How I Spent My Summer Vacation:
Research in Crawling the Virtual Web
Data Sciences Summer Institute (DSSI)

- Eight-week program for graduates and undergraduates funded by the Department of Homeland Security
- Run by the Multimodal Information Access and Synthesis (MIAS) group at the University of Illinois at Urbana-Champaign
- Program consisted of classes, tutorials and three research projects: Named Entity Recognition, Image Recognition, and the Virtual Web
- [http://mias.uiuc.edu/dssi/](http://mias.uiuc.edu/dssi/)
Virtual Web Project @ UIUC

- Run by Professor Kevin Chang
- Virtual Web stored on several servers which are automatically updated periodically
  - Fetch as many URLs and web pages as possible and store them locally
  - Uses a modified version of Nutch ([http://lucene.apache.org/nutch/](http://lucene.apache.org/nutch/)), which runs on Lucene
- Why have a Virtual Web?
  - The web is far too large and dynamic a test-bed for many projects
    - But Virtual Web projects can easily scale to the size and scope of the real web
  - Crawling is expensive, uses web resources that everyone shares, and can violate common courtesy
    - Many webmasters become angry if their site receives too many hits from the same IP
Crawling the Web and the Virtual Web

- What is a web crawler?
  - “a program or automated script which browses the World Wide Web in a methodical, automated manner” (Wikipedia)

- Web crawlers consist of one or more seed URLs or keywords, an operation and a classifier
  - Performs the operation on the URLs or keywords and classifies the results accordingly

- Want to build our own web crawlers simply by providing as input the URLs/keywords, operations and classifiers and receiving as output the crawler we want
  - Should be automatic, and intuitive
Intuitive?

- Consider the web as a graph, with pages as nodes and the links between them as edges
  - Or better yet, states and transitions
- Web crawling should function in the exact same way
  - Multiple crawlers should be linked together similarly as states and transitions
  - User should be allowed to visualize the crawler graph they wish to construct
The (Proposed) Visual Crawler

- Every web crawler consists of one or more “mini-crawlers”, which perform an operation with the seed URLs/keywords and classify the results.
- Seeds can be URL or keyword results from a previous mini-crawler.
  - This creates the transition between two mini-crawlers.
  - Any mini-crawler can connect to any other mini-crawler, creating loops and duplicate mini-crawlers.
- Inspired by Yahoo! Pipes (http://pipes.yahoo.com/pipes/)
An Example

Crawler 1
Normal Search
“urbana champaign”

Crawler 2
Backward Crawl
http://www.cs.uiuc.edu

Crawler 3
Get Child Links
http://www.kr.xemacs.org/

Crawler 4
Forward Crawl
http://www.es.xemacs.org/

http://www.uiuc.edu
http://www.cs.uiuc.edu
http://www.ucimc.org
...
...
...
http://www.kr.xemacs.org/
http://www.es.xemacs.org/
...

...and so on
Another Example


Crawler 1
Get Parent Links
http://www.facebook.com

Crawler 2
Forward Crawl
http://www.blogforward.com/money/
http://www.wibbler.com/index.php
http://www.mipsscan.com/

Crawler 3
Backward Crawl
http://www.feedburner.com

http://www.feedburner.com/

http://www.blogforward.com/money/
http://www.wibbler.com/index.php
http://www.mipsscan.com/

http://www.moxie-drive.com/links.php
http://performancing.com/blog/40998
http://www.sunblogger.de/category/allgemein/
What I Built

- Nowhere near finished 😊
- Written in Java 6 with Swing
- Not yet fully automated
  - Users still need to specify each mini-crawler and connect them somewhat manually
    - The GUI provides the states of the graph and the user provides the connections between them
  - Doesn’t contain classifiers
- Runs on top of the code used to build crawlers, so not a “pure” visual crawler
  - But still, much more intuitive and, in the future, efficient
    - Visual crawlers make it easier to reuse crawlers in a drag-and-drop fashion, as opposed to re-writing the code (remember Yahoo! Pipes?)
Web Crawlers as Finite State Push-Down Automata

- Can the visual crawler build all possible web crawlers?
  - Yes, if the crawler is perfect and complete
    - A perfect visual crawler can do all possible operations
    - Number of operations is finite, web is finite, therefore number of possible crawlers is finite
  - Finite number of operations and crawlers => finite number of transitions between operations
    - Crawlers consisting of mini-crawlers are equivalent to states and transitions, remember?
    - Multiple transitions are possible for every state, so the automaton needs information to determine which transition to take
      - Thus a stack is required, especially since crawlers will not necessarily be linear in structure
Who Cares?

- A better model for web crawling means we can build better web crawlers, and search the web more effectively.
- Improving web searches is a fundamental goal of Information Retrieval.