Dublin City University

School of Computing

CA4009: Search Technologies

Section 5: Web Retrieval

Gareth Jones

October 2019

Introduction

For many people Web Retrieval represents their most frequent experience of an IR application.

The emergence of the web marked a dramatic development in the scope and importance of IR technologies.

The user base of IR technologies expanded from focusing on trained professional information seekers, e.g. librarians, ...

to the general public who almost exclusively have no training in the use of IR systems or underlying IR technologies.

Introduction

Effective Web Retrieval raises many challenges.

To meet these challenges, companies offering web search engines seek to exploit all available information to provide the most effective possible search tools.

We will look first at the challenges of Web Retrieval and the differences between it and the classic IR setting that we have looked at so far.

We will then introduce the methods used in Web search engines to satisfy user information needs most effectively.

Introduction

Users of web search engines typically look for the following features:

- interactive processing of search queries (online).
- effective ranking of relevant content - particularly at the top of the retrieved list!
- no requirement to click “next page” button - motivates the need for high precision at the top end of the ranked list.
  - since most users will not (or don’t know how to) look at the second page of results.
- very rapid response time is essential.

Document ranking based only on scoring using $ms(j)$ with term weighting does not work well in the Web environment. Why?

Introduction

Classic IR methods generally assume:

Document collection to be searched is:

- professional or carefully authored and edited, e.g. newspapers, scientific papers, ...
- of uniform good quality, and
- typically of a consistent style for a particular document collection.

Search requests:

- best effort statement of information need.
- long (5+ words)
- words carefully chosen to seek to describe information need.

Introduction

In practice, for Web Search:

Documents:

- variety of content and style (genre, quality, languages, ...), length, format
- varied quality of authorship (typos, poor choice of vocabulary)
- deliberate misinformation (spam)
- (near) duplicates, the same or very similar content published multiple times, including mirror sites.

Search requests:

- written quickly without attention to detail.
- typically very short 2-3 words - poor description of information need.

Introduction

Overall the relevance of an individual document returned in response to a search request depends on the information need of the user, i.e. is the document relevant to this user and their current information need?

Particularly for topics described in many documents, search engines using only document matching IR methods (as described so far in this module) may return a very large number of documents with high matching scores.

It is often impossible to distinguish between the quality/reliability of these documents based on simple statistical analysis of their contents using term weighting.

Thus it is difficult to ensure that reliable and useful documents appear nearer the top of the list, than other documents which match the query, but are unreliable or not useful.

Introduction

For effective Web Search, we need IR methods that give high precision for very large document collections.

Essentially we need measures of document importance which go beyond the matching score between the query and the document contents.

From the software engineering perspective for commercial Web search engines: we need:

- IR systems able to index billions of documents, and to be able to update these indexes efficiently.
- Also to be able to provide rapid response to millions of search requests per day.
- While of course providing accurate retrieval!

Introduction

There have been many innovations in improving ranking of Web content.

These techniques augment the query-document matching score using a variety of factors which can include:

- link structure of the web
- ”spamminess” - likelihood of the content being spam or poor quality
- content diversity
- user interaction - logging queries and clicks
- subjective quality of pages

These and other features (generally referred to as *signals*) can be combined in a technique referred to as *learning-to-rank*.

Exploiting linked structure of the Web

For un-linked or “flat” document collections document importance may be based on metadata, e.g.

- source of the document
- the author of the document.

If the document is published by a formal source such as a national newspaper, it may be considered more reliable, than an informal leaflet from a previously unknown grouping with a particular viewpoint.

Exploiting linked structure of the Web

The *hypertext* structure of the Web adds considerable additional information about a Web page to the actual content of each individual page.

For example, the link between two pages tells us something about the relationship between the source and target pages.

In addition, the text in the anchor tags from the source to target page can be very valuable in describing the page (as we saw in earlier lectures).

- Link anchor text usually indicates the topic of the page pointed to.
- These link anchors can sometimes provide better descriptions of the page pointed to than the page itself! (e.g. when pointing to images or sound files.)

Exploiting linked structure of the Web

Inter-connected or hyperlinked collections such as the Web can provide measures by which expected user interest in each Web page can be estimated.

One example of such a measure is the *PageRank* score which forms a part of the *Google* search engine.

