

EECS 498 System Design of a Search Engine

Winter 2026

Lecture 17: Ranking

Nicole Hamilton

[https://web.eecs.umich.edu/~nham/
nham@umich.edu](https://web.eecs.umich.edu/~nham/nham@umich.edu)

agenda

1. details
2. ranking

agenda

1. details
2. ranking

details

1. Please read my Dynamic Ranking paper, posted on Canvas.
2. The Robots.txt assignment is posted.
3. It's more "paint by numbers" than I'd like but I'm late giving it to you and I'd like to keep the work reasonable. (If they let me teach this again, I may make it harder the next time.)
4. If you've already started your own design, I'm hopeful my interface is generic enough you can work with it.
5. I'm almost finished writing the AG tests. It'll test functionality, memory leaks, and performance.

```

class RobotsTxt
{
public:

    // Constructor to parse the robots.txt file contents into a
    // set of rules.

    RobotsTxt( const Utf8 *robotsTxt, const size_t length );

    // Destructor

    ~RobotsTxt( );

    // Check a full URL against the rules in this robots.txt.
    // Return true if access is allowed. Optionally report
    // any crawl delay.

    bool UrlAllowed( const Utf8 *user, const Utf8 *url,
        int *crawlDelay = nullptr );

    // Check the path portion of a URL against the rules in this
    // robots.txt. Return true if user is allowed to crawl the path.
    // Optionally report any crawl delay.

    bool PathAllowed( const Utf8 *user, const Utf8 *path,
        int *crawlDelay = nullptr );

    // Get the list of paths specified in any Sitemap statements.

    std::vector< std::string > Sitemap( );
};

```

tee shirts

We are very close to announcing the tee shirt.

It's now a yellow octopus on a faded cream color shirt.

You'll order and pay online but then they'll print and ship them at once



agenda

1. details
2. ranking

the problem

For any given query, matching pages likely fall into a pyramidal distribution.

1. A small number of really good ones.
2. More that are of middling quality, and
3. An enormous number that are terrible.

ranking

Objective is to order pages the same way a human would do using software.

Do a calculation based on what's known about:

1. All the documents and words in the index.
2. A matching page.
3. Match between the query and the page.

ranking

This has to be done without knowing:

1. How many other pages match,
2. What they look like,
3. Or even if they exist.

It must also be reasonably consistent across individual nodes to allow the results from different nodes to be compared and merged.

ranking

1. The rank value should obey the desired ordering relationship, that a better page will get a better score.
2. Since search engines typically broadcast a query to a large number of machines with differing fractions of the web and then combine the results, the calculation should be consistent even though the samples might be a little different.
3. Beyond ordering, the value is otherwise meaningless: If page A's rank is twice that of page B, it does not mean that A is twice as good as B.

Only two ways to get better at ranking:

1. Have more or better information.
2. Make better use of it.

static vs. dynamic rank

Static

Quality of the page independent of the query, e.g., PageRank, length of the URL, title or page, domain (.gov or .edu vs. .biz), whether it contains images, pornographic content, etc.

Dynamic

Quality of a page as possible result for a specific query considering both static rank and the quality of the match between the query and the page.

static rank

Some pages are just better than others before you know anything about the query.

Static rank

Some domains are better than others, e.g., .gov or .edu over .biz.

Short URLs are better.

Short titles are probably better.

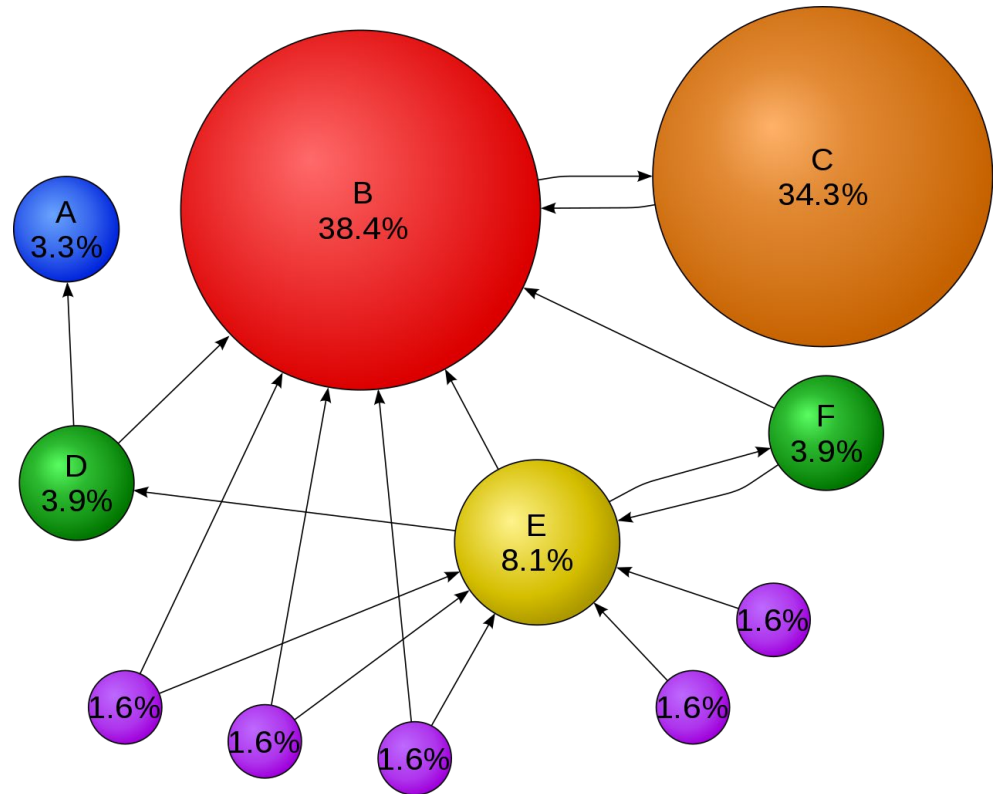
Some pages may be obvious spam.

Some pages may have lots of other pages pointing to them.

PageRank

A detour into the world's most famous link-analysis algorithm.

The basic idea: The more and better links to a page, the more likely it should rank higher.



