Applying the User-over-Ranking Hypothesis to Query Formulation

Matthias Hagen   Benno Stein

Bauhaus-Universität Weimar
matthias.hagen@uni-weimar.de

ICTIR 2011
Bertinoro, Italy
September 14, 2011

What is the User-over-Ranking hypothesis?

The User-over-Ranking Hypothesis

[Stein and Hagen, ECIR 2011]

Queries returning as many results as the user can consider increase retrieval performance.

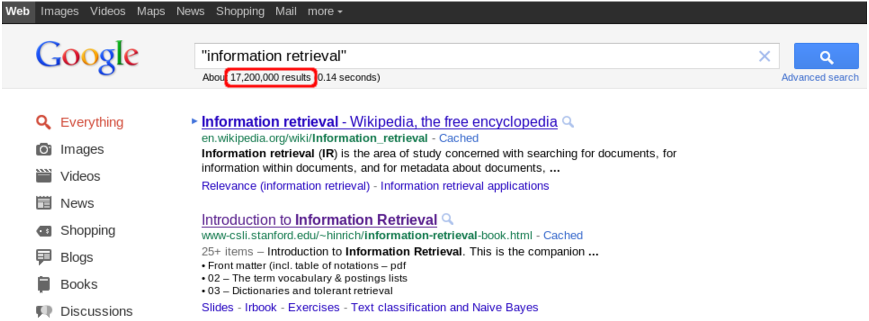
The User-over-Ranking Hypothesis

[Stein and Hagen, ECIR 2011]

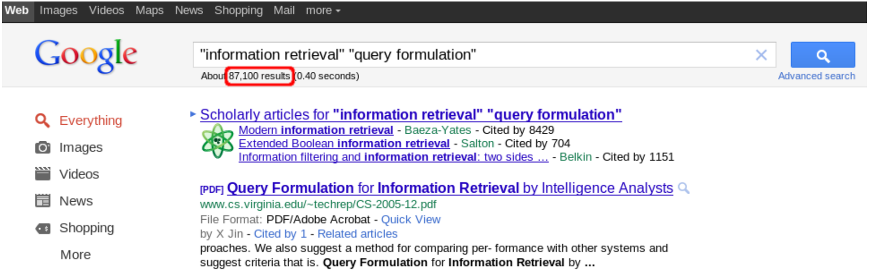
Queries returning as many results as the user can consider increase retrieval performance.

Fine print: If ranking works: great!
Use case is not some query like `ebay`.
But more involved information needs,
automatic systems, etc.

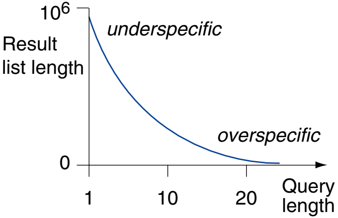
Assumption 1: More keywords = more specific



Assumption 1: More keywords = more specific

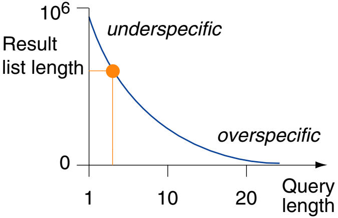


Assumption 1: More keywords = more specific



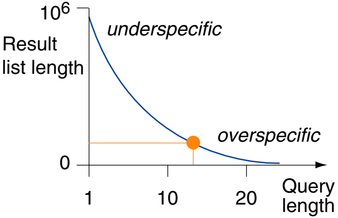
Specificity of Queries

Assumption 2: User can arbitrarily specify information need



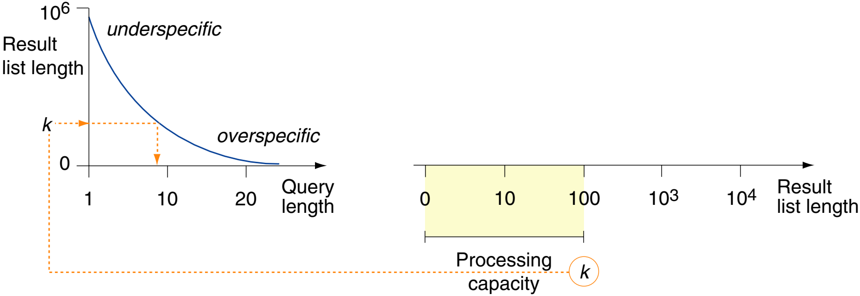
Specificity of Queries

Assumption 2: User can arbitrarily specify information need



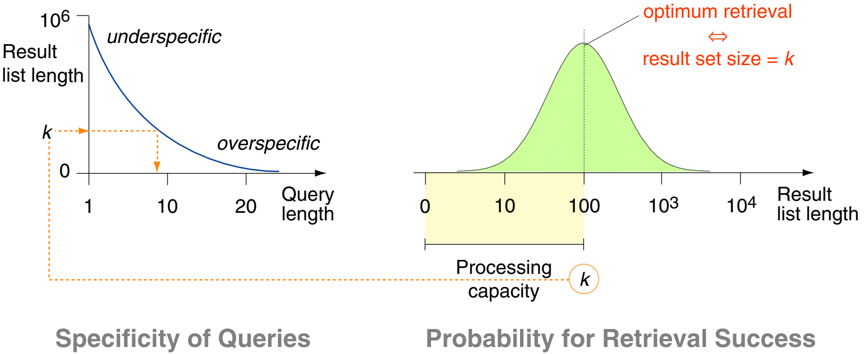
Specificity of Queries

Assumption 3: User can consider about k results.



Specificity of Queries

Hypothesis: Specificity matches k = Optimum retrieval



What is this hypothesis good for?

Query Formulation

Scenario

- Given a set W of keywords
- Find a good query $Q \subseteq W$

Query Formulation

Scenario

- Given a set W of keywords
- Find a good query $Q \subseteq W$

Previous approach [Lee et al., CIKM 2009]

Learnt ranking function identifies the m best keywords from W.

