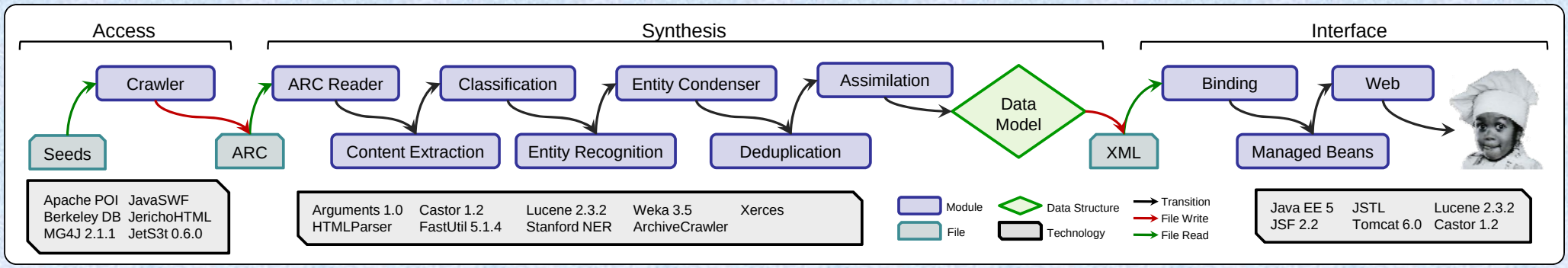
WEBSTER

The Illini-Sports Search Engine

Tim Weninger, Jaeyeon Kihm, Jacob Robinson, Nathaniel Ryckman, Ruiyang Wu, Rui Zhou



Information Access

To access the vast amount of multimodal data available on the World Wide Web we employed the use of the web crawler Heritrix. Heritrix was developed by the Internet Archive, makers of the *Wayback Machine*. Heritrix, written and executed in Java, was chosen because of its robustness and configurability.

We are able to access data from within a multitude of formats. Heritrix allowed for the adaptation of multiple document analyzers e.g. PDF, DNS, Flash, HTML, Word. JavaScript was also natively synthesized by Heritrix so sites that use scripting (AJAX) could be interpreted.

The crawler was run a single time on June 19, 2008. It was allowed to run for 18 hours before being manually terminated. In that time we accumulated over 500Mb of compressed data.

Synthesis

The synthesis stage of the search engine was developed as a standalone Java application that takes as input one or more ARC files from the Access stage and outputs a searchable data model and index.

Web documents, especially HTML pages, are rife with non-content text. We used Weninger's Text-To-Tag Ratio[1] algorithm to clean and extract content text from HTML.

Content Classification

The classification process has two steps. The first is to attempt to classify a document based on textual matches of the sport's name. Furthermore, in bi-gender sports gender clues are utilized to discriminate.

In the likely event a document does not contain the name of the sport then the Explicit Semantic Analysis[3] algorithm is used to classify the document.

Entity Recognition

After a document is classified, entities are found by utilizing Stanford's NER[2] package. Once the entities are extracted they are condensed, and dis-ambiguated textually before being assimilated into the data model.

The data model is arguably the most important part of *Webster*. Most of the work within the Synthesis step involves mapping recognized entities to their proper place in the data model and finding relationships between assimilated entities.

Searching/Ranking

Using the defined data model we can perform searches on the entities. When a query string contains cues to perform an entity search the data model is interrogated for the closest matching entity i.e. "p:Dee Brown" will search for a *Person* "Dee Brown."

This returns a single entity if a direct textual match is found. When no match is found a list of entities is returned ranked by a combination of the Levenstein[4] (edit distance) metric and the number of *mentions*.

Persistence

Lucene's indices are persisted automatically; the data model is marshaled to the file system with Castor.

User Interface

The interface essentially provides a way for users to interact with the data model. Java Server Faces technology is used to provide seamless integration of Java objects with the Web. Using Managed Beans we are able to wire-up our interface to our data model.

The end result is a search-able set of *Super*-pages which provide an aggregation of data, and the relationships therein, on a particular entity or set of entities[5].

Sources

[1] Weninger T., Hsu W. H. "Text Extraction from the Web via Text-To-Tag Ratio", In Proceedings of DEXA'08.
[2] Finkel J.R., Grenager T., Manning C. "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling", In Proceedings of ACL'05. pp. 363-370.
[3] Chang M-W, Ratinov L., Roth D., Srikumar V. "Importance of Semantic Representation: Dataless Classification". AAAI'08.
[4] Levenshtein V. I. "Binary codes capable of correcting deletions, insertions, and reversals", *Soviet Physics Doklady* vol. 10.1966. pp. 707–710.
[5] Luhn H.P., "The automatic creation of literature abstracts", *IBM Journal of Research and Development*. vol. 2. 1958. pp. 159-165.