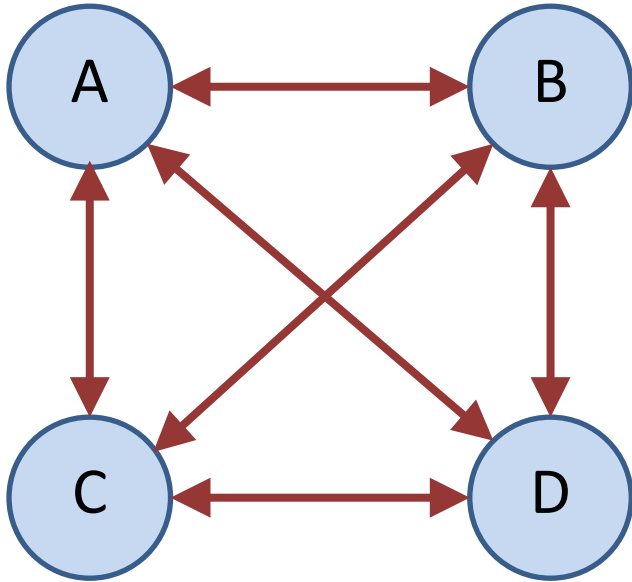
PageRank random surfer

The model was that people surfed the web, somewhat randomly either clicking on one of the links on the page or going somewhere else.

If an important page pointed to yours, some of that importance should bleed onto yours.

basic PR algorithm

1. PR output is a vector of probabilities that a person randomly clicking links will arrive at a particular page.
2. Initially all probabilities usually assumed equal (or maybe not!)
3. Links from a page to itself are ignored.
4. Multiple links from one page to another are treated as a single link.
5. The PR transferred from one page to another is its PR divided by number of pages it links to.
6. At each iteration, the new PR of a given page is calculated as the sum of the PRs transferred to it.
7. Repeat until it settles.

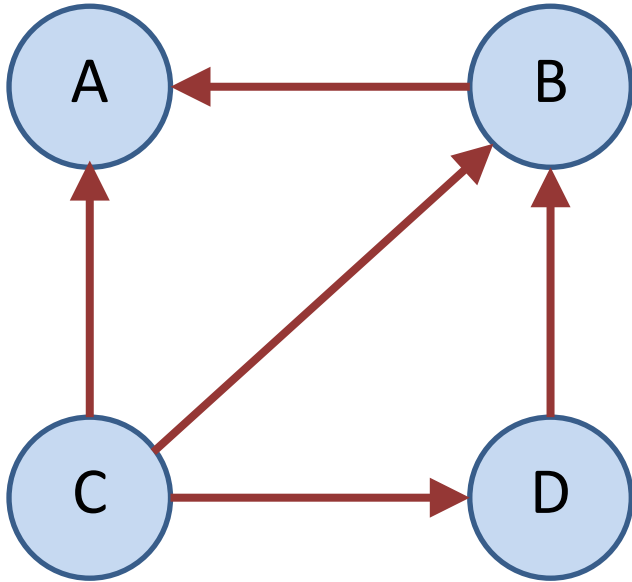


$$PR(A) = \frac{PR(B)}{3} + \frac{PR(C)}{3} + \frac{PR(D)}{3}$$

$$PR(B) = \frac{PR(A)}{3} + \frac{PR(C)}{3} + \frac{PR(D)}{3}$$

$$PR(C) = \frac{PR(A)}{3} + \frac{PR(B)}{3} + \frac{PR(D)}{3}$$

$$PR(D) = \frac{PR(A)}{3} + \frac{PR(B)}{3} + \frac{PR(C)}{3}$$

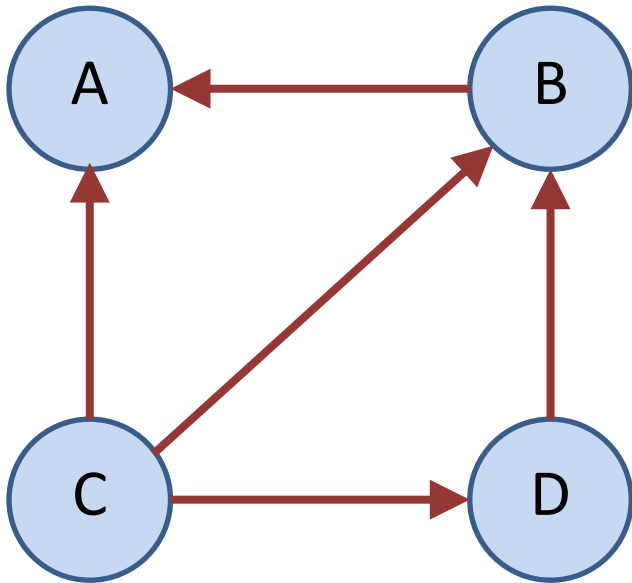


$$PR(A) = \frac{PR(B)}{1} + \frac{PR(C)}{3}$$

$$PR(B) = \frac{PR(C)}{3} + \frac{PR(D)}{1}$$

$$PR(C) = 0$$

$$PR(D) = \frac{PR(C)}{3}$$



$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

Where

u is a page,

$PR(u)$ is the PageRank of page u ,

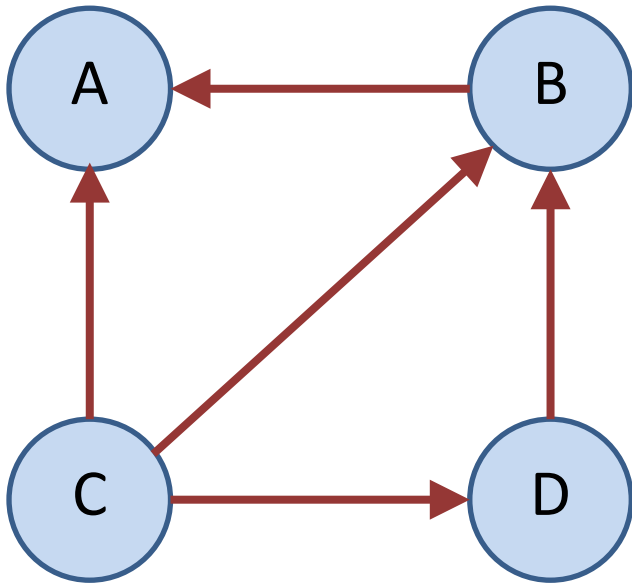
B_u is the set of all pages that link to u ,

$L(v)$ is the number pages linked from v .

damping factor

But in this basic version, PR sinks could happen, where at every iteration, a site just got more and rank.

They solved this by adding the notion that their imaginary surfer randomly clicking links will eventually stop clicking and simply start over at some other random page. This became a damping factor in PageRank.



$$PR(u) = \frac{1-d}{N} + d \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

Where

u is a page,

$PR(u)$ is the PageRank of page u ,

B_u is the set of all pages that link to u ,

$L(v)$ is the number of pages linked from v ,

d is the damping factor, typically ~ 0.85 .

PageRank

It obviously did work and Google got better results.

It also gave halo of special legitimacy to their results, that they were scientific and unbiased.

at Microsoft

We gamely expected our version of PageRank to represent about half the overall rank value, largely based on the hype around it.

Turned out it was very expensive to calculate and represented only a small part of the final rank score.

