



***Inside the
World WaCky Web***

Stefan Evert
University of Osnabrück
stefan.evert@uos.de

Off-the-shelf vs. roll-your-own

Google Linguist BETA

- off-the-shelf tool (or: on-the-Web)
- no setup required
- easy to use
- fast
- tries to be helpful
- knows what you want – better than you know yourself

WaCky DIY toolkit

- toolkit architecture
- components & interfaces
- do your own crawl
- build your own search engine
- full control
- requires some comp. expertise

Reasons for the DIY approach

- my own reason
 - full access to data for statistical models
- pragmatic reasons
 - dependence on commercial service or on unfunded organization
 - will it always be accessible and fast?
 - advertising or “premium content”
 - have they crawled the pages you want?
 - will your personal needs be more important to them than to Google?

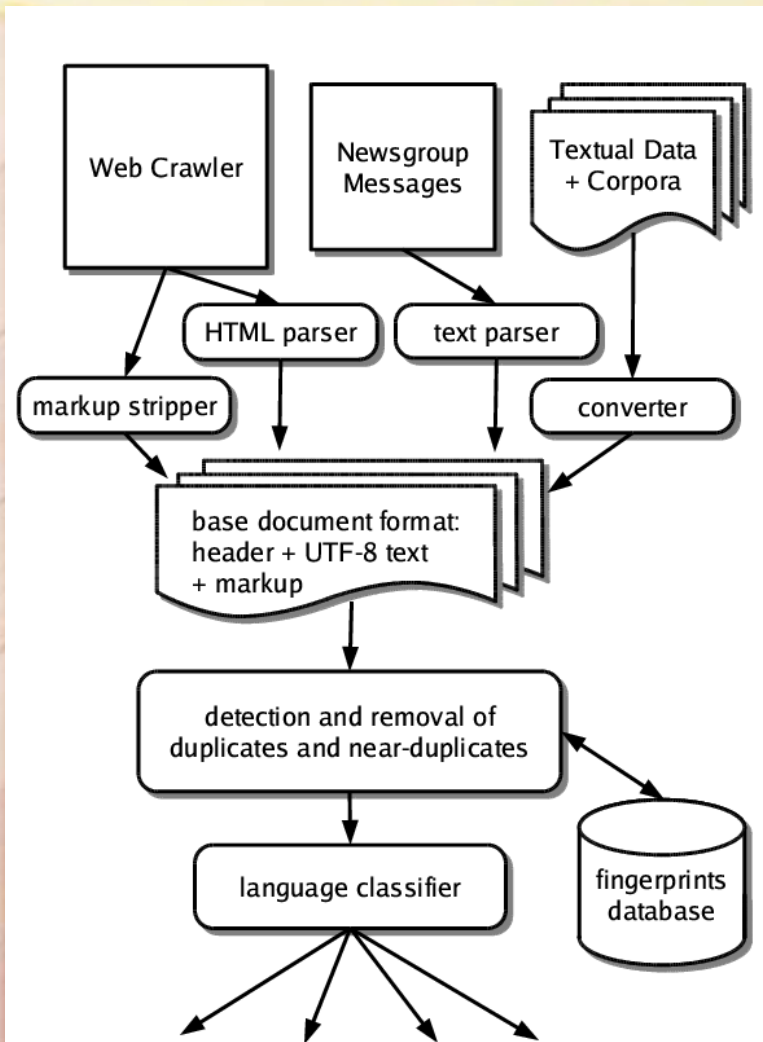
Reasons for the DIY approach

- linguistic quality
 - you may have better annotation tools and resources than Google Linguist
 - especially for minority languages (such as German or Italian)
 - classification of pages (semi-automatic)
- customisation
 - query language (simple but expressive)
 - presentation of results
 - frequency analyses, grouping, ...
 - impossible for Google Linguist to foresee all possible requirements