Based on:

- Known relevant documents
- Unrestricted index access
- Manually tuned m for each set W

Consider for instance . . . Known-Item Finding

Scenario

- User accessed a document
- But did not store it

Making the Most of a Web Search Session

Benno Stein and Matthias Hagen
Bauhaus-Universität Weimar
<first name>.<last name>@uni-weimar.de

***Abstract*—We tackle problems related to Web query formulation: given the set of keywords from a search session, 1) we find a maximum promising Web query, and, 2) we construct a family of promising Web queries covering all keywords. A query is promising if it fulfills user-defined constraints on the number of returned hits. We assume a real-world setting where the user is not given direct access to a search engine's index, i.e., querying is possible only through an interface. The goal to be optimized is the overall number of submitted Web queries.**

For both problems we develop search strategies based on co-occurrence probabilities. The achieved performance gain is substantial: compared to the uninformed baselines without co-occurrence probabilities the expected savings are up to 50% in the number of submitted queries, index accesses, and runtime.

***Keywords*-Web Search Session, Query Formulation, Query Cost Optimization, Maximum Query, Query Cover**

I. Introduction

The interactions between Web search users and search engines follow a classic scheme. The user comes up with a set of (in her opinion) appropriate keywords for a given information need. She submits a query containing some of these keywords and gets back a ranked result list. If the user does not find a match for her information need among the first results, she will hardly browse all the items but submit different queries based on her keywords until she is satisfied or decides to give up. This process forms a *search session*—the set of consecutive Web queries a user submits to a search engine in order to satisfy a given information need.

Experience shows that in many cases a user's first try is answered reasonably well, i.e., the first query brings up an appropriate result. Such one-query-sessions are not the focus of this paper; we consider longer sessions where the user did not succeed with her first query. Search engines provide different means to support unsuccessful users, e.g., query expansion for queries returning lots of hits or spelling correction for queries returning no hits due to typos. In this paper we present two other approaches having a more combinatorial flavor, while being easily combinable with existing technology:

1) The *maximum query* for a given set of keywords, i.e., a query containing as many of the keywords as possible, while returning a reasonable number of results.
2) The *query cover* for a given set of keywords, i.e., a family of queries (each returning a reasonable number of results) that contain as many keywords as possible.

The idea is to use the keywords submitted in a search session up to the current query, and suggest a maximum query or a query cover as the user's next query. The requirement to contain as many of the keywords as possible reflects the following rationale. Taken together, the keywords of a search session describe the user's information need. Some of the keywords might not be appropriate (e.g. typos) and should be omitted, but the more keywords are contained in a maximum query or a query cover the better is the descriptiveness of the user's information need.

The rationale for requiring a reasonable number of hits per query also deserves closer consideration. Queries with empty result pages are useless and the same often applies to queries returning only a handful of hits. This gives a lower bound on the number of desired results. But there is also an upper bound since the number of results a user will consider for a single query is usually constrained by a processing capacity l_{max}, determined by the user's reading time etc. If the user faces a query with millions of hits, she can only check a fraction of the results—typically the top-ranked ones. Relevant entries below are missed. We argue that the best queries are the ones that are sufficiently specific to not return millions of hits—but also not just one or two. For such queries the user can check the complete result list and will not miss any potential match for her information need due to search engine ranking issues that she cannot influence.

Hence, from the user's perspective, maximum and covering queries contain the best possible description of the information need and offer the chance to check all the results. However, finding a maximum query or a query cover "by hand" is not that straightforward. Several queries have to be submitted to identify appropriate keyword combinations. Hardly any user will take the time for such a lengthy procedure. Therefore, we give algorithms for both tasks. To be applicable at user site the algorithms are of *external* nature, i.e., they only use standard search engine interfaces. The Web search engine is handled as a black box, acting like an oracle that answers queries. There is no need to know the underlying retrieval model or implementation details.

Since Web searches are not for free but entail costs—at the very least some non-negligible amount of time is consumed, and monetary charges come into play for larger contingents of queries—we analyze the corresponding economic optimization problem for finding maximum and covering queries. We ask:

Which strategy minimizes the average number of submitted queries?

A. Related Work

A lot of research has been done on approaches for better results on better queries. An example of a very promising

Consider for instance … Known-Item Finding

Scenario

- User accessed a document
- But did not store it

How can she find it again?

Making the Most of a Web Search Session

Benno Stein and Matthias Hagen
Bauhaus-Universität Weimar
<first name>.<last name>@uni-weimar.de

***Abstract*—We tackle problems related to Web query formulation: given the set of keywords from a search session, 1) we find a maximum promising Web query, and, 2) we construct a family of promising Web queries covering all keywords. A query is promising if it fulfills user-defined constraints on the number of returned hits. We assume a real-world setting where the user is not given direct access to a search engine's index, i.e., querying is possible only through an interface. The goal to be optimized is the overall number of submitted Web queries.**

For both problems we develop search strategies based on co-occurrence probabilities. The achieved performance gain is substantial: compared to the uninformed baselines without co-occurrence probabilities the expected savings are up to 50% in the number of submitted queries, index accesses, and runtime.

***Keywords*-Web Search Session, Query Formulation, Query Cost Optimization, Maximum Query, Query Cover**

I. Introduction

The interactions between Web search users and search engines follow a classic scheme. The user comes up with a set of (in her opinion) appropriate keywords for a given information need. She submits a query containing some of these keywords and gets back a ranked result list. If the user does not find a match for her information need among the first results, she will hardly browse all the items but submit different queries based on her keywords until she is satisfied or decides to give up. This process forms a *search session*—the set of consecutive Web queries a user submits to a search engine in order to satisfy a given information need.

Experience shows that in many cases a user's first try is answered reasonably well, i.e., the first query brings up an appropriate result. Such one-query-sessions are not the focus of this paper; we consider longer sessions where the user did not succeed with her first query. Search engines provide different means to support unsuccessful users, e.g., query expansion for queries returning lots of hits or spelling correction for queries returning no hits due to typos. In this paper we present two other approaches having a more combinatorial flavor, while being easily combinable with existing technology:

1) The *maximum query* for a given set of keywords, i.e., a query containing as many of the keywords as possible, while returning a reasonable number of results.
2) The *query cover* for a given set of keywords, i.e., a family of queries (each returning a reasonable number of results) that contain as many keywords as possible.

The idea is to use the keywords submitted in a search session up to the current query, and suggest a maximum query or a query cover as the user's next query. The requirement to contain as many of the keywords as possible reflects the following rationale. Taken together, the keywords of a search session describe the user's information need. Some of the keywords might not be appropriate (e.g. typos) and should be omitted, but the more keywords are contained in a maximum query or a query cover the better is the descriptiveness of the user's information need.