Mark Najork of [Mercator](#) fame argued for lumping whole domains together in something called DomainRank. But it hadn't yet worked when I left.

Ranking process

1. Compile the query.
2. Search the index for matching pages.
3. Return a list of the n best with scores indicating estimated quality.
4. May also return debug information to allow the scoring calculation to be examined.

Question

1. How should you find the n best?
2. Should you get the entire list and then sort?

question

1. How should you find the n best?
2. Should you get the entire list and then sort?

No, you should probably insertion sort into array of n elements.

decorating

Add characters that get stripped out during HTML parsing to indicate special characteristics or types of posts, e.g.,

amazon	amazon in the body text
#amazon	amazon only in the URL
@amazon	amazon only in the title
\$amazon	amazon only in the anchor text
%	End-of-document token.

Might also be used for *stemming*:

swim*	swim, swims, swimming, etc.
-------	-----------------------------

decorating vs. attributes

Use decorating when you'd like to use it for searching, to shrink the size of a post or because you'd like to separate the scoring for hits in the title vs. the body for example.

Use attributes when the ranker will want the information about each post and it could be different every time.

Matches in order of importance:

1. Anchor text.
2. URL.
3. Title.
4. Body.

Exception is Japan where URL matches are less useful due to mismatch between Kanji or Katakana queries and transliterated URLs.

URLs may need dictionary word-breaking or regex-style matching to be useful. Again, beyond the scope.

Does it matter how many ISRs you use?

How much effect will this have on query search time?

What dominates search time?

basic ranking

1. Matching pages found by the constraint solver but that only finds the page and the static information about the page from the enddoc.
2. Queries are flattened. (Very hard to estimate the probability of phrases or other combinations of OR'ing and AND'ing terms.)
3. ISRs are reset to the beginning of the document, then advanced through the page, extracting data about where the search words were found.
4. Three strategies from there.

queries are flattened

These queries all match different sets of pages:

apollo moon landing

(apollo | moon) landing

“apollo moon landing”

But for scoring, they’re all flattened to the same list of search words.

three strategies

1. Bag of words: The more hits the better.
2. Heuristics: Look for exact matches, matches in the right places, hand or machine-tuned.
3. Machine learning, typically with a neural net.

bag of words

Count the number of matches of each of the search words, typically weighted by the frequency of the word within the corpus.

Two most famous:

1. tf-idf
2. BM25

tf-idf

Term-frequency, inverse document frequency.

Bag of words technique. The more occurrences of a rare word, the better.

Combined:

1. Term weighting based on frequency invented by to Hans Peter Luhn in 1957.
2. Statistical interpretation invented by Karen Spärck Jones in 1972.

Term frequency

$$TF(t) = \frac{\text{Number of times } t \text{ appears in a document}}{\text{Total number of terms in the document}}$$

Inverse document frequency

$$IDF(t) = \frac{\log_e(\text{Total number of documents})}{\text{Number of documents with term } t}$$

Tf-idf is the product. The more the better.

$$Tf - idf(t) = TF(t) * IDF(t)$$

Okapi BM25

Okapi Best Matching function, a similar bag of words technique.

$$score(D, Q) = \sum_{i=1}^n IDF(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})}$$

$f(q_i, D)$ is q_i 's term frequency in Document D.

$|D|$ is the length of the document in words.

$avgdl$ is the average document length.

k_1 and b are free parameters you get to choose.

Typically

$$k_1 \in [1.2, 2.0]$$

$$b = 0.75$$

bag of words

Problem is, bag of words techniques simply don't work very well on the web because they don't do well at distinguishing quality, especially, to find the best match.

They simply cannot distinguish that a page with all the search words in the right order, as an exact phrase, or near the top of the page is better than a page where the words are randomly scattered.

It's also susceptible to keyword spamming.

bag of words

Here's a sample NY Times page from Jan 25, 2021.

The New York Times - Breaking News x +

File | C:/Users/hamil/Google%20Drive/eecs440/ExampleCode/HtmlParserSolution/nytimes.html

Seattle PI WSJ WSJ NY Times WaPost Canvas Course Info Center Crappy Scheduler BofA Wikipedia GSI/IA Applications Student evals Patents Other bookmarks

U.S. INTERNATIONAL CANADA ESPAÑOL 中文

The New York Times

Monday, January 25, 2021
Today's Paper

29°F 31° 26°
Nasdaq -0.02% ↓

SUBSCRIBE FOR \$1/WEEK LOG IN

World U.S. Politics N.Y. Business Opinion Tech Science Health Sports Arts Books Style Food Travel Magazine T Magazine Real Estate Video

The Daily Listen to 'The Daily' Impeached, again.

The Book Review Podcast Charles Yu on his National Book Award-winning novel, "Interior Chinatown."

The Argument Listen to 'The Argument' Jane Coaston loves Section 230. Got a problem with that?

LIVE

Fractures in G.O.P. Widen After Impeachment Vote

- Some House Republicans have called on Representative Liz Cheney to resign from her leadership post after she voted to impeach President Trump.
- Ms. Cheney was one of 10 Republicans to break with the party on the vote, setting up a messy internal feud that could define its future.



