Query Completion / Expansion

COMP90042 LECTURE 18, THE UNIVERSITY OF MELBOURNE

by

Matthias Petri

Tue 8/5/2018

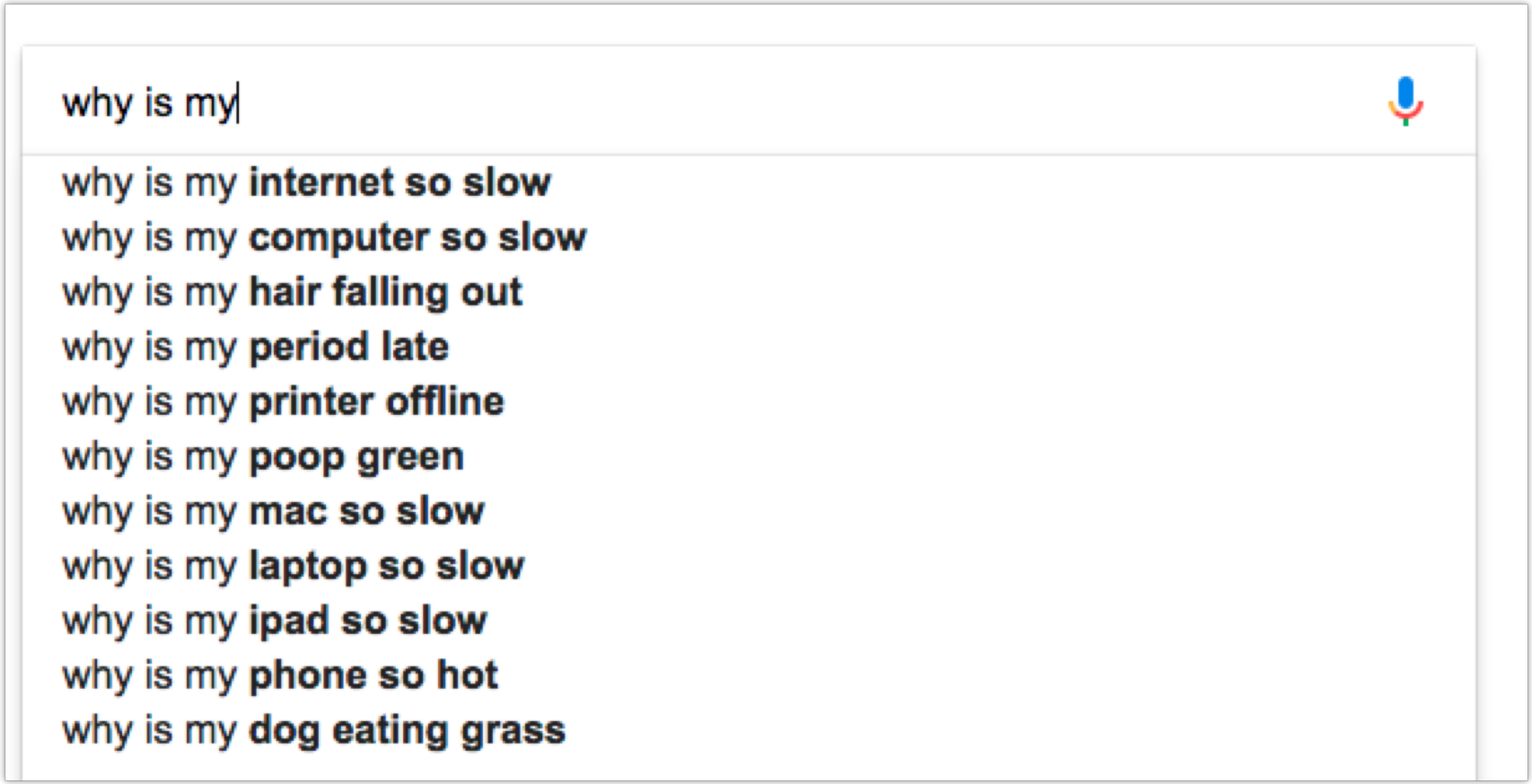
What is a query?

