics. Prominent examples include Wikipedia, IMDb, and MusicBrainz.
  - **Hubs** are web pages that organize authorities in a directory-like fashion without providing in-depth information on the topics. Examples include personal bookmark collections and websites such as Yahoo and Yellow Pages.
- It is essential to note that PageRank focused on the likelihood of a user visiting a page but did not assess the presence of authoritative content for a given query. PageRank is a query agnostic random walk across the entire web. In this section, we discuss methods that specifically examine the query's current topic, which is why they are often referred to as Topic Search methods.
- How can we identify a quality hub and a reputable authority?
  - A hub is a web page with numerous links to recognized authorities. This is characterized by the typical hub structure.
  - An authority is a web page with numerous incoming links from hubs. This exhibits a common sink structure.
  - For a hub to be considered good, it must link to reputable authorities on the relevant topic. Conversely, a good authority is one linked by many well-regarded hubs on the same topic.
  - It is crucial to emphasize that "on the topic" signifies that we are not just concerned with the quantity of incoming or outgoing links; they must also be relevant to the current subject. This distinction sets it apart from PageRank, where all links are considered regardless of the topic.



- Jon Kleinberg introduced the HITS algorithm in 1997. He recognized the significance of hubs and authorities in the early web landscape, which relied heavily on directories for information retrieval since search engines were not yet of sufficient quality. To enhance search guidance, he introduced two metrics for a web page, denoted as p:
  - $h(p)$ denotes the hub value of the page p, i.e., its ability to serve as a hub for the current context
  - $a(p)$ denotes the authority value of the page p, i.e., its ability to provide content to the current context
- Unlike PageRank, HITS operates within a user-defined context set by a base query. Therefore, it applies to a carefully selected portion of the web, consisting of relevant pages to the query and a set of related pages. The construction of this graph functions as follows:

  1. For a query Q, identify the top results using a search engine, forming the root set (dark blue set in the picture on the right). This set already includes several hubs and authorities, but not all relevant ones are included yet.
  2. Expand the root set by adding a) web pages that link to a page in the root set and b) pages that are referenced by a page in the root set. Links within the same domain (navigation links) can be excluded, and the number of incoming and outgoing links can be limited to maintain a manageable graph size. This expanded set is referred to as the base set (light blue set including the dark blue set in the figure on the right).



- For evaluation, we can choose to either maintain a comprehensive link matrix for the web to perform action 2a), or utilize search engines with a link-clause in their query interface to retrieve pages linking to a specific one. For 2b), it involves downloading pages from the root set to extract linked pages. An alternative approach is to use our own link matrix as for step 2a).

- We use the notation $p \rightarrow q$ to signify that page p contains a link to page q. Now, let's outline the HITS algorithm as an iterative process, considering a base set $\mathbb{P}$ comprising N pages:

1. Initialization: $h^{(0)}(p) = \mathrm{a}^{(0)}(\mathrm{p}) = \sqrt{1/N} \;\; \forall p \in \mathbb{P}$
2. Iteration:
   - Update: $a^{(t+1)}(p) = \sum_{q \rightarrow p} h^{(t)}(q)$ $\qquad h^{(t+1)}(p) = \sum_{p \rightarrow q} a^{(t)}(q)$
   - Normalize $a(p)$ and $h(p)$ such that: $\sum_{p} a^{(t+1)}(p)^2 = \sum_{p} h^{(t+1)}(p)^2 = 1$
   - Stop if $\sum_p \left| a^{(t+1)}(p) - a^{(t)}(p) \right| + \sum_p \left| h^{(t+1)}(p) - h^{(t)}(p) \right| < \epsilon$

Once computed, we can provide the top hubs (with the highest $h(p)$ values) and the top authorities (with the highest $a(p)$ values) to the searcher.

- We can represent the equations in matrix notation. Define $\boldsymbol{h}$ as the vector of hub values for all pages and $\boldsymbol{a}$ as the vector of authority values. We can create the adjacency matrix $\mathbf{A}$ from the graph as follows:

$$A_{i,j} = \begin{cases} 1 & \text{if } p_i \rightarrow p_j \\ 0 & \text{otherwise} \end{cases}$$