The rationale for requiring a reasonable number of hits per query also deserves closer consideration. Queries with empty result pages are useless and the same often applies to queries returning only a handful of hits. This gives a lower bound on the number of desired results. But there is also an upper bound since the number of results a user will consider for a single query is usually constrained by a processing capacity l_{max}, determined by the user's reading time etc. If the user faces a query with millions of hits, she can only check a fraction of the results—typically the top-ranked ones. Relevant entries below are missed. We argue that the best queries are the ones that are sufficiently specific to not return millions of hits—but also not just one or two. For such queries the user can check the complete result list and will not miss any potential match for her information need due to search engine ranking issues that she cannot influence.

Hence, from the user's perspective, maximum and covering queries contain the best possible description of the information need and offer the chance to check all the results. However, finding a maximum query or a query cover "by hand" is not that straightforward. Several queries have to be submitted to identify appropriate keyword combinations. Hardly any user will take the time for such a lengthy procedure. Therefore, we give algorithms for both tasks. To be applicable at user site the algorithms are of *external* nature, i.e., they only use standard search engine interfaces. The Web search engine is handled as a black box, acting like an oracle that answers queries. There is no need to know the underlying retrieval model or implementation details.

Since Web searches are not for free but entail costs—at the very least some non-negligible amount of time is consumed, and monetary charges come into play for larger contingents of queries—we analyze the corresponding economic optimization problem for finding maximum and covering queries. We ask:

Which strategy minimizes
the average number of submitted queries?

A. Related Work

A lot of research has been done on approaches for better results on better queries. An example of a very promising

Consider for instance . . . Known-Item Finding

Scenario

- User accessed a document
- But did not store it

How can she find it again?

Making the Most of a Web Search Session

Benno Stein and Matthias Hagen
Bauhaus-Universität Weimar
<first name>.<last name>@uni-weimar.de

***Abstract*—We tackle problems related to Web query formulation: given the set of keywords from a search session, 1) we find a maximum promising Web query, and, 2) we construct a family of promising Web queries covering all keywords. A query is promising if it fulfills user-defined constraints on the number of returned hits. We assume a real-world setting where the user is not given direct access to a search engine's index, i.e., querying is possible only through an interface. The goal to be optimized is the overall number of submitted Web queries.**

For both problems we develop search strategies based on co-occurrence probabilities. The achieved performance gain is substantial: compared to the uninformed baselines without co-occurrence probabilities the expected savings are up to 50% in the number of submitted queries, index accesses, and runtime.

***Keywords*-Web Search Session, Query Formulation, Query Cost Optimization, Maximum Query, Query Cover**

I. Introduction

The interactions between Web search users and search engines follow a classic scheme. The user comes up with a set of (in her opinion) appropriate keywords for a given information need. She submits a query containing some of these keywords and gets back a ranked result list. If the user does not find a match for her information need among the first results, she will hardly browse all the items but submit different queries based on her keywords until she is satisfied or decides to give up. This process forms a *search session*—the set of consecutive Web queries a user submits to a search engine in order to satisfy a given information need.

Experience shows that in many cases a user's first try is answered reasonably well, i.e., the first query brings up an appropriate result. Such one-query-sessions are not the focus of this paper; we consider longer sessions where the user did not succeed with her first query. Search engines provide different means to support unsuccessful users, e.g., query expansion for queries returning lots of hits or spelling correction for queries returning no hits due to typos. In this paper we present two other approaches having a more combinatorial flavor, while being easily combinable with existing technology:

1) The *maximum query* for a given set of keywords, i.e., a query containing as many of the keywords as possible, while returning a reasonable number of results.
2) The *query cover* for a given set of keywords, i.e., a family of queries (each returning a reasonable number of results) that contain as many keywords as possible.

The idea is to use the keywords submitted in a search session up to the current query, and suggest a maximum query or a query cover as the user's next query. The requirement to contain as many of the keywords as possible reflects the following rationale. Taken together, the keywords of a search session describe the user's information need. Some of the keywords might not be appropriate (e.g. typos) and should be omitted, but the more keywords are contained in a maximum query or a query cover the better is the descriptiveness of the user's information need.

The rationale for requiring a reasonable number of hits per query also deserves closer consideration. Queries with empty result pages are useless and the same often applies to queries returning only a handful of hits. This gives a lower bound on the number of desired results. But there is also an upper bound since the number of results a user will consider for a single query is usually constrained by a processing capacity l_{max}, determined by the user's reading time etc. If the user faces a query with millions of hits, she can only check a fraction of the results—typically the top-ranked ones. Relevant entries below are missed. We argue that the best queries are the ones that are sufficiently specific to not return millions of hits—but also not just one or two. For such queries the user can check the complete result list and will not miss any potential match for her information need due to search engine ranking issues that she cannot influence.

Hence, from the user's perspective, maximum and covering queries contain the best possible description of the information need and offer the chance to check all the results. However, finding a maximum query or a query cover "by hand" is not that straightforward. Several queries have to be submitted to identify appropriate keyword combinations. Hardly any user will take the time for such a lengthy procedure. Therefore, we give algorithms for both tasks. To be applicable at user site the algorithms are of *external* nature, i.e., they only use standard search engine interfaces. The Web search engine is handled as a black box, acting like an oracle that answers queries. There is no need to know the underlying retrieval model or implementation details.

Since Web searches are not for free but entail costs—at the very least some non-negligible amount of time is consumed, and monetary charges come into play for larger contingents of queries—we analyze the corresponding economic optimization problem for finding maximum and covering queries. We ask:

Which strategy minimizes the average number of submitted queries?