What is a query?

1. Obviously the stuff I type into the search box!
2. Most likely not the query that gets handed over to the search index.
3. Why not?

Query Completion

Query Completion



What is a Query Completion?

Goals:

1. Assist users to formulate search requests.
2. Reduce number of keystrokes required to enter query.
3. Help with spelling query terms.
4. Guide user towards what a good query might be.
5. Cache results! Reduce server load.

Strategy:

1. Generate list of completions based on partial query.
2. Refine suggestions as more keys are pressed.
3. Stop once users selects candidate or completion fails.
4. Why not a Language Model? Might not return results!

High Level Algorithm

Given a query pattern P,

1. Retrieve set of **candidates** "matching" P from set S of possible target queries.
2. Rank candidates by frequency.
3. Possibly re-rank highest ranked candidates with more complex ranking measure (e.g. personalized)
4. Return the top-K highest ranking candidates as suggestions.

Completion Targets

Where does the set S of possible completions come from?

1. Most popular queries (websearch)
2. Items listed on website (ecommerce)
3. Past queries by the user (email search)

Properties:

1. Static (e.g. completion for “twi”)
2. Dynamic (e.g. time-sensitive, “world cup”)
3. Massive or small (email search vs websearch)

Completion Types / Completion

Given a partial user query *P*, how is the initial candidate set retrieved?

Modes:

1. Prefix match.
2. Substring match.
3. Multi-term prefix match.
4. Relaxed match.

Example: Target “FIFA world cup 2018“:

P	Mode 1	Mode 2	Mode 3	Mode 4
`FIFA wo`	x	x	x	x
`orl`		x		
`FI wor`			x	x
`FIFO warld cu`				x

Prefix Completion

Problem:

Given a query prefix P, retrieve the top-K most popular completions.

Data:

Static query log consisting of all queries received by the search index.

Requirements:

1. Fast retrieval time required. What is fast?
2. Space efficient index.

Prefix match - Trie+RMQ based Index

Step 1: Preprocess data by sorting query log in lexicographical order and counting frequency of unique queries:

Before	After
bunnings	$< \textit{bunnings}, 47 >$
bachelor in paradise	$< \textit{big w}, 5 >$
bbc news	$< \textit{bbc news}, 12 >$
bunnings	$< \textit{bachelor in paradise}, 2 >$
big w	
bbc news	
big w	

Prefix match - Trie+RMQ based Index

Step 2: Insert all unique queries and their frequencies into a trie (also called a prefix tree).

What is a trie?

- A tree representing a set of strings.
- Edges of the tree are labeled.
- Children of nodes are ordered.
- Root to node path represents prefix of all strings in the subtree starting at that node.

Prefix match - Trie Example

Set of strings:

- nba
- news
- nab
- ngv
- netflix
- netbank
- network
- netball
- netbeans

`https://www.cs.usfca.edu/~galles/visualization/Trie.html`

Prefix match - Trie+RMQ based Index

Prefix search using a trie

Insert queries into trie. For a pattern P, find node in trie representing the subtree prefixed by P in $O(|P|)$ time.