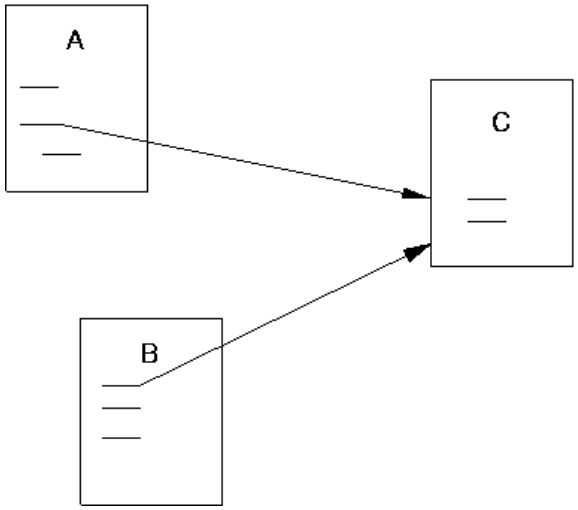
PageRank takes advantage of the link structure of the web to produce an approximate global importance score for each page based only on the link structure of the Web.

At any point in time, each page on the Web has a PageRank score associated with it.

The PageRank score is independent of the actual content of the page.

Link Structure of the Web - The Background to PageRank

Web pages generally have a number forward links (*outedges*) and back links (*inedges*).



A and B are back links of C. C is a forward link of A and B.

We can never know all the back links of a page, but we can know all its forward links (at least those that were present when it was last analysed by a web spider!).

Link Structure of the Web - The Background to PageRank

If Page A links to Page C, then Page A is indicating that Page C is an important page with respect to Page A.

PageRank also factors in the importance (as measured by their PageRank score) of the web pages with links pointing to a page.

- If a page has important links pointing to it, it has a high PageRank value, and its links to other pages also become important.

If Page A is deemed important, then its link to Page C is deemed important.

The actual text associated with a link is irrelevant in computing the PageRank score of a page.

Link Structure of the Web - The Background to PageRank

Pages vary greatly in the number of back links.

Generally a page with many back links is more “important” than one with only a few. But this is not always true:

- a single link from a site with high authority may indicate a very important page, e.g. a landmark node
- a page having many in-links from obscure sites may be less important.

A web page will have a high PageRank score if the sum of the contributions from the PageRanks of its back links is high.

- the pages wth the highest PageRank score will have a large number of high scoring back links.

Definition of PageRank

For a web page j. Let,

F_j = set of web pages j points to (outedges)

B_j = set of web pages that point to j (inedges)

$N_j = |F_j|$ = number of links from j

$$PR(j) = (1-d) + d \sum_{v \in B_j} \frac{PR(v)}{N_v} \tag{1}$$

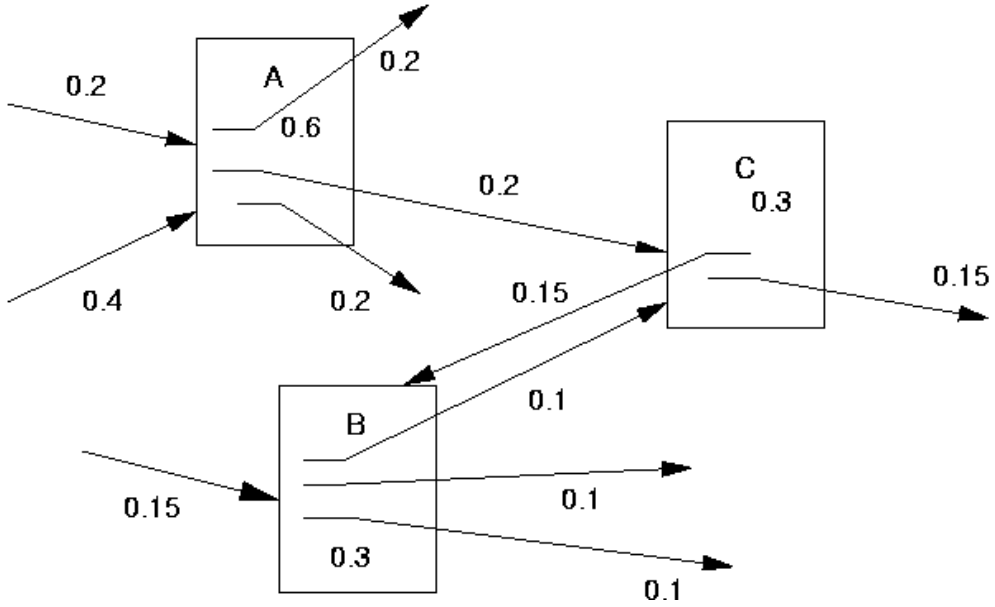
$PR(v)$ is the PageRank of page v point to page j.