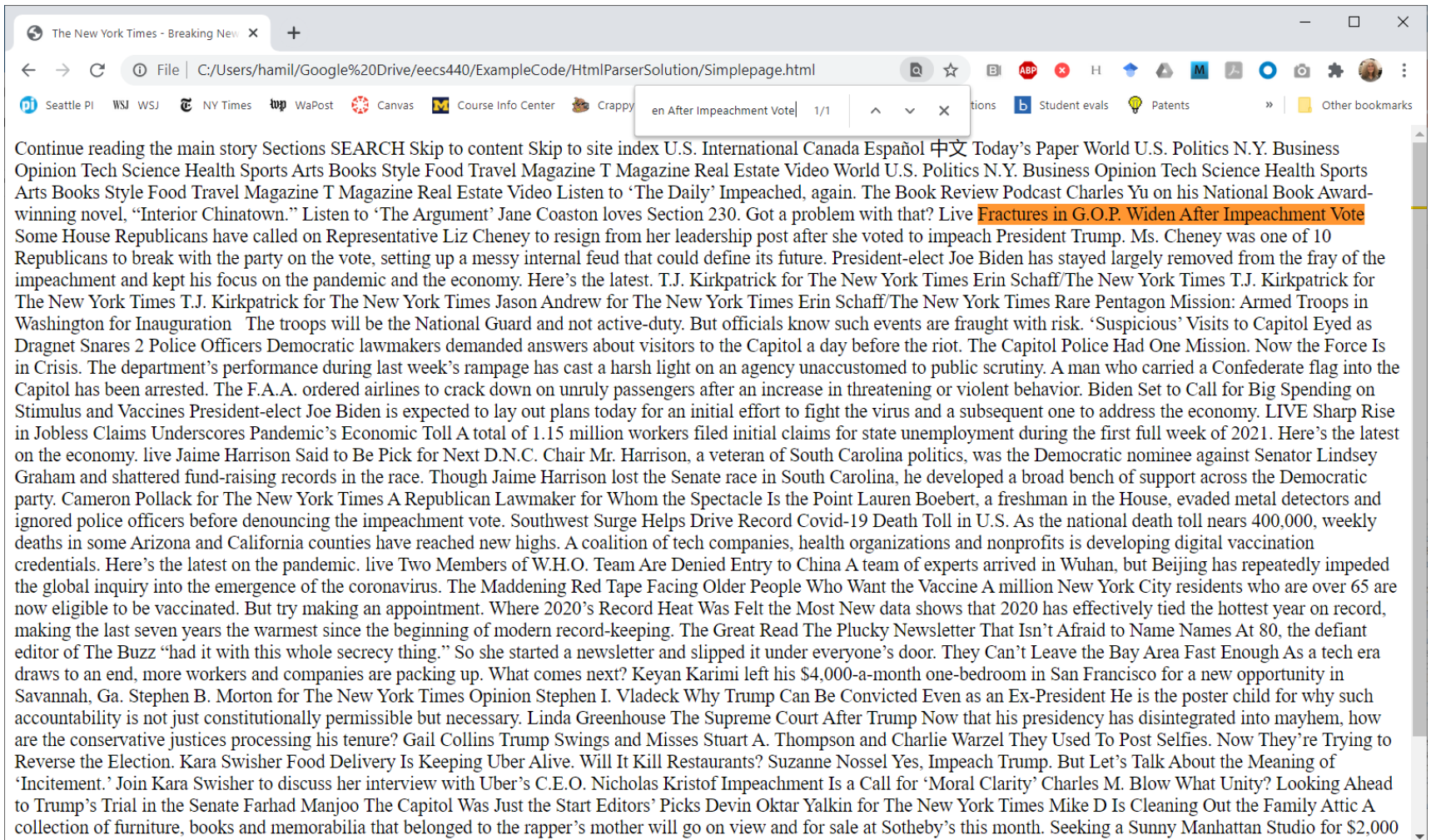
'Suspicious' Visits to Capitol Eyed as Dragnet Snares 2 Police Officers

Democratic lawmakers demanded answers about visitors to the Capitol a day before the riot.

<https://www.nytimes.com/2021/01/08/books/review/podcast-charles-yu-interior-chinatown-david-brown-henry-adams-last-american-aristocrat.html>

bag of words

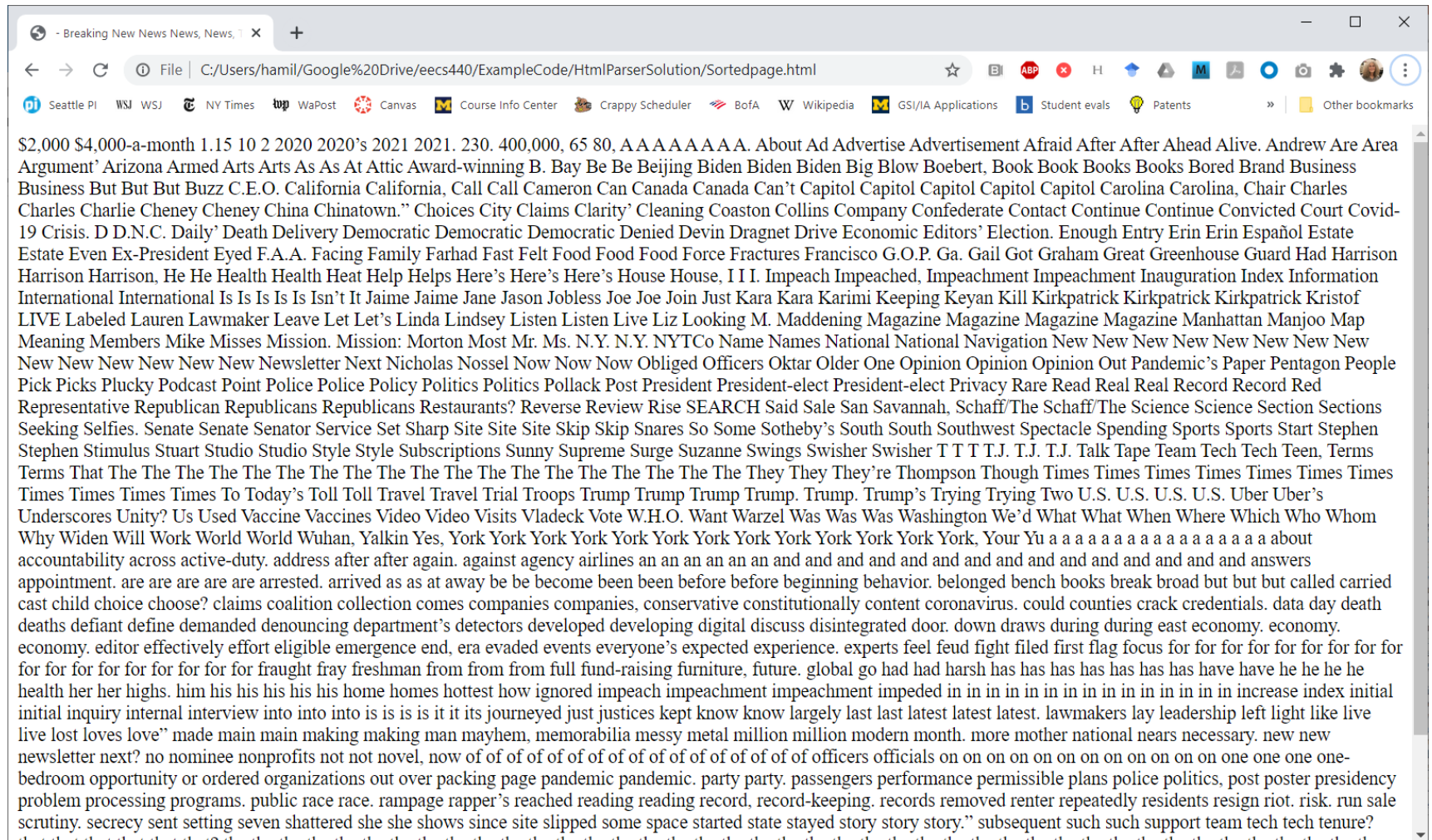
Here it is stripped of HTML and CSS but the text remains.