A. Related Work

A lot of research has been done on approaches for better results on better queries. An example of a very promising

Solution

- Remember some keywords

`information retrieval` `query formulation`
`web search` `search session` `user support`
`search engine` `cost optimization`

- Query a search engine

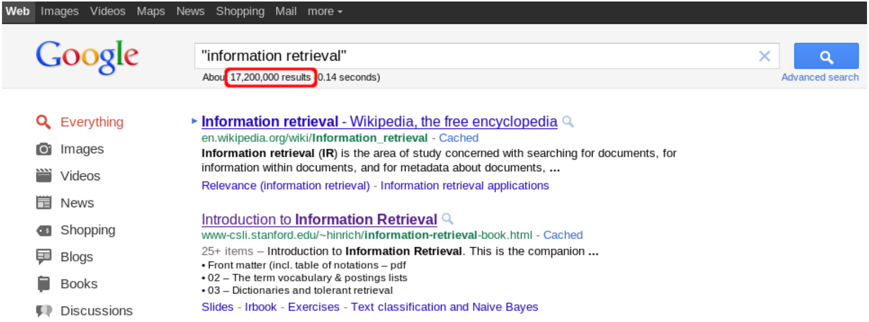
But what query to formulate with the keywords?

Single keywords?

information retrieval ~~query formulation~~

~~web search~~ ~~search session~~ ~~user support~~

~~search engine~~ ~~cost optimization~~

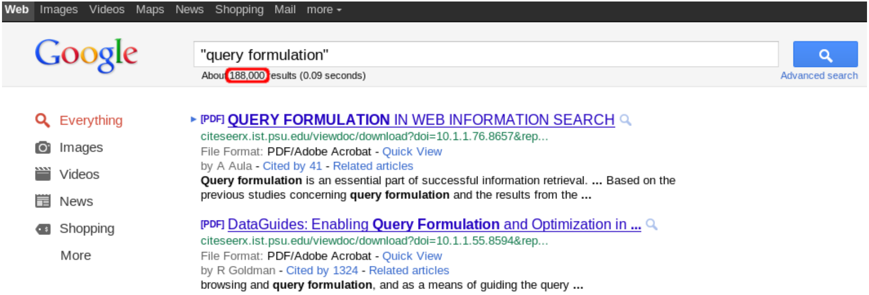


Single keywords?

~~information retrieval~~ query formulation

~~web search~~ ~~search session~~ ~~user support~~

~~search engine~~ ~~cost optimization~~

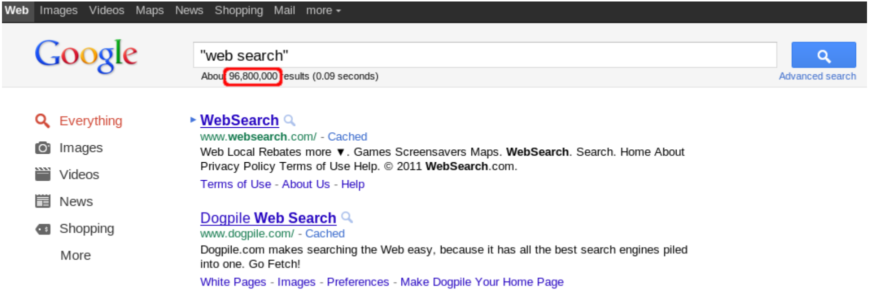


Single keywords?

~~information retrieval~~ ~~query formulation~~

`web search` ~~search session~~ ~~user support~~

~~search engine~~ ~~cost optimization~~

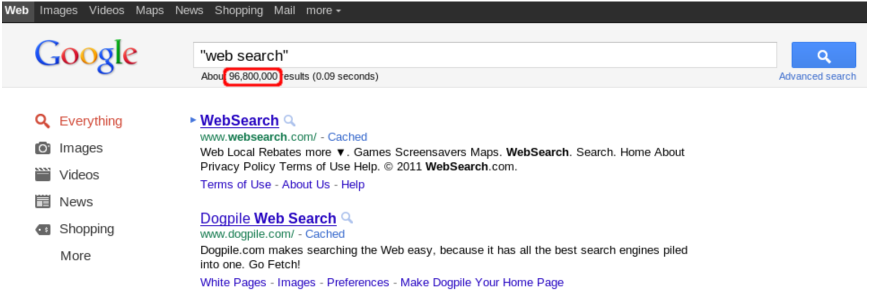


Single keywords? Underspecific!

~~information retrieval~~ ~~query formulation~~

web search ~~search session~~ ~~user support~~

~~search engine~~ ~~cost optimization~~

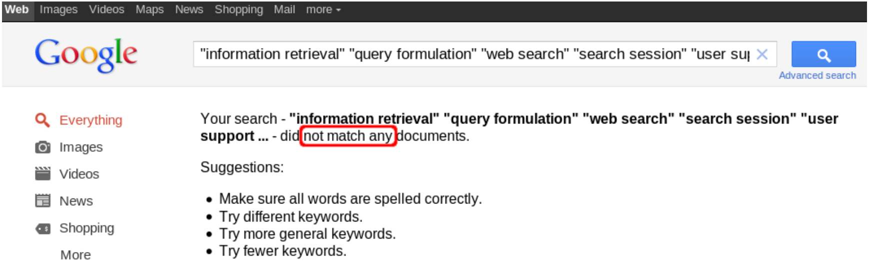


All keywords at once?

`information retrieval`   `query formulation`

`web search`   `search session`   `user support`

`search engine`   `cost optimization`

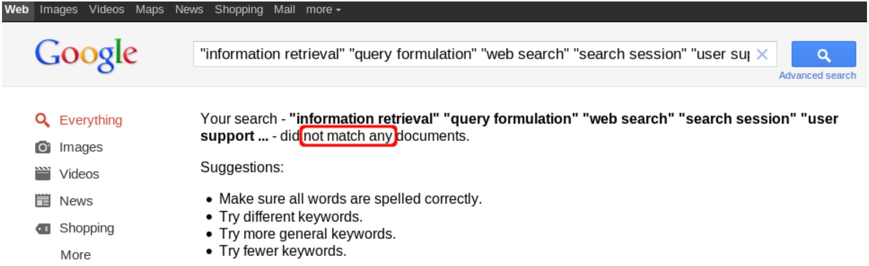


All keywords at once? Overspecific!