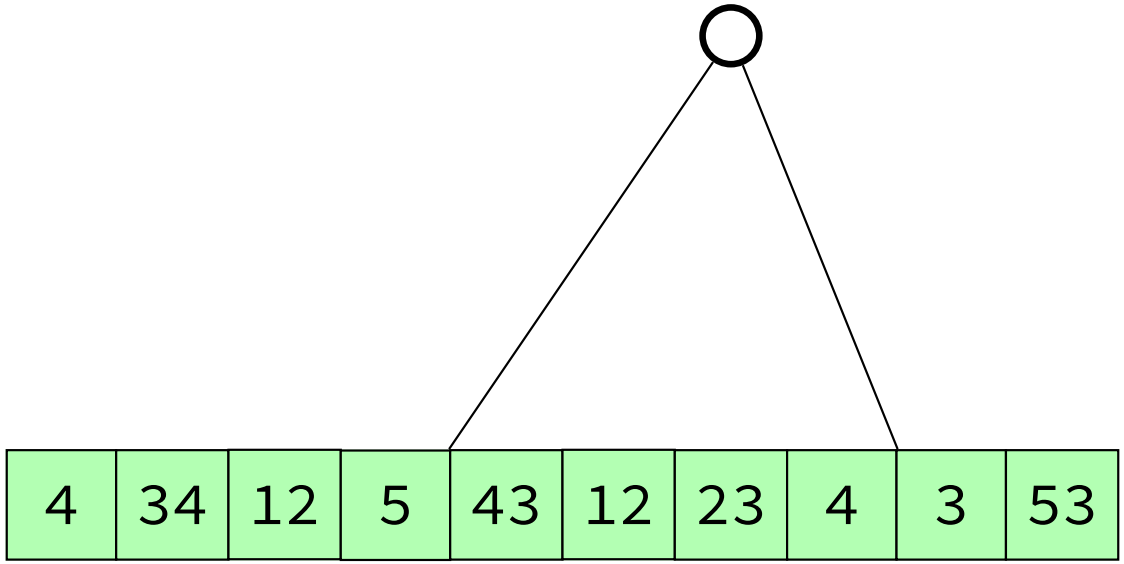
Observation:

The subtree prefixed by P corresponds to a continuous range.

Prefix match - Trie+RMQ based Index

Idea:

Store array with frequencies corresponding to each query. Subtree corresponds to range in frequency array. Find the top-*K* highest numbers in that range.



Range Maximum Queries

Task:

Given an array A of n numbers, and a range $[l, r]$ of size m, find the **positions** of the K largest numbers in $A[l, r]$.

Simple algorithm:

1. Copy $A[l, r]$ into an array B in $O(m)$ time.
2. Sort B in $O(m \log m)$ time.
3. Return positions of largest numbers in $A[l, r]$.

Problem:

- Runtime also depends on the size of the range m and requires $O(m)$ extra space.
- m can be large. We require low millisecond response times.

Range Maximum Queries - Index

Finding the Maximum in a Range in $O(1)$ time:

- Array A is size n.
- There are $O(n^2)$ different ranges $A[i, j]$
- For each range precompute the position of the maximum. Uses $O(n^2)$ space.

Extension to K largest numbers:

1. Find position p of largest element on $A[i, j]$.
2. Recurse to $A[i, p-1]$ and $A[p+1, j]$.
3. Keep going until you have the K largest elements.
4. Runtime $O(K \log m)$.

RMQ Index- Reduce space

Simple space reduction:

- Instead of precomputing all $O(n^2)$ ranges $A[i, j]$, for each position $A[i]$, precompute only $\log n$ ranges of increasing size: $A[i, i+1]$,$A[i, i+2]$,$A[i, i+4]$,$A[i, i+8]$.
- Any range $A[l, r]$ can be decomposed into two ranges $A[l, Y]$ and $A[Z, r]$ where $Y = l + 2^x$ and $Z = r - 2^y$ such that $Z \geq l, Y \leq r$ and, $A[l, Y], A[Z, r]$ overlap. Then, $RMQ(A[i, j]) = max(RMQ(A[l, Y]), RMQ(A[Z, r]))$
- Total space cost $O(n \log n)$.

Prefix Completion - In Practice

- Space efficient (compressed) Trie+RMQ representations used (more complex)
- RMQ+Trie requires roughly 10 bytes per string (roughly the size of gzip).
- 1 billion unique strings require an index of size 10GB RAM.
- Can answer top-10 queries in less than 10 microseconds.