The screenshot shows a web browser window with a single text file open. The address bar indicates the file is located at `C:/Users/hamil/Google%20Drive/eecs440/ExampleCode/HtmlParserSolution/Simplepage.html`. The browser's bookmark bar is visible, showing links to various news and educational sites. The main content area displays a long, continuous block of text that has been stripped of all HTML and CSS formatting. The text is a collection of words and phrases, including "Continue reading the main story", "Sections SEARCH Skip to content", "Skip to site index", "U.S. International Canada Español 中文", "Today's Paper", "World U.S. Politics N.Y. Business Opinion Tech Science Health Sports Arts Books Style Food Travel Magazine T Magazine Real Estate Video World U.S. Politics N.Y. Business Opinion Tech Science Health Sports Arts Books Style Food Travel Magazine T Magazine Real Estate Video Listen to 'The Daily' Impeached, again. The Book Review Podcast Charles Yu on his National Book Award-winning novel, 'Interior Chinatown.' Listen to 'The Argument' Jane Coaston loves Section 230. Got a problem with that? Live Fractures in G.O.P. Widen After Impeachment Vote Some House Republicans have called on Representative Liz Cheney to resign from her leadership post after she voted to impeach President Trump. Ms. Cheney was one of 10 Republicans to break with the party on the vote, setting up a messy internal feud that could define its future. President-elect Joe Biden has stayed largely removed from the fray of the impeachment and kept his focus on the pandemic and the economy. Here's the latest. T.J. Kirkpatrick for The New York Times Erin Schaff/The New York Times T.J. Kirkpatrick for The New York Times T.J. Kirkpatrick for The New York Times Jason Andrew for The New York Times Erin Schaff/The New York Times Rare Pentagon Mission: Armed Troops in Washington for Inauguration The troops will be the National Guard and not active-duty. But officials know such events are fraught with risk. 'Suspicious' Visits to Capitol Eyed as Dragnet Snare 2 Police Officers Democratic lawmakers demanded answers about visitors to the Capitol a day before the riot. The Capitol Police Had One Mission. Now the Force Is in Crisis. The department's performance during last week's rampage has cast a harsh light on an agency unaccustomed to public scrutiny. A man who carried a Confederate flag into the Capitol has been arrested. The F.A.A. ordered airlines to crack down on unruly passengers after an increase in threatening or violent behavior. Biden Set to Call for Big Spending on Stimulus and Vaccines President-elect Joe Biden is expected to lay out plans today for an initial effort to fight the virus and a subsequent one to address the economy. LIVE Sharp Rise in Jobless Claims Underscores Pandemic's Economic Toll A total of 1.15 million workers filed initial claims for state unemployment during the first full week of 2021. Here's the latest on the economy. live Jaime Harrison Said to Be Pick for Next D.N.C. Chair Mr. Harrison, a veteran of South Carolina politics, was the Democratic nominee against Senator Lindsey Graham and shattered fund-raising records in the race. Though Jaime Harrison lost the Senate race in South Carolina, he developed a broad bench of support across the Democratic party. Cameron Pollack for The New York Times A Republican Lawmaker for Whom the Spectacle Is the Point Lauren Boebert, a freshman in the House, evaded metal detectors and ignored police officers before denouncing the impeachment vote. Southwest Surge Helps Drive Record Covid-19 Death Toll in U.S. As the national death toll nears 400,000, weekly deaths in some Arizona and California counties have reached new highs. A coalition of tech companies, health organizations and nonprofits is developing digital vaccination credentials. Here's the latest on the pandemic. live Two Members of W.H.O. Team Are Denied Entry to China A team of experts arrived in Wuhan, but Beijing has repeatedly impeded the global inquiry into the emergence of the coronavirus. The Maddening Red Tape Facing Older People Who Want the Vaccine A million New York City residents who are over 65 are now eligible to be vaccinated. But try making an appointment. Where 2020's Record Heat Was Felt the Most New data shows that 2020 has effectively tied the hottest year on record, making the last seven years the warmest since the beginning of modern record-keeping. The Great Read The Plucky Newsletter That Isn't Afraid to Name Names At 80, the defiant editor of The Buzz "had it with this whole secrecy thing." So she started a newsletter and slipped it under everyone's door. They Can't Leave the Bay Area Fast Enough As a tech era draws to an end, more workers and companies are packing up. What comes next? Keyan Karimi left his \$4,000-a-month one-bedroom in San Francisco for a new opportunity in Savannah, Ga. Stephen B. Morton for The New York Times Opinion Stephen I. Vladeck Why Trump Can Be Convicted Even as an Ex-President He is the poster child for why such accountability is not just constitutionally permissible but necessary. Linda Greenhouse The Supreme Court After Trump Now that his presidency has disintegrated into mayhem, how are the conservative justices processing his tenure? Gail Collins Trump Swings and Misses Stuart A. Thompson and Charlie Warzel They Used To Post Selfies. Now They're Trying to Reverse the Election. Kara Swisher Food Delivery Is Keeping Uber Alive. Will It Kill Restaurants? Suzanne Nossel Yes, Impeach Trump. But Let's Talk About the Meaning of 'Incitement.' Join Kara Swisher to discuss her interview with Uber's C.E.O. Nicholas Kristof Impeachment Is a Call for 'Moral Clarity' Charles M. Blow What Unity? Looking Ahead to Trump's Trial in the Senate Farhad Manjoo The Capitol Was Just the Start Editors' Picks Devin Oktor Yalkin for The New York Times Mike D Is Cleaning Out the Family Attic A collection of furniture, books and memorabilia that belonged to the rapper's mother will go on view and for sale at Sotheby's this month. Seeking a Sunny Manhattan Studio for \$2,000

bag of words

Here it is with the words in the title and body sorted. Tf-idf can't tell the difference.



heuristics

1. Incrementing a flurry of low-level counters, e.g., number of times each word occurred, number of times an exact phrase matching the query was found.
2. Large number of cheap heuristics that are expected to provide evidence of the quality of the match.

query processing

1. Compile the query into a structure of ISRs.
2. Pass the ISRs to the constraint solver to find matching pages.
3. For each matching page:
 - a. Move the ISRs back to the beginning of the page and scan for hits.
 - b. Calculate a rank value and insertion sort the page into a list of n best.

Example phrase query "quick brown fox" and their 3 posting lists.

Individual search words appear individually many times but there are only two exact phrases. The constraint solver will stop on the first phrase match, then it's up to the ranker to decide what next.

	brown	
	83	
	94	
	170	
quick	179	
	216	
62	227	
69	400	
84		fox
311	417	
421	422	284
430	516	580
559	795	612
619	826	796
794	828	912
952	957	958

	brown	
	83	
quick	94	
	170	
62	179	
69	216	
84	227	
311	400	fox
421	417	
430	422	284
559	422	423
619	516	580
619	516	612
794	795	796
952	826	912
	828	958
	957	

Example phrase query "quick brown fox".