`information retrieval` `query formulation`

`web search` `search session` `user support`

`search engine` `cost optimization`



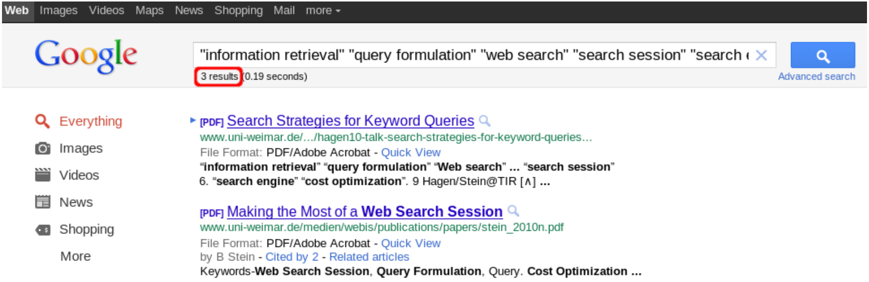
Remember the hypothesis . . . not too many results!

Solution: As many keywords as possible!

information retrieval   query formulation

web search   search session   ~~user support~~

search engine   cost optimization



“As many keywords as possible”-Query

Characteristics

- Captures most of the remembered keywords
- Best possible description of the known-item
- Not too many results → user can check complete list

“As many keywords as possible”-Query

Characteristics

- Captures most of the remembered keywords
- Best possible description of the known-item
- Not too many results $\rightarrow$ user can check complete list

Problem

- Relevant documents not known
- No web index at user site
- Query size not known

"As many keywords as possible"-Query

Characteristics

- Captures most of the remembered keywords
- Best possible description of the known-item
- Not too many results $\rightarrow$ user can check complete list

Problem

- Relevant documents not known
- No web index at user site $\rightarrow$ Lee et al. not applicable
- Query size not known

We propose an approach for this scenario . . .

Problem Statement with Capacity

Promising Query

- Given:
  1. A set W of keywords
  2. An upper bound k on the result list length
- Find a largest query $Q \subseteq W$ yielding at most k results

Optimization Problem!

Minimize the number of submitted web queries to find Q.

Problem Statement with Capacity

Promising Query

- Given:
  1. A set W of keywords
  2. An upper bound k on the result list length
- Find a largest query $Q \subseteq W$ yielding at most k results

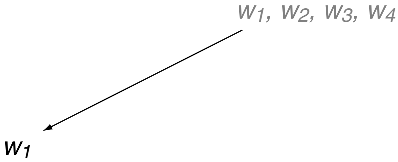
Optimization Problem!

Minimize the number of submitted web queries to find Q.

Baseline: Depth-First Search

w_1, w_2, w_3, w_4

Baseline: Depth-First Search

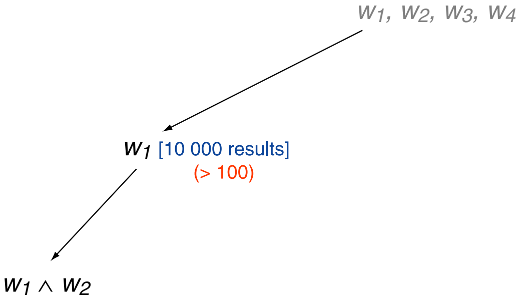


Baseline: Depth-First Search

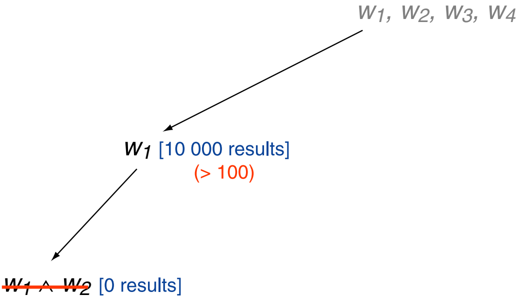
w_1, w_2, w_3, w_4

w_1 [10 000 results]
(> 100)

