LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

The implementation of WiTTFind

Presentation Wittgenstein Scholarship 2014

Florian Fink

Centrum für Informations- und Sprachverarbeitung (CIS) LMU

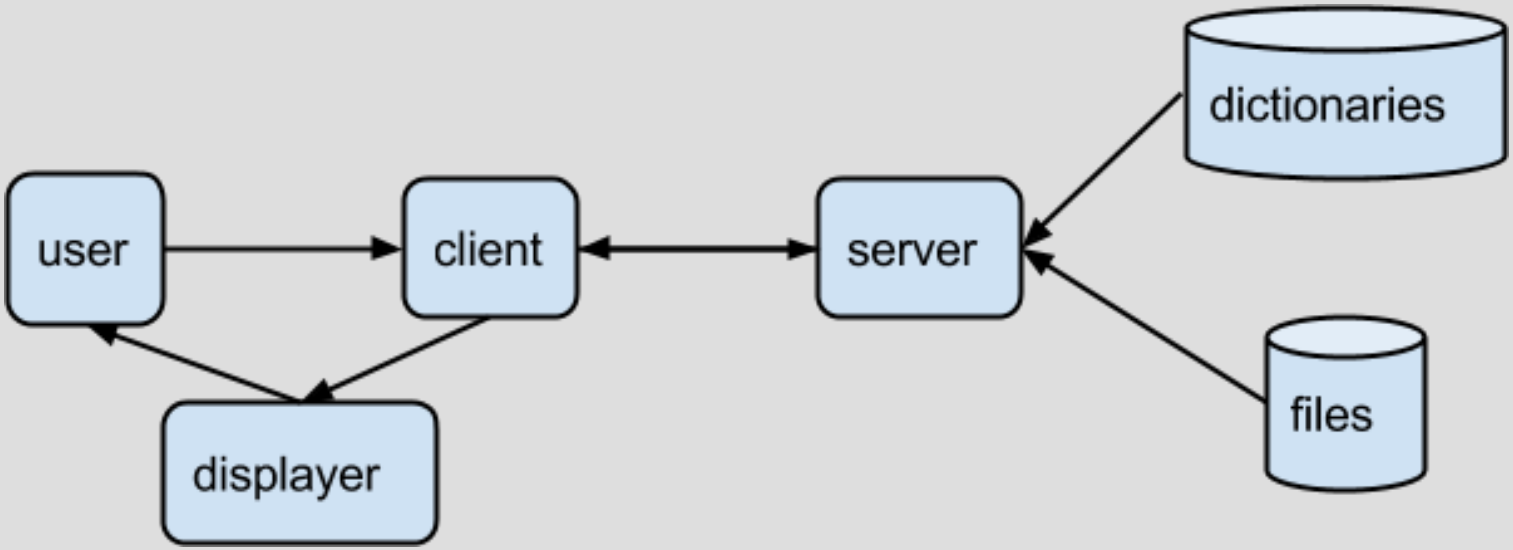
6. Juni 2014

Wittfind

The tool Wittfind (`wf`) is part of the Wittgenstein Advanced Search Tools (`WAST`).

- It acts as search-backend for other tools of `WAST`.
- It is written entirely in C++ and uses a simple XML-DOM parser called *pugixml*.
- It searches over any compatible files using both external lexical resources and embedded part-of-speech tags.
- It finds phrases in sentences using graphs that combine different matching algorithms.
- It shows the results of the search within the original document.