A suggested first step in ranking is to flatten the query, pulling out all the individual ISRs and seeking them to the beginning of the document.

brown			quick	brown	fox
83			62	83	284
94			69	94	423
170			84	170	580
quick	179		311	179	612
62	216		421	216	796
69	227		430	227	912
84	400	fox	559	400	958
311	417	284	619	417	
421	422	423	794	422	
430	516	580	952	516	
559	795	612		795	
619	826	796		826	
794	828	912		828	
952	957	958		957	

Example phrase query "quick brown fox".

From there, you can move the ISRs any way you like to extract data as long as they only go forward.

Word counts:

quick	10
brown	14
fox	7

Two possible strategies:

1. Simply count the words.
2. Look for places where the words occur together in interesting ways.

quick	brown	fox
62	83	284
69	94	423
84	170	580
311	179	612
421	216	796
430	227	912
559	400	958
619	417	
794	422	
952	516	
	795	
	826	
	828	
	957	

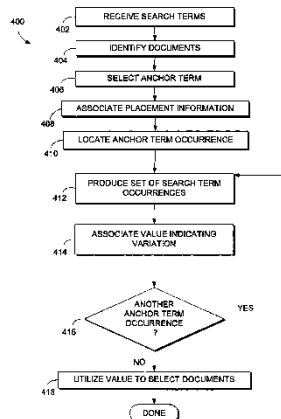
Example phrase query "quick brown fox".

I'm going to show you a simpler strategy I invented at Microsoft.


US007254576B1

(12) United States Patent Hamilton	(10) Patent No.: US 7,254,576 B1 (45) Date of Patent: Aug. 7, 2007
(54) SYSTEM AND METHOD FOR LOCATING AND PRESENTING ELECTRONIC DOCUMENTS TO A USER	
(75) Inventor: Nicole Ashley Hamilton, Redmond, WA (US)	
(73) Assignee: Microsoft Corporation, Redmond, WA (US)	
(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 452 days.	
(21) Appl. No.: 10/847,598	
(22) Filed: May 17, 2004	
(51) Int. Cl. G06F 17/30 (2006.01)	
(52) U.S. CL. 707/6; 707/3; 707/7; 707/10	
(58) Field of Classification Search 707/1, 3, 707/6, 10, 7	
See application file for complete search history.	
(56) References Cited	
U.S. PATENT DOCUMENTS	
5,983,221 A * 11/1999 Chridy 707/3	
6,012,053 A * 1/2000 Pant et al. 707/3	
6,460,029 B1 10/2002 Fries et al. 707/3	

40 Claims, 5 Drawing Sheets



quick	brown	fox
62	83	284
69	94	423
84	170	580
311	179	612
421	216	796
430	227	912
559	400	958
619	417	
794	422	
952	516	
	795	
	826	
	828	
	957	



US007254576B1

(12) **United States Patent**
Hamilton

(16) **Patent No.:** **US 7,254,576 B1**
(45) **Date of Patent:** **Aug. 7, 2007**

(54) **SYSTEM AND METHOD FOR LOCATING
AND PRESENTING ELECTRONIC
DOCUMENTS TO A USER**

6,523,021 B1 2/2003 Monberg et al. 707:2
6,549,897 B1 4/2003 Kalaria et al. 707:5
6,766,320 B1 7/2004 Wang et al. 707:3
2002/0031269 A1* 3/2002 Fukushima 582:228

(75) **Inventor:** **Nicole Ashley Hamilton**, Redmond,
WA (US)

* cited by examiner

(73) **Assignee:** **Microsoft Corporation**, Redmond, WA
(US)

Primary Examiner Uyca Lee
(74) *Attorney, Agent, or Firm*—Shook, Hardy & Bacon, LLP

(*) **Notice:** Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 452 days.

(57) **ABSTRACT**

A system and a method for locating and presenting elec-
tronic documents most-likely of interest to the user. A
plurality of search terms to be located in a set of electronic
document is received. One of the search terms is selected as
the anchor term, and occurrences of the anchor term are
located within the documents. For each located occurrence
of the anchor term, a set of search term occurrences is
selected. These sets include an occurrence of each search
term, and the occurrences are selected by choosing the
search term occurrences that are closest to a desired place-
ment for the search terms. With each set of search terms, the
method associates a value indicating the extent to which the
selected occurrences vary from the desired placement. The
electronic documents are ranked and presented to the user in
accordance with this value. The invention further includes
systems and methods for locating and presenting Web pages
and for searching the Internet.

(21) **Appl. No.:** **10/847,598**

(22) **Filed:** **May 17, 2004**

(51) **Int. Cl.**
G06F 17/30 (2006.01)

(52) **U.S. CL.** 707/6; 707/3; 707/7; 707/10

(58) **Field of Classification Search** 707/1 3,
707/6, 10, 7

See application file for complete search history.

(56) **References Cited**

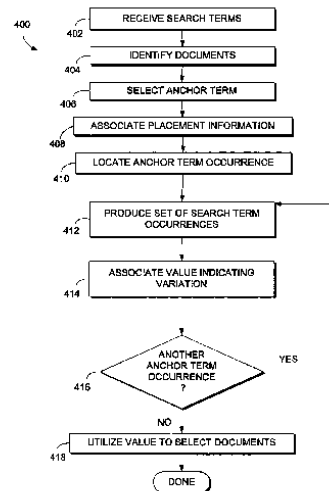
U.S. PATENT DOCUMENTS

5,983,231 A * 11/1999 Christy 702:5
6,012,053 A * 4/2000 Pratt et al. 707:7
6,460,039 B1 10/2002 Fries et al. 707:3

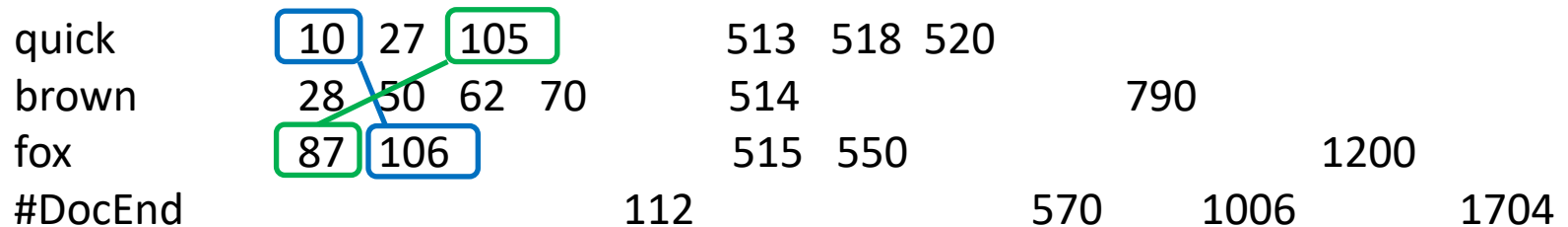
40 Claims, 5 Drawing Sheets

I was very concerned that my ranker
would single-handedly blow us out of
the water on perf and I wanted
something really simple.