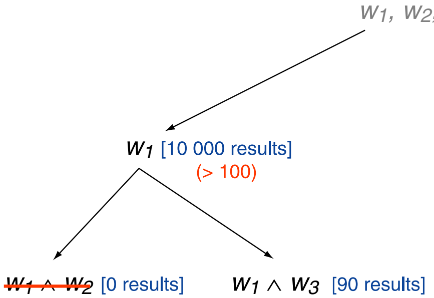
Baseline: Depth-First Search



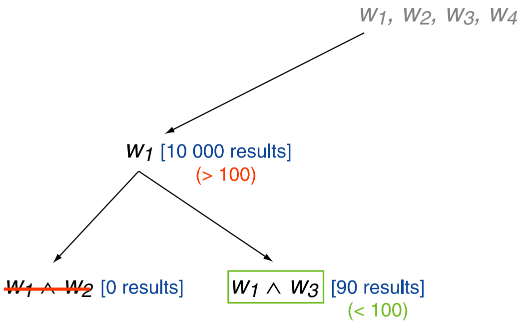
Baseline: Depth-First Search



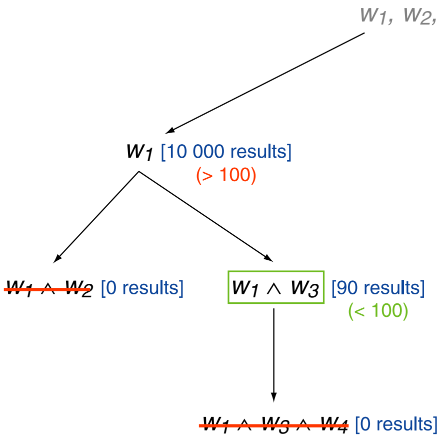
Baseline: Depth-First Search



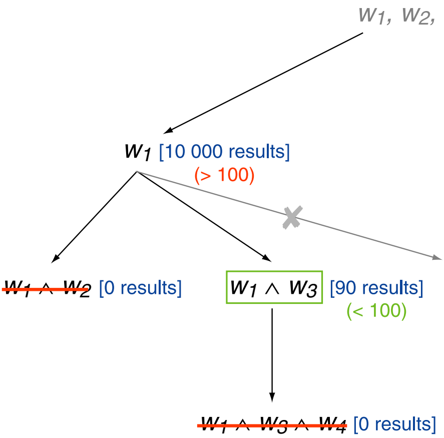
Baseline: Depth-First Search



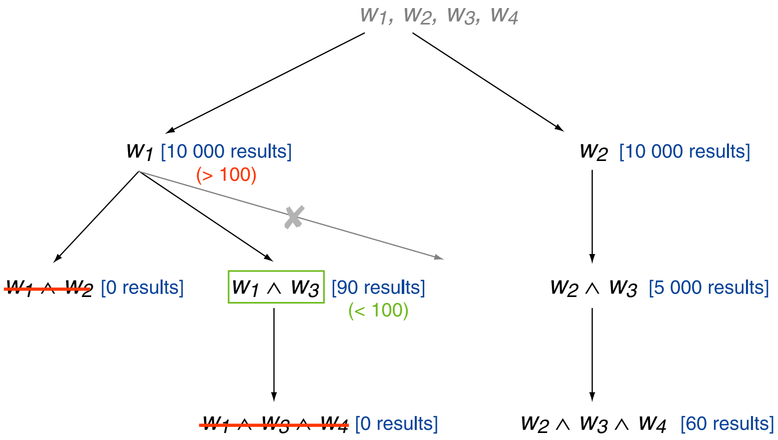
Baseline: Depth-First Search



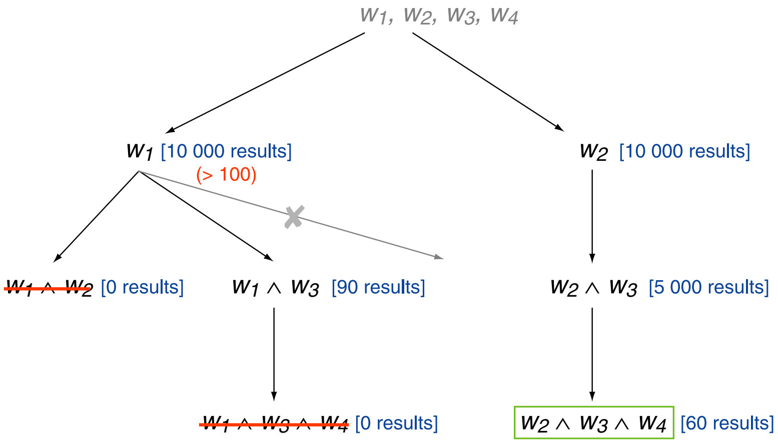
Baseline: Depth-First Search



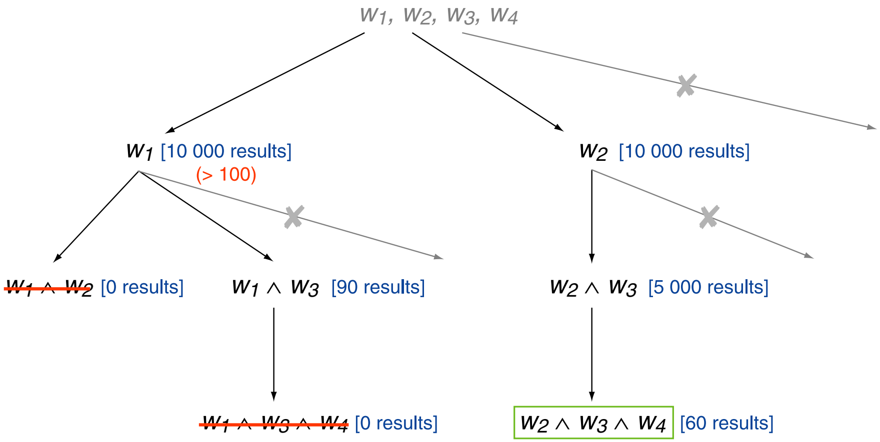
Baseline: Depth-First Search



Baseline: Depth-First Search



Baseline: Depth-First Search



Baseline's Analysis

Major Drawback

All intermediate queries submitted. $\rightarrow$ Bad run time!

Baseline's Analysis

Major Drawback

All intermediate queries submitted. $\rightarrow$ Bad run time!

Idea

Estimate the result list length before query submission.

Co-occurrence based Estimations

Estimate: `"information retrieval" "query formulation" + "web search"`

Co-occurrence based Estimations

Estimate: `"information retrieval" "query formulation"` + `"web search"`

Known: `"information retrieval" "query formulation"` 87 100 results

Co-occurrence based Estimations

Estimate: `"information retrieval" "query formulation" + "web search"`

Known:	`"information retrieval" "query formulation"`	87 100 results
	`"information retrieval" + "web search"`	16 % remain

Co-occurrence based Estimations

Estimate: `"information retrieval" "query formulation" + "web search"`

Known:	`"information retrieval" "query formulation"`	87 100 results
	`"information retrieval" + "web search"`	16 % remain
	`"query formulation" + "web search"`	22 % remain

Co-occurrence based Estimations

Estimate: `"information retrieval" "query formulation" + "web search"`

Known:	`"information retrieval" "query formulation"`	87 100 results
	`"information retrieval" + "web search"`	16 % remain
	`"query formulation" + "web search"`	22 % remain

Our estimation scheme:

$$\text{avg}(16\,\%\,,\ 22\,\%) \;=\; 19\,\%$$

$$87\,100 \cdot 0.19 \;=\; 16\,500 \text{ results}$$

Co-occurrence based Estimations

Estimate: `"information retrieval" "query formulation" + "web search"`

Known:	`"information retrieval" "query formulation"`		87 100 results
	`"information retrieval"`	`+ "web search"`	16 % remain
	`"query formulation"`	`+ "web search"`	22 % remain

Our estimation scheme: $\mathrm{avg}(16\,\%\,,\ 22\,\%) = 19\,\%$

$87\,100 \cdot 0.19 = 16\,500$ results

Control: Google 35 700 results

Co-occurrence based Estimations

Estimate: `"information retrieval" "query formulation" + "web search"`

Known:	`"information retrieval" "query formulation"`	87 100 results
	`"information retrieval" + "web search"`	16 % remain
	`"query formulation" + "web search"`	22 % remain

Our estimation scheme:

$$\mathrm{avg}(16\,\%\,,\ 22\,\%) = 19\,\%$$

$$87\,100 \cdot 0.19 = 16\,500 \text{ results}$$

Control: Google 35 700 results

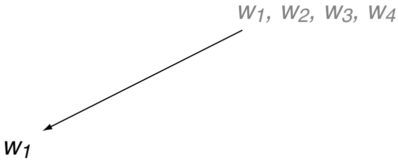
Observation

Our scheme usually underestimates the real result list length.