Tools of `wf`



`wf` is split into 3 parts:

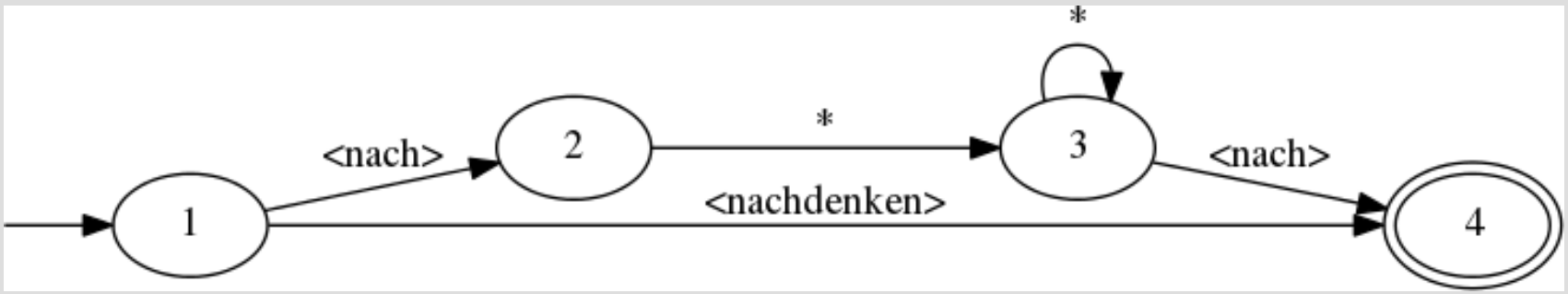
- `wf_server` implements the search and handles the background resources.
- `wf_client` creates search-graphs and controls the server.
- `wf_display` handles the presentation of the search results.

wf_server

The wf_server handles all background resources used by the program and searches the file index.

1. applies preprocessing to the file index and adds all available information to the token in order to speed up the lookup.
2. accepts search graphs from the client
3. searches the given file(s) with the search graph
4. returns the results back to the client

Representation of search graphs



Internally search graphs are represented as a set of states that are connected with transitions. Each search graph has on starting state and one final state. If a final state is encountered during the search, a hit is reported back to the client.

Transitions of the search graph

Transitions represent links from one state to another. If a transitions matches the current token it return the target state of the transition. There are different transitions that implement different matching algorithms:

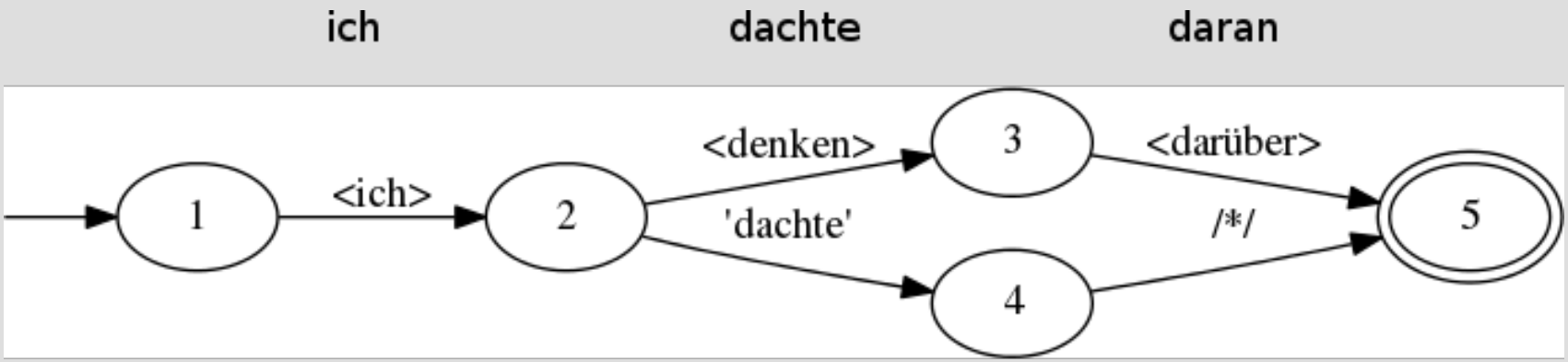
- algorithms using the information of the background dictionaries
  - lematized matching
  - matching on morphological forms
  - matching on semantic forms
- algorithms using the embedded information of the text
  - regular expression matching on the token
  - simple string matching
  - matching on the embedded pos tags

Searching

The search algorithm simply iterates over token and the states in parallel using a stack of the active states. It borrows from algorithms that are used to traverse NFA's.