N_v is the number of outlines from page v.

d is typically set to 0.15

Definition of PageRank

Thus, for a Page v pointing to the Page j, its contribution to $PR(j)$ depends on the number of out links from Page v.



(assuming $d = 1$)

Page A has 3 outlinks, its PageRank is 0.6, since there are 3 outlinks is contributes 0.2 to the PageRank value of each page it points to.

Random Surfer Model

PageRank is the likelihood that a “random surfer” visits a page.

The random surfer starts at a random page and keeps clicking on links never going back.

Eventually the surfer gets bored and starts at another random page.

d can be interpreted as the probability that the surfer will move to a randomly chosen page linked to the current page.

$(1 - d)$ is the probability at each page that the random surfer will get bored and move to a randomly chosen new web page not linked to the current page.

Random Surfer Model

d also indicates the influence of the parents, grandparents, etc on the value of $PR(j)$ - see Equation (1).

As the value of d increases the influence of previous pages is increased.

In the example where $d = 1$, it is assumed that the surfer will always follow a random link from the current web page.

If $d = 0$, the surfer never follows a link from the current web page and all pages have $PR(j) = 1.0$.

If $PR(j) = 1.0$ for all j, what does PageRank tell us about the importance of each page?

Definition of PageRank

Computation of $PR(j)$ requires the value of $PR(v)$ of each page pointing to it; and hence the PR of every page pointing to every page!

So to start with values of $PR(j)$ have to be assumed for each page j.

Using these assumed starting values we can compute a revised value for the PageRank of each page using the equation for calculating $PR(j)$.

Assuming an initial value of $PR(j) = 0$ for all pages and $d = 0.15$.

The revised value of $PR(j) = 0.85 + 0.15 \times 0 = 0.85$ for all pages.

The calculation must be repeated until the values of $PR(j)$ do not change after each recalculation, i.e. it reaches *convergence*.

Definition of PageRank

The calculation becomes more interesting after the first iteration since the link structure becomes part of the calculation and the pages start to have unique $PR(j)$ values.

Provided that d is less than 1, convergence will occur. (The value 0.15 is taken from the original Google research papers.)

Whatever value we start with, we will end up with the same final values or *limiting* values.

In practice the calculation can be stopped when the value of $PR(j)$ changes by only a very small amount between iterations, say less than 0.0000000001.

Definition of PageRank

This calculation of the PageRank value assumes that the set of documents is static.

But the web is changing all the time!

What does this mean in terms of calculating PageRank values?

Searching with PageRank

Searching using PageRank without a query would simply return documents ranked by their PageRank score.

Documents returned for a simple one word request (and assuming no term weighting) could be ranked by their PageRank score (essentially a subset of the full document collection list).

A combined ranking score $cms(j)$ for document j incorporating a traditional $ms(j)$ and PageRank can be computed as follows:

$$cms(j) = ms(j) \times PR(j)$$

Thus a document's ranking may improve either through increasing the content-based scoring $ms(j)$ or by receiving a better PageRank $PR(j)$.

Combining Content-Based and PageRank Scores

The value of the content-based $ms(j)$ can be increased by modifying the contents of the page.

The maximum score for a document will be achieved by a document containing all, and only, the words of a search query with high frequency.

In other words, the content-based measure has a maximum value.

PageRank has no such limitation.

Thus, the only way to achieve a high ranking for your page relative to your competitors for a popular search query with many similar looking pages available, is for your page to have a high PageRank score!

Spam Filtering

Web spam is broadly speaking *content that will never be deemed relevant to a query*.

Defining spam is not as straightforward as might be assumed.

For our purposes, we take a broad definition of **spam** as **all content that the user would not wish to see** or **which is designed to influence the behaviour of a search engine in a detrimental way**.

As such, spam pages may contain content which is knowingly deceptive or harmful, either to the user or a search engine, or low-quality ”junk” which the user will never be interested to access.

Spam Filtering

Spam filtering services are intended to identify this type of content during the indexing phase when it is being ingested into the search engine.

Malicious spam generally attempts to promote the rank of a document by one of two methods:

- Self-promotion: most commonly via *keyword stuffing*: adding content to a page to increase counts of words in the page or adding words that are not in visible content of the page.
- Mutual promotion: typically using *link farms*.

Link Farms

Many popular and important pages on the Web have large numbers of onlinks from other pages which recommend them.

Liink farms are plausible looking web pages which attempt to manipulate PageRank type measures by inflating link counts of their target pages.