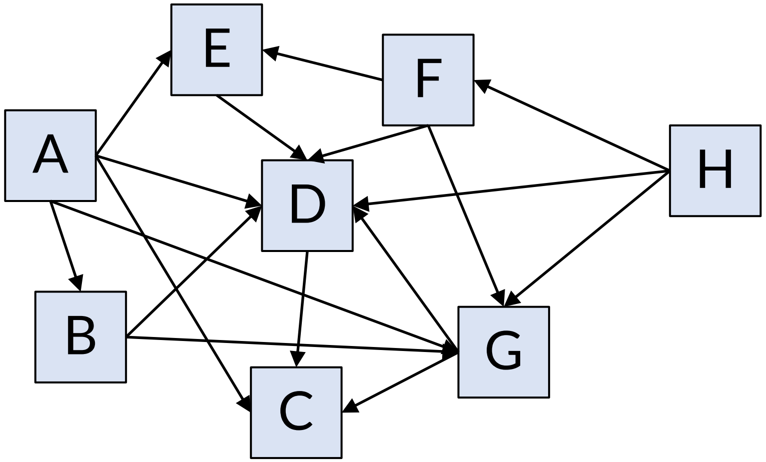
The rows of $\mathbf{A}$ include all outgoing links with a 1, and the columns include all incoming links with a 1. This allows us to rewrite the computational process for the iteration as follows:

$$\boldsymbol{h}^{(t+1)} = \mathbf{A}\boldsymbol{a}^{(t)}$$

$$\boldsymbol{a}^{(t+1)} = \mathbf{A}^{\top}\boldsymbol{h}^{(t)}$$

Since the adjacency matrix is sparse, we should use an sparse matrix representation and optimized libraries for efficient matrix vector multiplications with sparse matrices.

- **Example:** consider the following graph



Adjacency Matrix

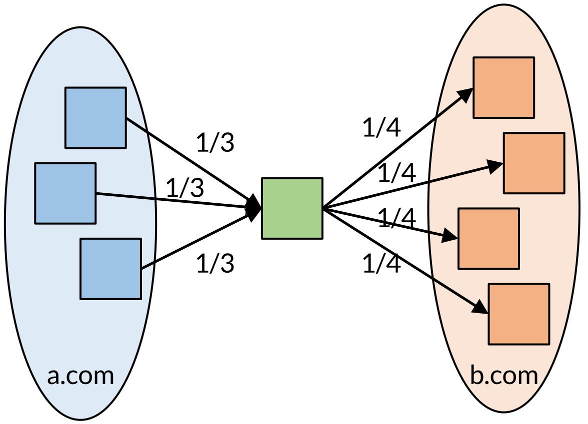
	A	B	C	D	E	F	G	H		*h*	*a*
A		1	1	1	1		1			60%	0%
B				1			1			36%	18%
C										0%	29%
D			1							9%	69%
E				1						20%	31%
F				1	1		1			46%	12%
G			1	1						29%	54%
H				1		1	1			40%	0%

  – We can observe that A is the best hub since it links to the best authorities, namely D, G, and E.
  – E is a slightly better authority than C, even though it has only 2 incoming links compared to C's 3. The reason for this is that E is referenced by the good hubs A and F, while C is referenced by the good hub A, the average hub G, and the bad hub D.
  – It's important to note that it's not always clear whether a page should be classified as a hub or an authority. For example, B functions as a bad authority and an average hub. C is neither among the top authorities nor a good hub. G, on the other hand, is a top authority but also has some hub value.
  – Lastly, remember that we are examining the base set of nodes, which includes pages related to the topic. Unlike PageRank, hub and authority values can change with different queries or topics.

- In conclusion, the HITS algorithm, developed by Jon Kleinberg in 1997, introduced the concepts of hubs and authorities to the realm of web search. This method provided a valuable alternative to the PageRank algorithm, which mainly considers the global importance of web pages. HITS operates within a user-defined context, focusing on a specific query or topic, making it highly adaptable to different search scenarios.
  - HITS helps us distinguish between pages that serve as authoritative sources of information and those that act as hubs, aggregating these authorities. By focusing on the interplay between these two roles, HITS identifies valuable content in relation to a given query. This approach is particularly useful in cases where query relevance extends beyond the number of incoming or outgoing links, placing a strong emphasis on topic relevance.
  - Compared to PageRank, which offers an absolute measure of a page's importance across the entire web, HITS is more query-specific and topic-sensitive. The HITS algorithm allows us to evaluate pages based on their relevance to the user's query, ultimately improving the quality and precision of search results.
  - While HITS has been a valuable addition to web search algorithms, it's essential to recognize that there is no one-size-fits-all solution. Both PageRank and HITS serve unique purposes, and their effectiveness may vary depending on the specific requirements of a search engine. As search technologies continue to evolve, having a diverse set of algorithms like HITS in our toolkit enables us to provide more tailored and accurate results for users.

