Zawilinski: Helping Beginning Programmers Conduct MediaWiki-based Research

Zachary Kurmas
Grand Valley State University

Opportunity:

Wikipedia and Wiktionary are excellent sources for undergraduate research projects:

1. Students are familiar with, and like to use these sites.
2. They represent a large, free data set.
3. Students with relatively little programming experience can design and conduct interesting analyses.

Challenge:

Parsing and loading the large MediaWiki dumps presents a "catch 22":

- Stream (e.g., SAX) parsers are efficient, but difficult to use.
- Tree (e.g., "DOM") parsers are easy to use, but require too much memory.

Using a SAX parser requires the researcher to write a significant amount of challenging code.

The standard DOM parser produces an object tree that requires about 10 times as much memory as the XML file.

Our Solution: The *Zawilinski* Library

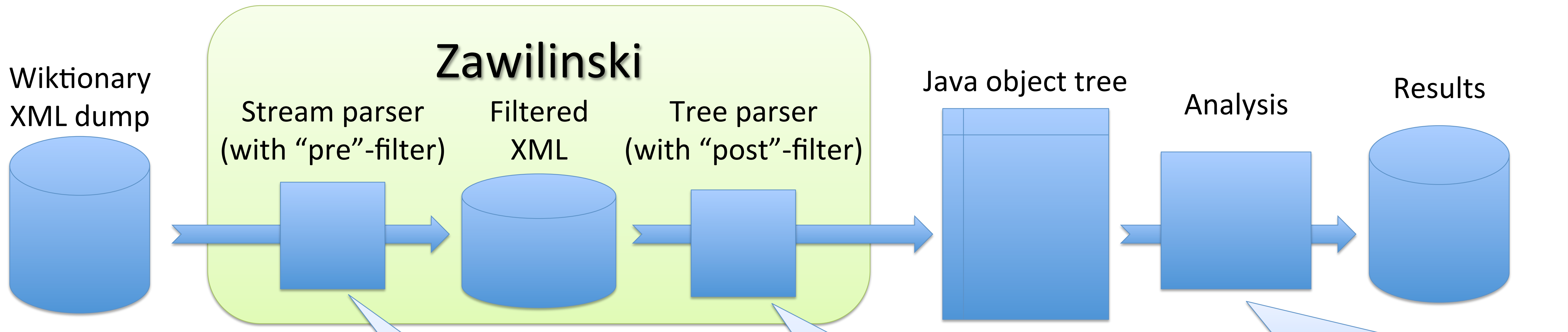
- Provides a Java interface to MediaWiki XML dumps
- Understandable / usable by undergraduates who understand Java interfaces
- Runs on a standard PC. Doesn't require a database.

Removing entire articles here requires buffering XML events.

Key Idea: Use *two* steps to remove irrelevant data from MediaWiki XML dumps

- Many studies focus on a small fraction of articles and/or a small fraction of each article's text.
- A stream-based parser can easily filter unneeded article text; but it cannot easily filter entire articles.
- A tree-based parser can filter unneeded articles; but, can be overwhelmed by articles with a lot of text.
- Zawilinski filters MediaWiki XML dumps in two steps:
  - A "pre-filter" on the stream parser efficiently removes unwanted text, which significantly reduces the work and memory requirements of the tree-parser.
  - A "post-filter" on the tree parser removes unwanted articles, which significantly simplifies the pre-filter code.
- Researchers implement filters by implementing Java interfaces. The library "hides" the complex details of interacting with the parsers.

Example: Polish inflection data in Wiktionary



Code to analyze correctness of inflections

JAXB-based unmarshaller with "post-filter"

- Creates Java objects for each XML element.
- Discards objects without Polish data

Sample XML data

```
<mediawiki>
 <page>
  <title>pies</title><id>57</id>
  <revision>
   <id>7772129</id>
   <timestamp>2009-11-18T20:44:27Z</timestamp>
   <contributor>
    <username>Conrad</username><id>345</id>
   </contributor>
   <text xml:space="preserve">
==English==
===Noun===
'''pies'''# {{plural of|[[pie]]}}
...
==Polish==
===Etymology===
From {{proto|Slavic|pьsъ|lang=pl}}.
===Pronunciation===* {{IPA|/ˈpʲɛs/|lang=pl}}*
 {{audio|Pl-pies.ogg|Audio}}
===Noun===
{{infl|pl|noun|gender=m|animate}}# [[dog]]
====Declension====
{{pl-decl-noun|pies|psy|psa|psów|psu|psom|psa|
  psy|psem|psami|psie|psach|psie|psy}}
...
```

SAX-based "pre-filter"

- Retains all meta-data
  - Small compared to size of article text
  - Don't know whether to keep until text analyzed
- Discards text unrelated to language under study
  - Unnecessary text easy to identify and remove.
  - Most text is unnecessary. Removing it significantly reduces work of tree parser.
- Retains text related to language under study (Polish)

Download from:

- http://www.cis.gvsu.edu/~kurmasz/Software/Zawilinski

Goals:

- Provide students with as little as two semesters of programming experience an opportunity to conduct MediaWiki-based research.
- Research should require only a standard PC with no special configuration.

Our Code

SAX XML Parser

Our Code

DOM XML Parser

Creates Java object for each XML element .

Objects not containing data of interest (e.g., inflection data) are immediately discarded.

Searches each Revision object for a MediaWiki template containing inflection data, then creates a Java object containing that data.

Key Benefits:

•Analyzing changes in inflection data over time requires only 300 additional lines of Java code

•Users can quickly and easily write additional pre- and post- filters to support different analyses of different grammatical data.

word and revision

nom sing.	pies
gen sing.	Psa
Dat sing.	Psu
Acc. Sing.	Psa
Instt. Sing	Psem
Loc. Sing	Psie
Voc sing	Psie
Nom pl	Psy
Gen. pl.	Psów
Dat pl.	Psom
Acc. Pl	Psy

•Creates Java object for each XML element .

•Discards objects without inflection data