It was all integer math and done in a
single pass over each page.



AND'ing streams



quick fox *How many possible combinations? 6*
Can you reach all of them in a single pass, all ISRs only moving forward? No.

Would prefer a technique that allows the ISRs to be moved only in a single pass.

basic strategy

Choose rarest word as an anchor and advance that ISR through each occurrence in the document.

At each occurrence of the rarest word, advance the other ISRs to position them as close as possible to the desired position in the flattened list.

Requires only one stage of look-ahead.

spans

1. If there is at least one occurrence of each query term, that's a "span" which can be further distinguished as ordered, short, an exact phrase, etc.
2. Only the rarest word is guaranteed to a unique occurrence.
3. The other words may be reused.
4. Also count doubles and triples, meaning various combinations of just 2 or 3 of the words in the query, one of which must be the rarest.
5. Parameterized threshold for short vs. long spans, frequent vs. infrequent, etc.
6. Most features are binary and either occur or do not occur unambiguously.

Example phrase query "quick brown fox".

What I did was *pick the rarest word* and *then arrange the other ISRs* to as close as possible to their desired locations in the query relative to each occurrence of the rarest word.

Word counts:

quick	10
brown	14
fox	7

This reduces the number of combinations to be scored in this example from $10 \times 14 \times 7 = 957$ to 7 and can be done a single pass with a single stage of lookahead.

Call each combination a *span*.

quick	brown	fox
62	83	284
69	94	423
84	170	580
311	179	612
421	216	796
430	227	912
559	400	958
619	417	
794	422	
952	516	
	795	
	826	
	828	
	957	

Example phrase query "quick brown fox".

Iterate over the rarest word hits, arranging the other ISRs to as close as they could be to their desired locations in the query. Here are the first 3 *spans*.

brown			brown			brown		
	83			83			83	
quick	94			94			94	
62	170			170		quick	170	
69	179		quick	179		62	179	
84	216	fox	62	216		69	216	
311	227	284	69	227		84	227	
421	400	423	84	400	fox	311	400	fox
430	417	580	311	417	284	421	417	284
559	422	612	421	422	423	430	422	423
619	516	796	430	516	580	559	516	580
794	795	912	559	795	612	619	795	612
952	826	958	619	826	796	794	826	796
	828		794	828	912	952	828	912
	957		952	957	958		957	958

Example phrase query "quick brown fox".

For each combination, decide if it's an *exact phrase* or all the words *in order*, or *close together*, incrementing an associated counter. Shown here, there is one exact phrase but none of the rest are in order or particularly close together.

brown			brown			brown		
	83			83			83	
quick	94			94			94	
62	170			170		quick	170	
69	179		quick	179		62	179	
84	216	fox	62	216		69	216	
311	227	284	69	227		84	227	
421	400	423	84	400	fox	311	400	fox
430	417	580	311	417	284	421	417	284
559	422	612	421	422	423	430	422	423
619	516	796	430	516	580	559	516	580
794	795	912	559	795	612	619	795	612
952	826	958	619	826	796	794	826	796
	828		794	828	912	952	828	912
	957		952	957	958		957	958

Example phrase query "quick brown fox".

	brown	
	83	
quick	94	
62	170	
69	179	
84	216	
311	227	fox
421	400	284
430	417	423
559	422	580
619	516	612
794	795	796
952	826	912
	828	958
	957	

	brown	
	83	
quick	94	
62	170	
69	179	
84	216	
311	227	
421	400	fox
430	417	284
559	422	423
619	516	580
794	795	612
952	826	796
	828	912
	958	
	957	

	brown	
	83	
	94	
	170	
quick	179	
62	216	
69	227	
84	400	
311	417	
421	422	fox
430	516	284
559	795	423
619	826	580
794	828	612
952	957	796
		912
		958

Example phrase query "quick brown fox".

This last one is *close together* (say, less than 10 words apart) and *in order* but it's *not an exact phrase*.

	brown	
	83	
	94	
	170	
quick	179	
62	216	
69	227	
84	400	
311	417	fox
		284
421	422	423
430	516	580
559	795	612
619	826	796
794	828	912
952	957	958

Example phrase query "quick brown fox".

Set some thresholds and accumulate some counts, which can be scored at the end.

Thresholds	Values
Max to be short	10
Min to be frequent	?
Min to be most	?
Min to be near the top	?

Heuristic	Count	Weight	Score
Number of short spans	1	5	5
Number of in order spans	1	2	2
Exact phrases	2	10	20
Number of spans near the top	?	?	?
All word are frequent	?	?	?
Most words are frequent	?	?	?
Some words are frequent	?	?	?

Total dynamic rank score ?

brown		
	83	
	94	
	170	
quick	179	
62	216	
69	227	
84	400	
311	417	fox
421	422	284
430	516	423
559	795	580
619	826	612
794	828	796
952	957	912
958		

Example phrase query "quick brown fox".

One reason to decorate words as *anchor*, *URL*, *title* or *body* is so they can be separated in to separate *streams* (separate sets of ISRs) and *scored separately* with the same algorithm but different scoring parameters.

<i>Stream</i>	<i>Weight</i>	<i>Score</i>
Anchor	?	?
URL	?	?
Title	?	?
Body	?	?
<i>Total dynamic rank score</i>		?

Each of the streams can start at location 1 relative to the start of the document. The document length is the length of the longest stream.

Not all streams may contain all the words.

	brown	
	83	
	94	
	170	
quick	179	
62	216	
69	227	
84	400	
311	417	fox
421	422	284
430	516	423
559	795	580
619	826	612
794	828	912
952	957	958

Example phrase query "quick brown fox".

We do a similar heuristic calculation of the *static rank*, the quality of the page *independent of the query*.

<i>Heuristic</i>	<i>Weight</i>	<i>Score</i>
Short title	?	?
Nice document length	?	?
Short URL	?	?
Lots of anchor text	?	?
edu/gov/com/etc domain	?	?
PageRank if known	?	?
<i>Total static rank score</i>		?

	brown	
	83	
	94	
	170	
quick	179	
62	216	
69	227	
84	400	
311	417	fox
421	422	284
430	516	423
559	795	580
619	826	612
794	828	796
952	957	912
		958

Example phrase query "quick brown fox".

Combine static and dynamic rank to get a final score.

<i>Component</i>	<i>Weight</i>	<i>Score</i>
Static rank	?	?
Dynamic rank	?	?
<i>Total rank score</i>		?