Link farms exist merely to give very high back link counts to their target pages with the intention of improving their PageRank value and search rank.

Search engines providers such as *Googl*e seek to automatically identify and block the influence of these artificial links.

How might these be identified?

Link Farms

Once it is clear to the operator of a link farm that it has been detected by the search engine providers and is no longer effective, they have the option to modify it in some way, e.g. change page or site structure, content, location, etc.

It may then go undetected again until the search engines operators change their link farm detection methods.

At which point the operators of the link farm may change something again.

And so the process goes on and on.

This sort of game between search engine providers and those who seek to influence search engine behaviour in an adverse way is referred to as *adversarial search*.

Spam Classification

Collections of known spam documents can be used to train statistical spam classifiers.

These models assign a ”spamminess” score for each page. This spamminess score can be used in various ways, e.g.

- A threshold can be applied to classify a page as either spam or not spam.
  Pages classified as spam are excluded from the search engine results.
- The spamminess score can be used to adjust the rank of the retrieved web pages.
  The more likely a web page is to be spam, the lower it will appear in the ranked list, regardless of how well its contents match the search query.

Query Sense Identification

Queries can be ambiguous, that is, it is not clear what topic or aspect of a topic the user is interested in.

This is particularly common when the query is very short.

For the query ”cancer”, we don’t know whether the searcher is interested in symptoms, diagnosis, treatment, prognosis, research, state funding, etc.

How should a search engine respond to this situation?

Query Sense Identification

Special cases:

“apples” $\neq$ ”apple” - ”apple” as a query is often found to relate the company, whereas ”apples” is found to relate to the fruit - we shouldn’t apply stemming here

“bio” $=$ ‘biography” - ”bio” is a common abbreviation of ”biography”, in this context we can apply what might often be considered to be overstemming.

- We could have context-sensitive rules for words relating to apples/apples, bio/biography. and special cases.

For logged in individuals, the IR system can monitor their querying and browsing behaviour to determine likely intent of future queries.

Diversity

A query may be ambiguous because the user has not taken the trouble to specify what they are interested in in detail, or because they may have a genuine desire for broad coverage of the topics covered by the query.

A sensible response to ambiguous queries is to diversify the retrieved results (i.e. to ensure that the returned results include multiple interpretations of the query), in the hope that at least one of them will satisfy the user.

E.g., for a product search, e.g. “bmw” , “ford”, the output could include:

- manufacturer’s website
- blog
- place to buy
- price comparison website

Diversity Strategies

implicit diversity: compare retrieved documents to each other, select documents for display which are most likely to be relevant to the query while being most dissimilar from each other.

explicit diversity: account for different aspects of the query. Identify interpretations for the query, and classify retrieved documents for each of the sub-queries.

Display the highest ranked documents for each interpretation of the query in the Search Engine Results Page (SERP).

If the user selects a document from the list, apply relevance feedback:to

- find more documents like this,
- re-rank to place documents similar to the clicked one higher in the list.

Web Diversity

In the case of Web retrieval, a more basic problem often needs to be addressed for many queries – filtering of repeated copies of materials:

- Much content is repeated more than once on the Web.
  - mirrors and copies of files in multiple places.
- Sometimes the top 10 list of a web search engine would be dominated by identical content indexed from multiple different websites.
- Automatic filtering can be used to remove these from display in the ranked list - which effectively gives a more diversified result list.

The potential presence of these multiple copies is generally hidden from the user, since they are filtered out by search engines to produce diverse output.

Exploiting User Behaviour with Click Models

- Billions of queries are submitted to Web search engines each day.
- These queries, with the timestamp of their submission, the list of documents retrieved, and whether each document is clicked by the searcher, can be captured in user Web *logs*.
- These web logs can be used as implicit user relevance feedback of retrieved documents.
- This feedback will be noisy (see earlier discussion of use of clicks in relevance feedback), but the volume of data from multiple submissions of the same query by multiple searchers can be used to smooth out much of the noise.

Exploiting User Behaviour with Click Models

- The “query-clicked document” relationship indicates potentially relevant documents for each query.
- Documents clicked in response to different entries of a query can be clustered based on the similarity of the contents of the documents.
- These clustered documents are likely to correspond to alternative interpretations of (probably) ambiguous queries.
- These clusters can be used to promote diversity in subsequent retrieval results for this query by ensuring a representative document from each cluster appears in the ranked list.

Exploiting User Behaviour with Click Models

Click information can be used as an assessment of the user experience in using the search engine, but also potentially to improve the effectiveness of the search engine.