Reasons for the DIY approach

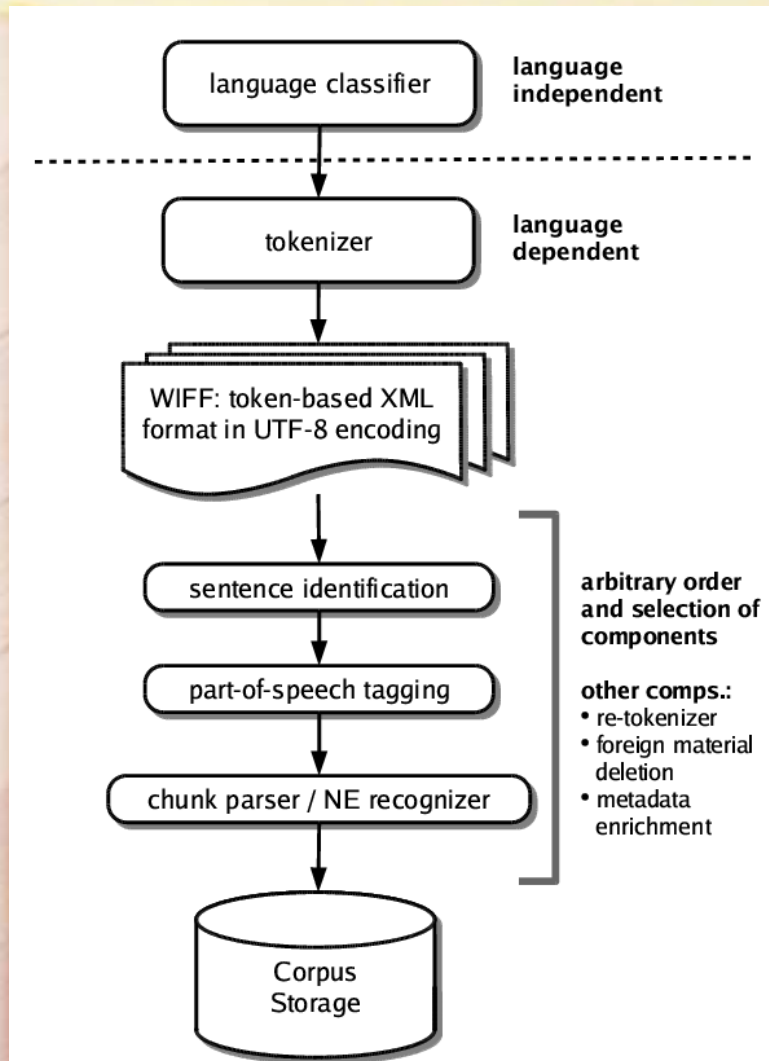
- de-centralise Google Linguist
 - WaCky architecture allows computer-savvy people to set up search engine for local user community
 - moderate computing resources needed
 - can use best annotation tools available
 - non-commercial, can easily be replaced
 - more likely to listen to and satisfy specific user requests

Inside WaCky: what you need



Part 1: the Web crawler


Inside WaCky: what you need



Part 2: linguistic annot.

- insert your own tools & resources
- modules available
- you have to provide the “glue”
- Unix-style “pipe”
- connection: common file format