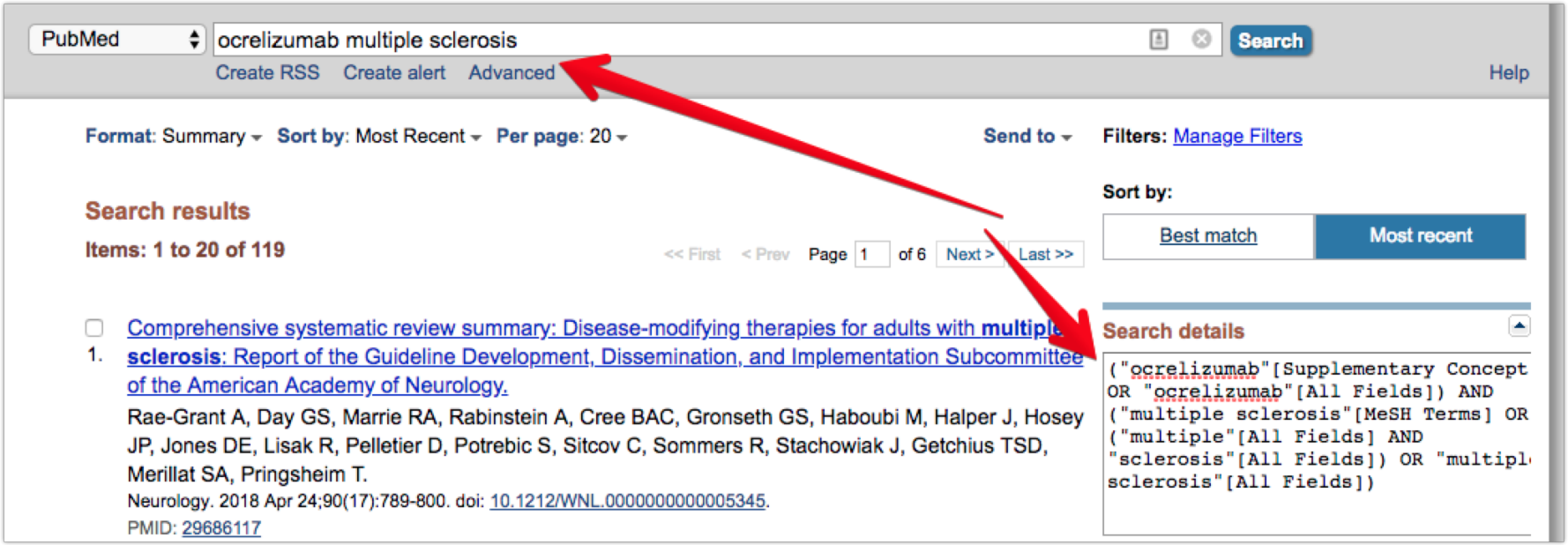
Query Expansion

Query Expansion - What is it?

- User and documents may refer to a concept using different words (poison $\leftrightarrow$ toxin, danger $\leftrightarrow$ hazard, postings list $\leftrightarrow$ inverted list)
- Vocabulary mismatch can have impact on recall
- Users often attempt to fix this problem manually (query reformulation)
- Adding these synonyms should improve query performance (query expansion)