The result is a *linear combination* of features. We're just adding them up.

You should be able to achieve reasonably good results with a reasonably small number of heuristics and simple *hand-tuning* starting from some rough guesses at, e.g., the relative importance of an exact phrase versus lots of individual hits.

	brown	
	83	
	94	
	170	
quick	179	
	62	216
	69	227
	84	400
	311	417
	421	422
	430	516
	559	795
	619	826
	794	828
	952	912
	957	958
		fox
		284
		423
		580
		612
		796

The actual score is calculated as a linear combination of features, which may be thought of as:

$$R = \sum C_i(Q) * A_i(P, F_i) * S_i(F_i)$$

Where:

R = Overall Rank

Q = The Query and its characteristics, e.g., the number of rare vs. common words in the query.

P = The Page and its characteristics, e.g., the number of words in the URL or title.

F_i = An arbitrary feature observation, e.g., an exact phrase in the title, or a raw value, e.g., raw PageRank.

S_i = Scaling for feature F_i from the raw number space of the feature into a nominal 0.0 .. 1.0 range.

A_i = Attribute scaling for feature F_i, possibly dependent on the characteristics of the page P.

C_i = Coefficient for feature F_i, depending on the characteristics of the query Q.

F_i = E_i(Q, P, T)

steps to ranking

1. Decide what information to collect and how to measure it.
2. Decide how to measure the quality of the result.
3. Pick a method for scoring a page based on the inputs.
4. Tune the system by testing it on sample queries and adjusting parameters, collecting more information or changing the scoring algorithm.

measuring quality

At Microsoft, initially, I just ran queries and eyeballed the results and fiddled with the parameters.

Results were surprisingly good.

measuring quality

We were also collecting labeled pages. We scraped queries and results from several engines and then paid people to rate results on a 0 to 5 scale.

- 5 Definitive result. Unlikely any other page could be a better result for this query, e.g., whitehouse.gov for "whitehouse".
- :
- 0 Completely deleterious result, e.g., porno, spam, phishing.

This is similar to what Google is doing with their [Page Quality Rating Guidelines](#), creating specificity in the ratings.

Altogether, I think we had about 50K queries and about 150K labeled results.

measuring quality

Next came a tool allowing us to do side-by-side comparison of results with current best parameters in a frame on the left and with a set you could tweak on the right.

If the pages had been labeled, it reported the quality and attempted to score the resulting set.

We ran a competition to see who could come up with the best settings.

measuring quality

Finally, we added a tuner that could adjust the parameters mechanically by gradient descent: Tweak an individual parameter rerun all the searches and measure whether the results got better (more highly ranked pages for each query.)

But we had lots of problems in the methodology of what to do with unlabeled pages.

measuring quality

Initially unlabeled pages were assumed to be “average”. Later, a “promising proximity” heuristic was added to the tuner bump the estimate for unlabeled pages if the search words were found close together.

The effect was to tune my complex ranker to behave like the tuner’s naïve ranker.

Better strategy would have been to only tune based on labeled results, discarding any unlabeled results or construct an index that only contained labeled pages.

generating labeled results

You can't afford to pay people to label results but you might assume that Google is pretty good, so you might simply try to get pages in the same order as Google.

Your ranker won't be as good as Google's, so it might not be able to discern the difference between the top results on the first page, but perhaps it could distinguish between results on the first page and results on the tenth page.

strategy

1. Create a list of popular short queries.
2. Scrape Google for first 10 pages of results and label them.
3. Create an index of only those pages.
4. Make some initial estimates of the weights of your heuristics.
5. While you're still making progress:
 - a. Run each query and count how many you got right.
 - b. Iterate over each heuristic, adjusting it up or down a little to see if there's any improvement.
 - c. Pick the best as a new starting point.

remarks

It is much easier to tune weights for binary features than for those encoding a range of “goodness”.

It is difficult to estimate the improvement you’ll get from a new feature without actually building it. A lot of it is intuition.

remarks

Think about what “blindspot” is created by not having the new proposed features.

1. When would the ranker not be able to tell the difference between two pages it should be able to distinguish?
2. How egregious would that be and how often would it occur?

What is your ranker getting wrong right now and would a simple heuristic fix it?

This kind of thinking led us to add features like “exact match” for the title.

remarks

Mechanically tuned parameter weights can be counter-intuitive if some of your heuristics overlap.

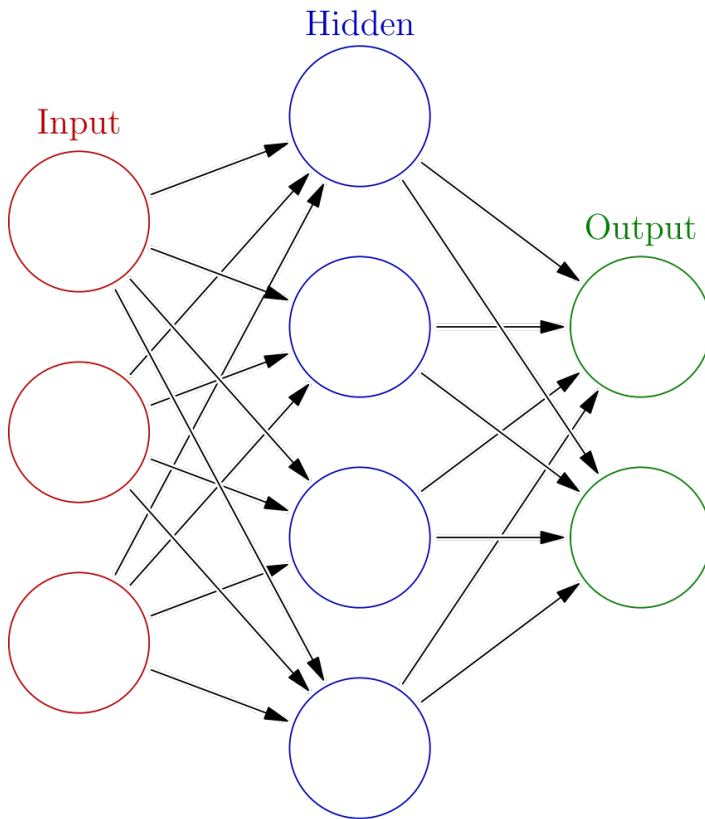
You can get a negative weight for something that should be good because it overlaps with something better.

how to do better

To do better than heuristics, you will probably need a neural network.

Example phrase query "quick brown fox".

A machine learning strategy would be to simply collect all the lists and pass everything to a neural network, which must be trained. Beyond the scope here.



quick	brown	fox
62	83	284
69	94	423
84	170	580
311	179	612
421	216	796
430	227	912
559	400	958
619	417	
794	422	
952	516	
	795	
	826	
	828	
	957	