- The HITS algorithm faces three key issues:
  1. **Overpowered Authorities and Hubs**: If an entire domain links to the same external page, it becomes an excessively dominant authority. Similarly, if a page links to numerous pages within the same domain, it becomes an overly strong hub.
  2. **Automated, non-related Links**: Links established automatically, like advertisements or web host/designer links, can grant linked sites an inflated authority value, even if they are off-topic.
  3. **Query Term Frequency:** Queries like "jaguar car" tend to over-emphasize the more frequent term, "car" in our example. This can overshadow less common terms, although the intent might be to fine-tune the more generic term. In the example "jaguar car", the most authoritative pages could contain any car manufacturer, and not necessarily only the brand Jaguar.
- Monika Henzinger et. al refined the base algorithm in 1998 to tackle the three problems mentioned above.

- The first enhancement deals with domains to reduce overpowered authorities and hubs. Instead of each page casting a single vote for the authority (hub) of an external page, the entire domain casts a single vote. Let's adjust the iteration of HITS by incorporating these weights as follows:
  - If there are k pages q_i in a domain linking to a page p, then we adjust the authority formula for page p by weighing the hub values, where $aw(q_i, p) = \frac{1}{k}$.
  - Similarly, if a page p links to l pages q_i in the same domain, we adapt the hub formula for page p by weighing the authority values, where $hw(p, q_i) = \frac{1}{l}$.



$$a^{(t+1)}(p) = \sum_{q \to p} aw(q,p) \cdot h^{(t)}(q) \qquad h^{(t+1)}(p) = \sum_{p \to q} hw(p,q) \cdot a^{(t)}(q)$$

- The second improvement focuses on the topic and introduces penalties for pages deviating from the topic, particularly to filter out automated links. To promote less frequent terms, a $tf \cdot idf$ scheme similar to methods in vector space retrieval is employed:
  - We create a reference document C using terms with the highest $tf \cdot idf$ values from all documents in the root set.
  - Compute a similarity value $s(p)$ for each page p using the $tf \cdot idf$ vectors of p and the reference document C, i.e., $s(p) = \frac{\boldsymbol{c}^\top \boldsymbol{p}}{\|\boldsymbol{c}\| \cdot \|\boldsymbol{p}\|}$ with the cosine similarity measure.
  - Set a threshold t, and remove all pages p with $s(p) < t$ from the base set. The median of all $s(p)$ values, which eliminates 50% of pages from the base set, is a useful threshold.
  - Utilize the similarity values $s(p)$ to adjust the distribution of authority and hub values to a page. We adapt the iteration of the HITS algorithm accordingly:

$$a^{(t+1)}(p) = \sum_{q \to p} aw(q,p) \cdot s(q) \cdot h^{(t)}(q) \qquad h^{(t+1)}(p) = \sum_{p \to q} hw(p,q) \cdot s(q) \cdot a^{(t)}(q)$$

This extension has resulted in a 45% improvement over the original HITS algorithm.

9.5 Literature and Links

Papers

- Brin, S.; Page, L. (1998). **The anatomy of a large-scale hypertextual Web search engine**. Computer Networks and ISDN Systems. 30: 107–117. http://infolab.stanford.edu/pub/papers/google.pdf
- Page, Lawrence; Brin, Sergey; Motwani, Rajeev; Winograd, Terry (1999). **The PageRank citation ranking: Bringing order to the Web**, published as a technical report on January 29, 1998.
- Lempel, R.; Moran S. (April 2001). **SALSA: The Stochastic Approach for Link-Structure Analysis**. ACM Transactions on Information Systems. 19 (2): 131–160. https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.5859
- G. Jeh and J. Widom. **SimRank: A Measure of Structural-Context Similarity**. In KDD'02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 538-543. ACM Press, 2002. http://ilpubs.stanford.edu:8090/508/1/2001-41.pdf
- J. Dean, M. Henzinger, **Finding related pages in the World Wide Web**, Computer networks 31 (11), 1467-1479. http://www.ra.ethz.ch/cdstore/www8/data/2148/PDF/PD1.PDF
- J. Kleinberg. **Hubs, Authorities, and Communities**. ACM Computing Surveys 31(4), 1999. http://www.cs.brown.edu/memex/ACM_HypertextTestbed/papers/10.html