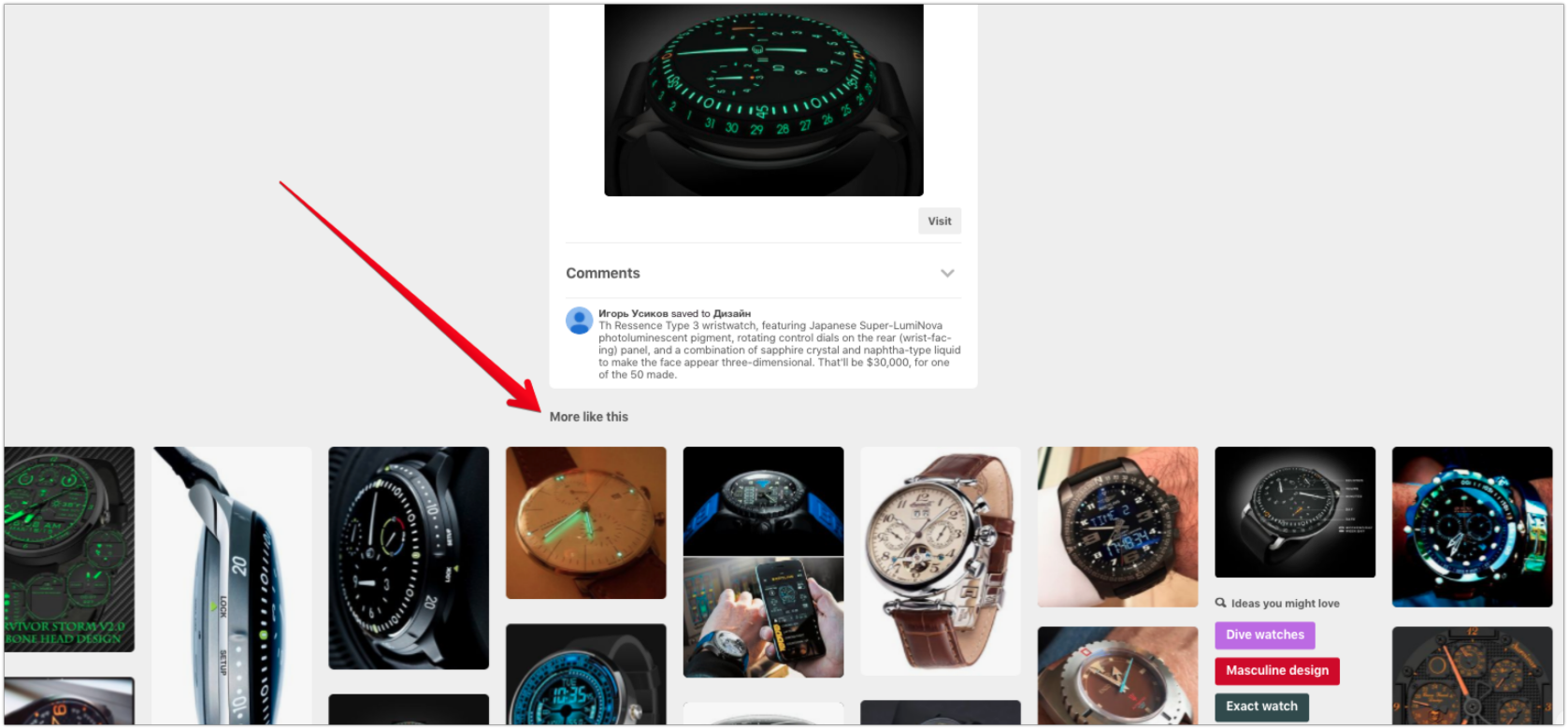
Global Query Expansion

- Retrieve synonyms from thesaurus or WordNet (medical domain)
- Spell correction (importamt $\rightarrow$ important)
- Word2Vec (what words are close to the query words?)



User relevance feedback

Relevance Feedback. User provides feedback to the search engine by indicating which results are relevant



Pseudorelevance feedback

- Take top-K results of original query
- Determine important/informative terms/topics (topic modelling!) shared by those documents
- Expand query by those terms
- No explicit user feedback needed (also called blind relevance feedback)

Example

- Original query: what is a prime factors
- Expanded query: what is a prime factors integer number composite common divisor

Indirect relevance feedback

- For a query look at what users click on in the result page
- Use clicks as signal of relevance
- Learning-2-Rank uses neural models to rerank result pages (later this semester)

Query Expansion - Summary

- Helps with vocabulary mismatch
- Can improve recall
- Global expansion
- User, pseudo or indirect relevance feedback

Further Reading

Reading:

- Manning, Christopher D; Raghavan, Prabhakar; Schütze, Hinrich; Introduction to information retrieval, Cambridge University Press 2008. (Chapter 9)

Additional References:

- Unni Krishnan, Alistair Moffat, Justin Zobel: A Taxonomy of Query Auto Completion Modes. ADCS 2017: 6:1-6:8
- Amati, Giambattista (2003) Probability models for information retrieval based on divergence from randomness. PhD thesis.