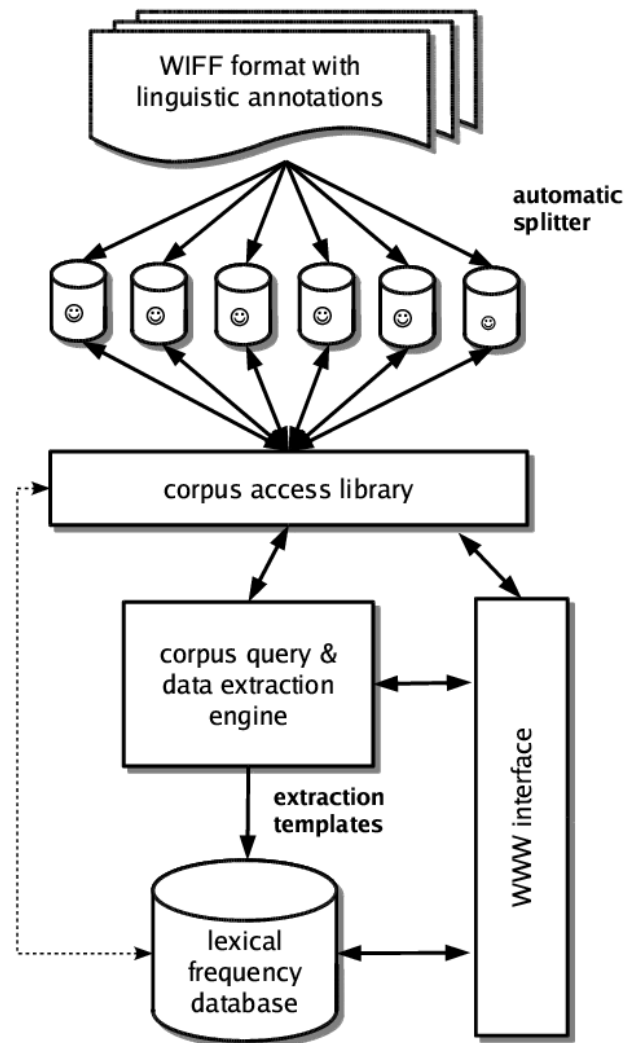
WaCky Interchange File Format



```
# TOKEN word, pos, lemma
# STRUC <p>, <s>, <np>
# ... meta data ...
<p lang="en">
<s>
<np head="Web">
The      DET      the
Web      NE        Web
</np>
is       VV        be
<np head="source">
a        DET      a
rich    ADJ      rich
source  NN        source
</np>
of       PRP      of
...
```

- WaCky Interchange File Format (WIFF)
- version 1 (first year)
- token-based: surface form + annotations
- interspersed with arbitrary XML tags
- file header
 - attribute declarations
 - meta data
- file = Web page(s)
- XML representation

Inside WaCky: what you need



Part 3: indexing, search and interfaces

- corpus indexer
- query engine
- store pre-compiled frequency tables
- united by uniform Web interface

Web interface design

- requirements for good Web interface
 - user accounts (sessions, preferences)
 - customize colours, fonts, layout
 - query history, temporary storage
 - interconnect different components
- opposite of Unix-style “pipes”
 - indexing, search, databases available
 - most components are easy to write
 - combining them is the tricky part
- monolithic interface + plugins
 - choice of programming language

Available software

- Web interface
 - Apache Web server
 - Java servlets or Perl CGI scripts
 - Web publishing frameworks
- pre-compiled frequency tables
 - relational database (MySQL, SQLite)
- indexing & search
 - IMS Corpus Workbench (CWB), Manatee
 - open-source Lucene search engine
 - relational database (commercial)

Indexing & search requirements

- should be open-source & extensible
 - ✓ Lucene, CWB, MySQL, SQLite
- scalable to several billion words
 - ✓ Lucene, commercial database
 - ? MySQL, SQLite, CWB, Manatee
- must allow sophisticated queries that access linguistic annotation
 - ✓ CWB, Manatee
 - Lucene and databases need work
- none of the engines will be immediately useable

The WaCky Challenge

- I propose a WaCky Indexing Technology CHallenge (**WITCH**)
- goals of WITCH
 - identify the most promising engine
 - this engine will be extended as needed
 - clarify WaCky user requirements
 - performance benchmarks: fast enough?
 - understand challenge posed by several billion words with linguistic annotations

How the WITCH works

1. collect tasks ("benchmarks")
 - contributions by user community
 - preferably as concrete examples
 - what to do, how simple, how fast
2. benchmark corpus: 1 billion words
 - one English, one German or Italian
 - rich linguistic annotations (as available)
3. meeting the WITCH
 - each tool adapted by an expert
 - ad-hoc solutions allowed, but must explain potential for generalisation

How the WITCH works

- benchmarking tests
 - on computer at WaCky headquarters (Forlì, Italy)
 - measure speed and memory use
 - speed will not be paramount
- follow the WITCH and participate:

<http://wacky.sslmit.unibo.it/>