- How many clicks are on the SERP for this query? Does it look like the searcher might be confused?
- What is the average rank of the clicks? Lower rank probably means poorer rank of relevant content.
- Do different ranking algorithms produce more clicks nearer the topic of the ranked list?
  - Clicks nearer the top of the list generally mean less work (reading snippets and thinking) for the user, before finding relevant results.

Exploiting User Behaviour with Click Models

- *Click models* attempt to model user search behaviour based on web search logs and to predict future activities.
- A click model can estimate document relevance based on search behaviour.
- This relevance value indicates the degree of correlation between the query and likely relevance of each document.
- Inferred relevance values for retrieval for a query can be used as input to a ranking model for search if this query is entered again by another user.

Content “freshness”

“freshness” refers to how recently content was created or changed.

Should a web search engine return the most recently created or edited pages, or the most stable which have proved popular over a longer period?

“President Trump’s speech about north Korea last week at the White House”

“north Korea” - is likely to be the most distinguishing term - how might we work this out?

Content “freshness”

Google has found that simply biasing in favour of new pages degrades search results much of the time.

Users don’t always want the most recent page which matches their query.

Google “Query Deserves Freshness (QDF)”: tries to determine when users want new information and when they don’t.

Try to identify “hot” news sites or blogs to see when people are actively writing about a topic.

If Google determines that this topic is hot, related queries should receive “fresh” content in response to them.

Can also look at the live or current query log.

Content “freshness”

Topics which appear in many current queries are “‘hot”, and again output can be biased to them.

Google: a city blackout will be written about in 15 minutes and queried for in 2 seconds!

So we need to make content on hot topics available for search as quickly as possible!

How can we do this? Source authority - just give the user matching content from trusted news feeds.

Rapid indexing to enable search of content shortly after entry is particularly a challenge for microblog services such as *Twitter*.

Search Engine Optimization (SEO)

Search Engine Optimization (SEO) refers to the process of adjusting the contents of a web page or website to maxmize its ranking in the output of a search engine.

Suitable adjustments to promote rank depend on the rules used by the search engine to determine document ranking.

Spamdexing or *black hat SEO* refers to the process of adjusting contents of a web page or website to promote rank in contravention of the search engine guidelines for webmasters.

Search engines may eliminate or down rank web pages and websites suspected of spamdexing.

Spamdexing

Factors which are considered spamdexing include:

- Keyword stuffing: as discussed previously.
- Link schemes: link farms as discussed previously.
- Cloking: presenting different contents to search engine spiders and user browsers.
- Deliberate creation of duplicate content: copying excessive amounts of material between pages on a website to increase its rank.

Google *Penalty* aims to decrease the rank of undesirable websites.

A specific example is the Google *Penguin* algorithm which aims to decrease rank of websites which violate Google's webmaster guidelines.

Google Panda

The *Panda* algorithm aims to lower the rank of “low-quality sites” or “‘thin sites” and place “higher-quality” sites near the top of the search results.

Introduction of the Panda algorithm by Google as part of their document ranking function improved the ranks of news websites and social networks, and demoted the ranks of sites with large amounts of advertising.

Google defined 23 bullet points on its blog answering the question “What counts as a high-quality site?” These are intended to help webmasters “step into Google’s mindset.”

Google Panda

To create Panda scores:

- Human judges rate the quality of thousands of websites based on measures of quality, including design, trustworthiness, speed and whether or not they would return to the website.
- Google use a machine-learning algorithm to look for similarities between websites that the judges found to be high quality and low quality.
- These characteristics can then be used to classify websites and incorporate this classification score into the overall ranking of the website.

Google Panda

Introducing Panda decreased the importance of the PageRank algorithm in determining the page score.

Panda affects the ranking of an entire website or a specific section of a website, rather than just an individual page.

Google have also introduced a penalty for “over-optimized” websites:

The aim is to give websites which have great content an increased chance of being ranked above websites which contain content which is not as good, but which have better SEO.

Content-Based Scoring Measures

The query-document matching score introduced in the Text Retrieval section, $ms(j)$ based on the query and the contents of the web page is a simple single measure which takes all words on a page as equally important.

But in practice, different words have different levels of significance to a page. To take account of this terms:

- appearing in title sections may be upweighted.
- may be upweighted depending on font size or style (e.g. bold or italic).
- in start sections of documents and/or paragraphs may be upweighted.
- terms proximity may be taken into account - upweighted if they are close together and therefore likely to be related.
- not displayed (meta information) may be handled differently, e.g. downweighted or ignored.