1. At first the initial state is pushed onto the stack.
2. The current state is popped from the stack and the next token is read.
3. The target state of every transition of the popped state that matches the current token is pushed onto the stack.
4. If the final state lies on top of the stack, a match is reported.
5. If the stack is empty nothing could be found and the search starts over.

Example of the search

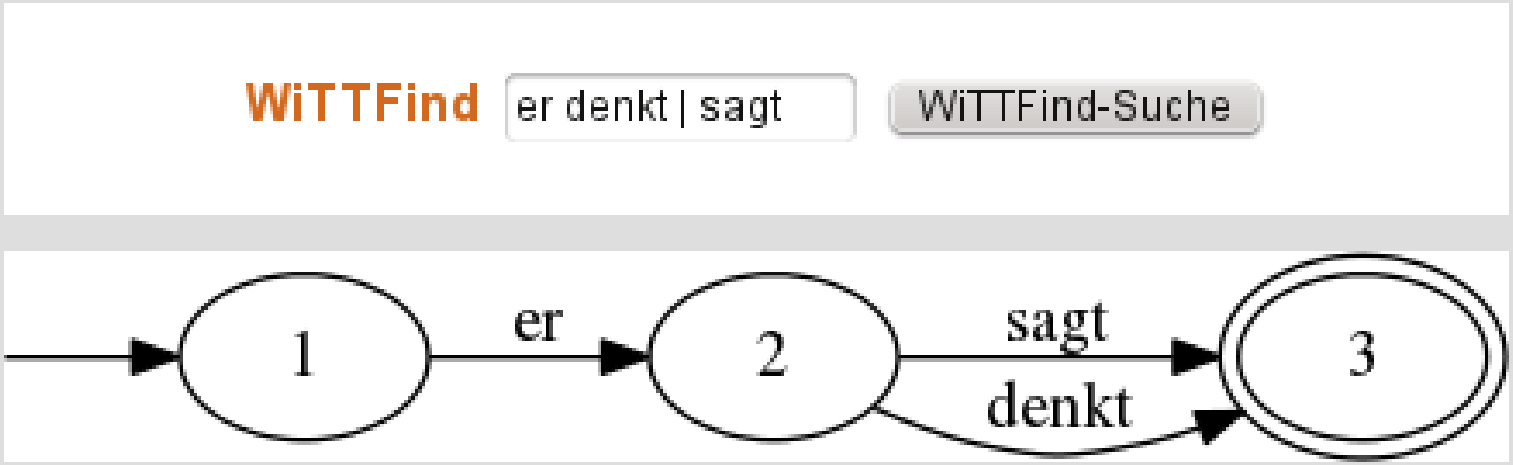


wf_client

The main purpose of the wf_client is the communication with wf_server.

- constructs search-graphs
- sends search-graphs to the server
- recieves the search results from the server

Flat queries



In order to provide a simple interface to searching without the need to explicitly constructing the search graphs, `wf_client` is able to transform flat queries to search-graphs before sending them to the server.

Transformation of flat queries

The algorithm used for the transformation are borrowed from algorithms that transform regular expressions to NFA's, which are used by simple search algorithms of regular expressions.

- The current algorithm is based on the Thompson construction, that imposes a lot of superfluous *empty* transitions.
- Another (optional) algorithm uses the Glushkov construction, which results in less transitions.

wf_display

The last tool in the pipeline is wf_display. It is used to mark the matches in the matching sentences. The tool just attaches some information to the original document without changing any other information in it and sends it as answer back to the user.

Conclusion

- The search is implemented like a simple algorithm for regular expressions search on the basis of token.
- Instead of character-based comparison, different token matching algorithms can be used.
- The server preprocesses the file-index for faster access.
- The client constructs search-graphs and controls the server.
- The displayer displays the results in the original file.

Thank you!