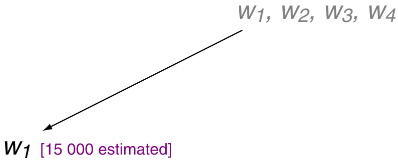
"Informed" Baseline

w_1, w_2, w_3, w_4

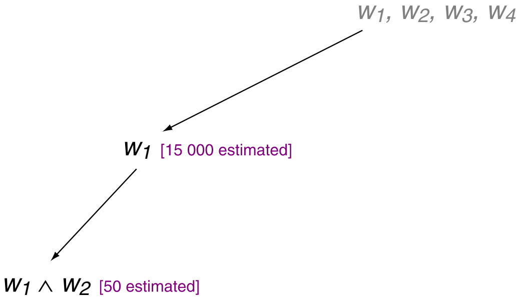
"Informed" Baseline



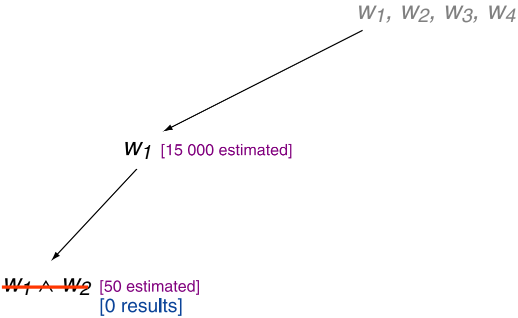
“Informed” Baseline



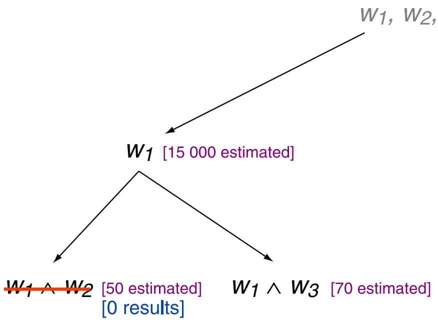
“Informed” Baseline



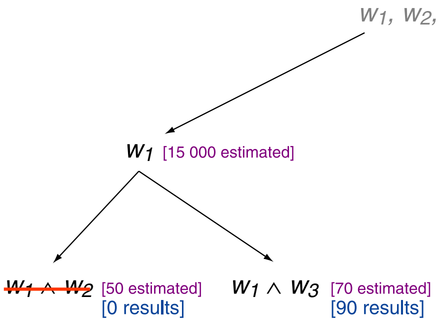
"Informed" Baseline



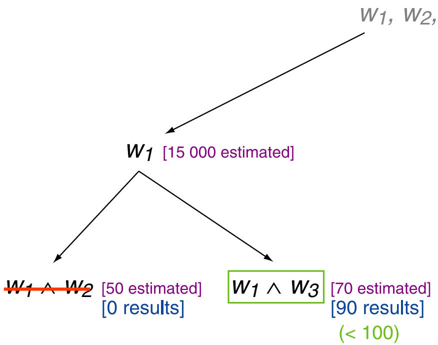
“Informed” Baseline



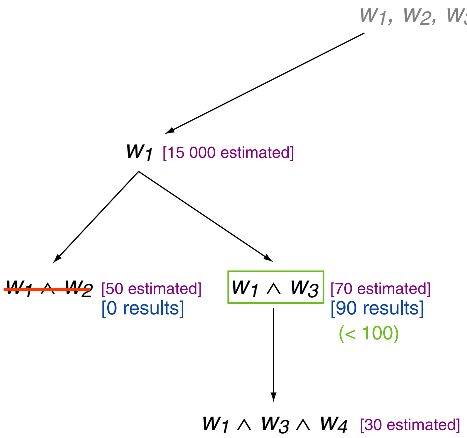
"Informed" Baseline



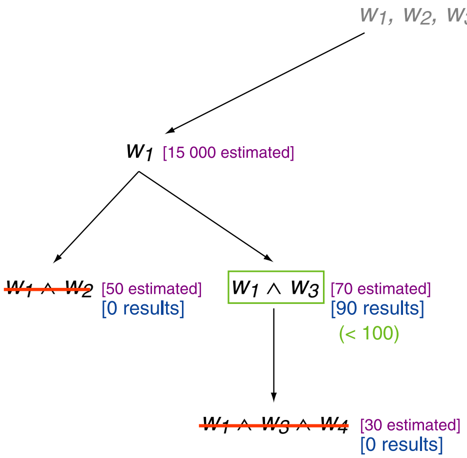
"Informed" Baseline



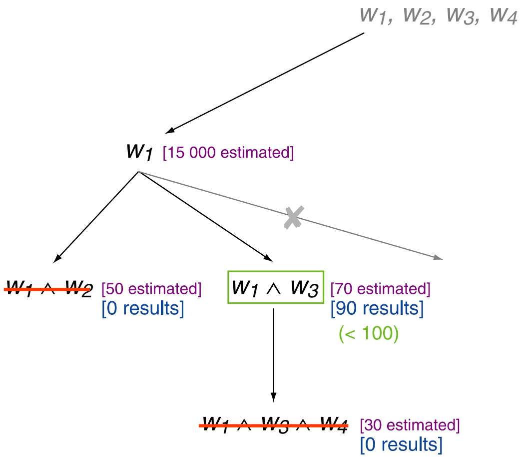
"Informed" Baseline



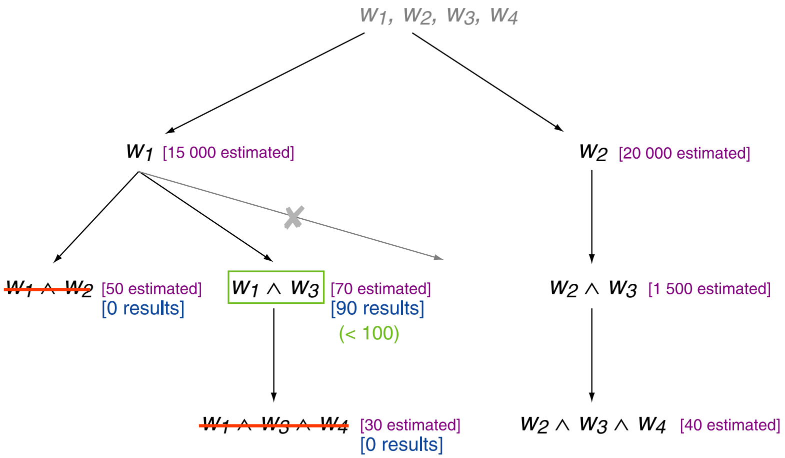
“Informed” Baseline



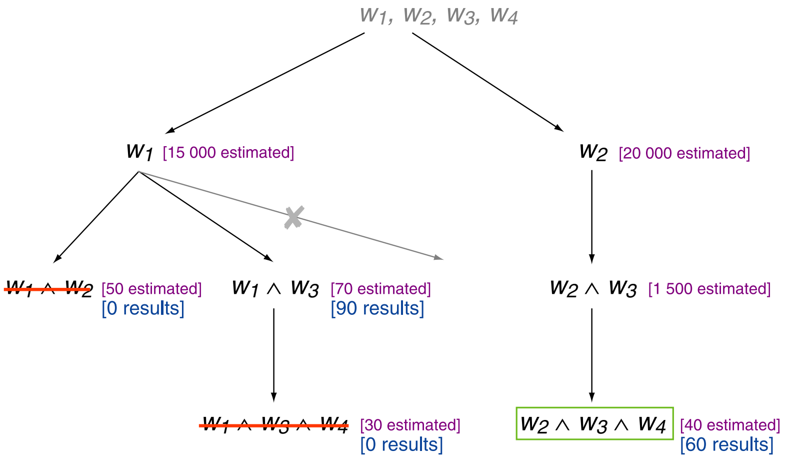
“Informed” Baseline



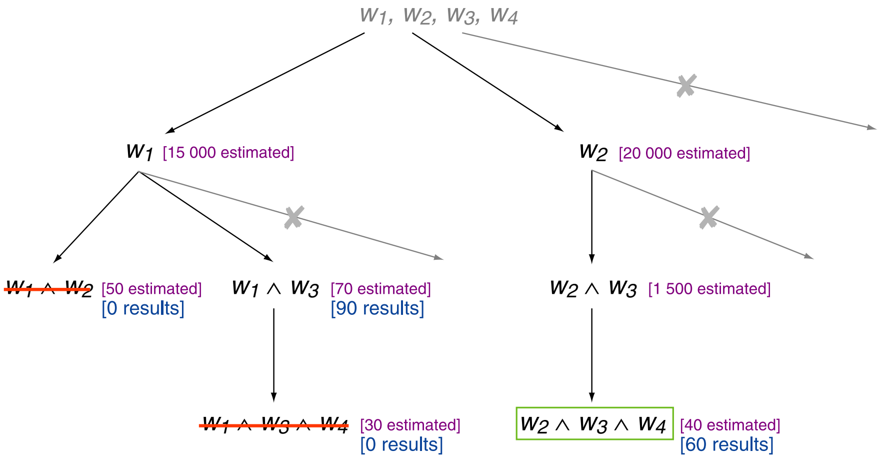
“Informed” Baseline



"Informed" Baseline



"Informed" Baseline



Heuristic

Informed baseline $+$ heuristic reordering of the keywords at each step

Experimental Setup

Corpus

- 775 papers on Computer Science (the known-items)
- 15 keywords extracted from each

System

- Bing API as search engine
- Set $k = 100$
- Measure number of submitted Web queries

Experimental Setup

Corpus

- 775 papers on Computer Science (the known-items)
- 15 keywords extracted from each

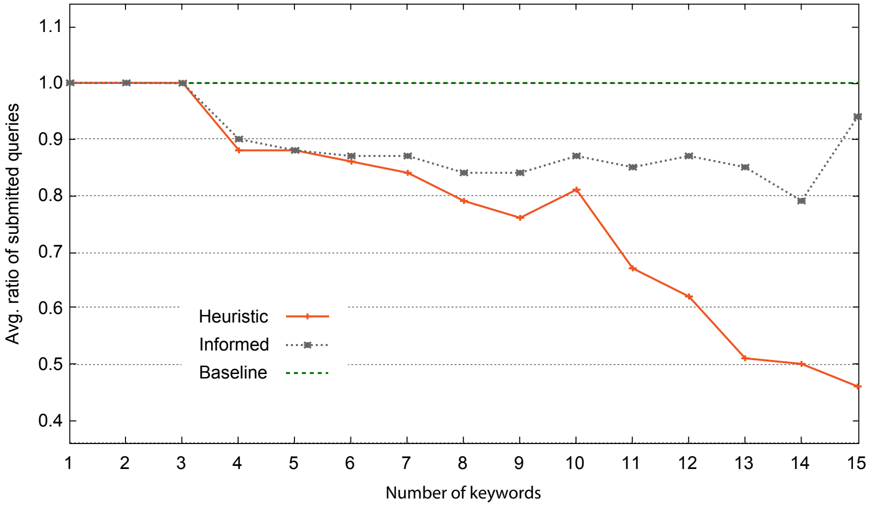
System

- Bing API as search engine
- Set $k = 100$
- Measure number of submitted Web queries

Experimental Results

Number of keywords		5	10	15
Promising query	not possible	614	328	86
	found	161	447	689
Avg. queries submitted	heuristic	10.39	**24.93**	**53.78**
	informed	**10.36**	27.01	108.78
	baseline	11.81	30.94	116.22

Experimental Results



Almost the end: The take-away messages!

What we have done

Results

- User-over-Ranking
  - longer queries → fewer results
  - optimum retrieval performance → user capacity
- Heuristic for promising queries
- Use cases:
  - Known-item finding
  - Empty results lists
  - Query sessions

Future Work

- Co-occurrence source
- User study

What we have (not) done

Results

- User-over-Ranking
  - longer queries → fewer results
  - optimum retrieval performance → user capacity
- Heuristic for promising queries
- Use cases:
  - Known-item finding
  - Empty results lists
  - Query sessions

Future Work

- Co-occurrence source
- User study

What we have (not) done

Results

- User-over-Ranking
  - longer queries → fewer results
  - optimum retrieval performance → user capacity
- Heuristic for promising queries
- Use cases:
  - Known-item finding
  - Empty results lists
  - Query sessions

Future Work

- Co-occurrence source
- User study

Thank you

☺