Content-Based Scoring Measures

Anchor text can be used to supplement the text indexed in the search engine for the web page pointed to.

Anchors pointing to a web page can be combined into a single document unit.

All anchortext pointing to a document may be simply be concatenated or a more complex fusion mechanism may be adopted.

Anchor documents can either be combined with the text of the document that they are pointing to, or be indexed as a separate field for each document.

Indexed anchortext documents can be scored against a query, and their matching score combined with the matching score of the web page itself.

Learning to Rank

The factors or *signals* discussed in this section and others can be combined to give a composite score for each document to determine the ordering of the ranked retrieved list.

A simple way to do this is a weighted linear sum of components:

$$\begin{aligned} ms(j) &= a1 \times factor1 + a2 \times factor2 + a3 \times factor3 + \ldots \\ &\quad + aN \times factorN \end{aligned}$$

$a1$, $a2$, etc are scalar constants.

Setting these values manually is impractical, particularly when there are many parameters.

Learning to Rank uses machine learning methods to automatically tune the scalar constants, essentially to train the ranking model.

Learning to Rank

As well as the features introduced in this section, other factors which are typically used for Learning to Rank in Web Retrieval include:

- No. of inlinks to the page
- No. of outlinks from the page
- No. of reciprocal links between this page and another one
- URLLength of the page - shorter URLs associated with pages nearer a root and likely to me more important
- URLType of the page - root, sub root, path, file
- No of matching query terms in the page

Learning to Rank

Training the IR system:

1. Collect a training set of queries with corresponding relevance data using a pooling procedure (like developing an IR test collection).
2. For each query obtain an initial document ranking, e.g using ranking using BM25.
3. For the top K documents in the initial ranking, work out the values of the features to be used, e.g. spamminess score, click model score, etc.
4. Apply a machine learning method to train optimal parameter weights to maximise the ranking performance of the system. Parameters tuned to maximise a selected retrieval evaluation metric, e.g. mean average precision (MAP).

Learning to Rank

The relevant pages for each training query are typically derived from click data - using the same query from multiple searchers (as discussed earlier).

If sufficient users click on a document when entering query, it is assumed to be relevant.

Learning to Rank

Using the IR system:

1. For a new query, create a ranked list using a standard method such as BM25.
2. Extract ranking features for each of the top K ranked documents from the initial ranked list.
3. Apply the learned model on the extracted features, and sort in descending order to relevance scores calculated using the learned model.

A/B Testing for Web Search

- Laboratory-based evaluation of IR systems using test collections to calculate precision and recall is very effective for the development of new IR techniques and extensions to existing ones.
- However, once an IR system is deployed, .e.g. it is available as a working online search engine, there are other evaluation opportunities available which can take advantage of the actions of its users.
- A/B testing of operational online systems enables developers to evaluate new ideas by getting very rapid feedback on their effectiveness.

A/B Testing for Web Search

- Suppose that we have an existing online search engine and we wish to try out a new idea which might improve ranking of relevant documents, e.g. a new text-based ranking algorithm or introduction of a new “signal” into the ranking algorithm.
- Let the current system be A and the system integrating the new ranking method be B.
- The search engine provider can perform a live online experiment to evaluate the user response to the new ranking method.
- At some point both systems are set to run live.
- They should have the same interface so that the user will interact with each one in the same way, but different backend ranking algorithms.

A/B Testing for Web Search

- System A or B is then assigned to each searcher at random when they access the user interface and submit a request to the search engine.
- The responses of the user to the returned ranked list are then monitored and logged.
- The search engine provider is then able to evaluate features such as: how many returned documents are clicked, the rank of the clicked items, and the time between clicks.
- The number of users of an online search engine will typically mean that a very large amount of log data is captured very quickly.

A/B Testing for Web Search

The search engine provider is then able to evaluate any change in user behaviour arising from the change in ranking algorithm, e.g.

- Is the average rank of clicked items higher or lower?
- Is the searcher looking at more snippets before clicking?

which would indicate that they are doing more work and that relevant items are probably positioned further down the ranked list, i.e. the new algorithm is less good than the existing one.

A/B Testing for Web Search

Note: The outcome of the evaluation may not follow the expectations of the search engine provider. There have been multiple examples of A/B testing producing counterintuitive or unexpected results

- e.g. users were found to engage more with a search engine to reformulate queries to make them more effective, for the version of the search engine where the ranking of relevant documents was found to have decreased.