Search engine based on FCA

Lukas Havrlant

DEPARTMENT OF COMPUTER SCIENCE
PALACKÝ UNIVERSITY, OLOMOUC

CLaSeek

- CLaSeek = Concept Lattice Seeker
- Web search engine, works with a static or dynamic set of documents
- Main feature: after submitting a query, CLaSeek offers suggestions for a new query:
  - More general query (remove a word),
  - more specific query (add a word),
  - and similar query.
- Query: "jaguar car":
  - More general: "jaguar" (remove "car"),
  - more specific: "jaguar car used" (add "used"),
  - similar: "prices jaguar"

First phase: indexing

- Input: a set of URLs.
- Output: an index, like in a book:
- Hash table: word $\rightarrow$ a set of documents which contain a word.
- "computer" $\rightarrow$ [['doc1', 5], ['doc7', 10], ['doc42', 2]]
- CLaSeek Downloads files, converts them to plain text, remove accents/diacritics, stemming words (finding a root of words: "fishing", "fished" $\longrightarrow$ "fish"), builds an index.
- CLaSeek supports HTML, PDF, ODT, RTF, TXT.

Second phase: Finding keywords of documents

- Keywords of document = a set of "most important" words in a document.
- A keyword should be frequently occur in the document, but should not be frequently occur in the rest of documents.
- Example: "javascript" is probably not a keyword of a site "loops in javascript".

The tf-idf algorithm

- $\mathrm{tf}_{t,d}$ – term frequncy. Number of words t in a document d.
- df_t – document frequency. Number of documents, which contain a word t.
- $\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t}$ – inverse document frequency. (N is number of all documents)
- $\mathrm{tf} - \mathrm{idf}_{t,d} = \mathrm{tf}_{t,d} \cdot \mathrm{idf}_t$
- Greater value of $\mathrm{tf} - \mathrm{idf}_{t,d}$ = a word t is more important in a document d.
- Set of keywords for each document = set of words with greatest $\mathrm{tf} - \mathrm{idf}_{t,d}$ value.
- Two approaches are implemented: top n words or all words with $\mathrm{tf} - \mathrm{idf}_{t,d} \geq$ treshold (or combination).

Third phase: retrieving results

- Basic boolean operators are supporoted: AND, OR and NOT.
- ((computer AND science) OR (mathematics)) AND NOT programming
- AND = intersection, OR = union, NOT = difference.
- For ordering the $\mathrm{tf} - \mathrm{idf}$ function is used.

Fourth phase: computing suggestions via FCA

- Search for an extended query: "computer science" $\longrightarrow$ "computer OR science".
- Build a context $\langle X, Y, I \rangle$:
  - X is a set of documents from the extended query,
  - Y is a set of keywords from the extended query,
  - I is a realation "is a keyword contained in a document?"

	loop	function	arithmetic	integer	...
1.html	$\times$			$\times$	
2.html		$\times$			
3.html	$\times$	$\times$		$\times$	
4.html			$\times$	$\times$	
...					

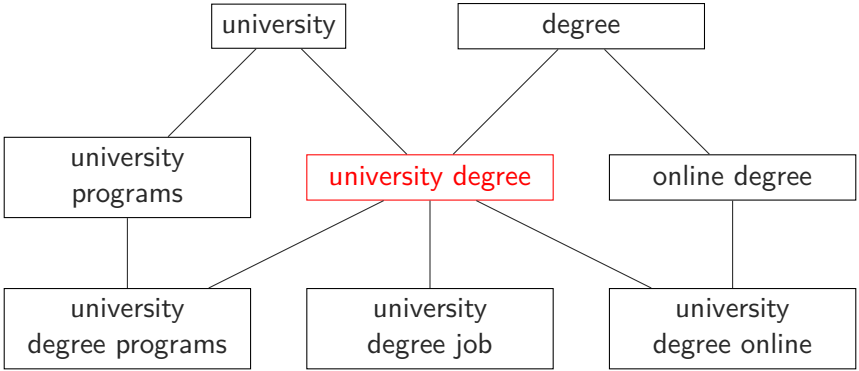
Query concept

- Query concept $\mathcal{C}$ is a concept (rectangle) which can represent the query.
- "computer science" $= \{\text{computer}, \text{science}\}$.
- For an "AND"-query Q: $\mathcal{C} = \left\langle Q^{\downarrow}, Q^{\downarrow\uparrow} \right\rangle$.
- For example, for a query "loop integer" we will get:

	loop	function	arithmetic	integer	...
1.html	×			×	
2.html		×			
3.html	×	×		×	
4.html			×	×	
...					

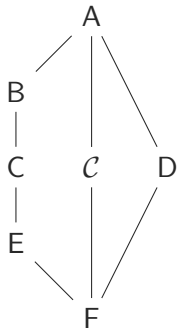
Generating suggestions

First, we find upper/lower neighbours (Lindig's Algorithm) and siblings. We will get a part of a concept lattice.



More general queries are in upper neighbours, similar are in siblings and more specific are in lower neighbours.

How to find siblings



$$\text{siblings}(\mathcal{C}) = (\text{LN}(\text{UN}(\mathcal{C})) \cap \text{UN}(\text{LN}(\mathcal{C}))) \setminus \{\mathcal{C}\}$$

Dynamic set of documents

- CLaSeek is connected to the search engine Bing.
- A user submit a query, the query is send to the Bing, Bing returns results, CLaSeek use the results to compute suggestions.
- CLaSeek has a public API, anyone can write an extension like this.
- Don't like Bing? Use the API and connect CLaSeek to Google!

Conclusion

- Google can suggest you a new query, but it is (probably) based on a search history.
- CLaSeek uses a content of documents only.
- Results of CLaSeek are good enough if documents are good enough. For example CLaSeek can't handle too short documents very well.
- CLaSeek can recognize a typign error.
- There are (or were) similar search engines: CREDO, FooCA a SearchSleuth.
- Demonstration...

Thank you for your attention!