Constructionism and the pedagogy of text analysis in the DH classroom

Aris Xanthos

University of Lausanne
Department of language and information sciences

aris.xanthos@unil.ch

Innovative Teaching Methods and Practices in Digital Humanities
(DARIAH workhop at DH 2014)

UNIL | Université de Lausanne

PEDAGOGICAL CONTEXT

Setting: Unil, Faculty of Arts

Domain: computer-assisted text analysis

Audience: undergraduates, no formal training in CS

Format: 1, 2, or 4 semesters depending on concentration

COURSE STRUCTURE: SEMESTER 1

- Basic concepts, methods, tools, and practices:
  - corpus building (planning, acquisition, sampling, legal issues, ...)
  - corpus exploration (segmentation, indexing, concordances, collocations, regular expressions, ...)
  - corpus annotation (XML, TEI, ...)
  - basic quantitative notions (type and token, absolute and relative frequency, document-term matrix, factorial analysis, complexity measures, ...)
- Emphasis on practical work: $\sim 2/3^{rd}$ tutorial sessions and $1/3^{rd}$ lectures

COURSE STRUCTURE: SEMESTER 2

- Students work on small but complete projects (alone or in pairs), usually involving the following steps:
  - formulation of a (simple) research question
  - data collection and preprocessing (incl. annotation)
  - corpus analysis using specialized software (e.g. *AntConc*) as well as more general tools (e.g. *Excel*)
  - oral presentation of results
- Part of the work is carried out in the classroom between oral presentations

COURSE STRUCTURE: 2ND YEAR

- Only for students in a Humanities computing program
- Semester 3: introduction to (Perl) programming for text analysis (text file manipulation, text preprocessing and recoding, text segmentation, frequency counting, ...)
- Semester 4: advanced topics and algorithms (e.g. Markov chains, entropy, random sampling, string edit distance, ...)

PEDAGOGICAL APPROACHES TO TEXT ANALYSIS

- Distinction based on the underlying technologies:
  - either specialized text analysis software (e.g. *Voyant Tools*)
  - or the text processing facilities of a general-purpose programming language (e.g. Perl or Python)
- Illustrations in Hirsch, B.D. (Ed.) (2012). *Digital Humanities Pedagogy: Practices, Principles and Politics*. Cambridge: Open Book Publishers; respectively
  - Sinclair, S. & Rockwell, G., *Teaching Computer-Assisted Text Analysis: Approaches to Learning New Methodologies* (pp.242–263)
  - Ramsay, S., *Programming with Humanists: Reflections on Raising an Army of Hacker-Scholars in the Digital Humanities* (pp.227–239)

Strengths and weaknesses

- Specialized software:
  - \+ productivity (resulting from user-friendly design)
  - – hiding complexity may hinder understanding:

    > *When using text analysis tools, especially those that are interactive, students can likewise arrive at some display or result that they cannot recapitulate and, therefore, they don't really understand. (Sinclair & Rockwell, 2012: 253)*

- General-purpose programming language:
  - \+ getting things done presupposes and/or fosters understanding
  - – slow learning curve

IN SEARCH OF A MIDDLE GROUND

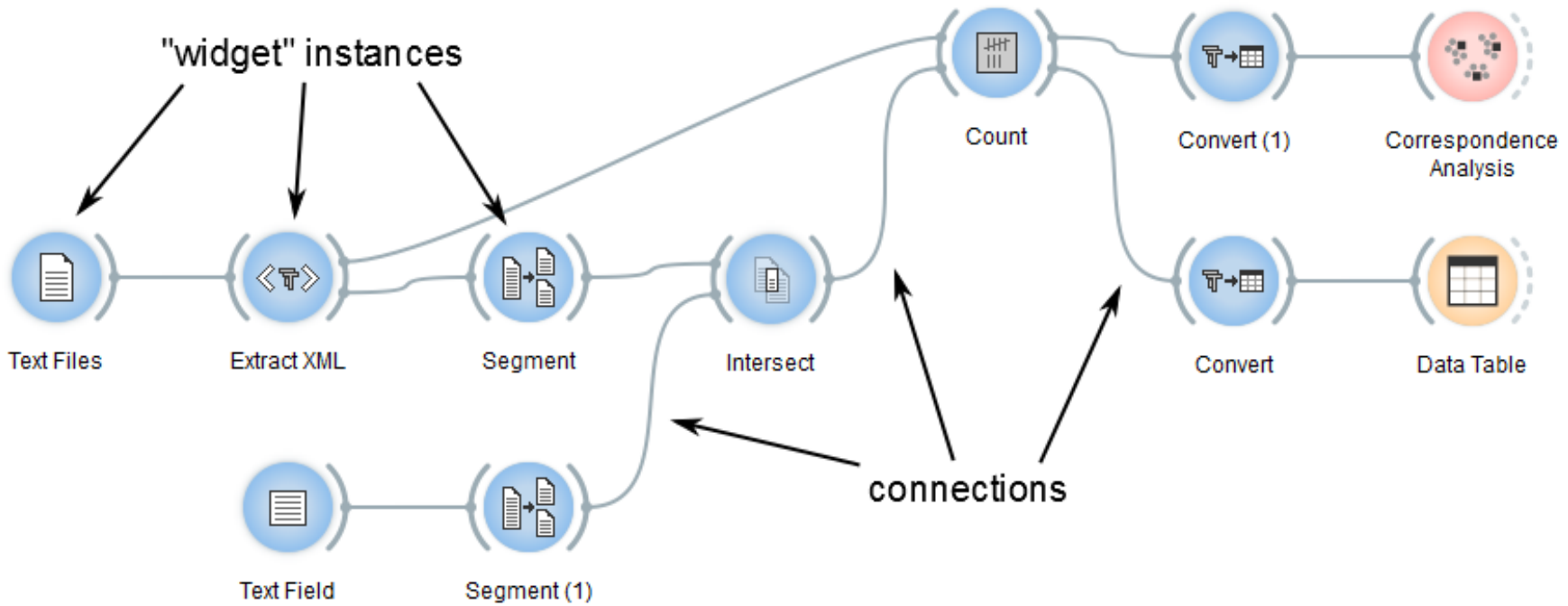
- (Idealized) requirements:
  - engage students in construction-like activities
  - faster learning curve than general-purpose programming
- Steps toward a solution:
  - design a specialized programming environment for text analysis
  - adopt a (mostly) graphical user interface

THE TEXTABLE PROJECT

- Initial subsidy from Unil's Teaching Innovation Fund ("Fonds d'innovation pédagogique = FIP") in 2012
- Public release of *Textable 1.0* in 2012, in the form of an add-on for *Orange Canvas* (open source data mining package)
- Renamed to *Orange Textable* in autumn 2013
- Further funding from the Faculty of Arts, as well as maintenance funding from FIP in late 2013
- Current version is 1.4.2 (released in April 2014)

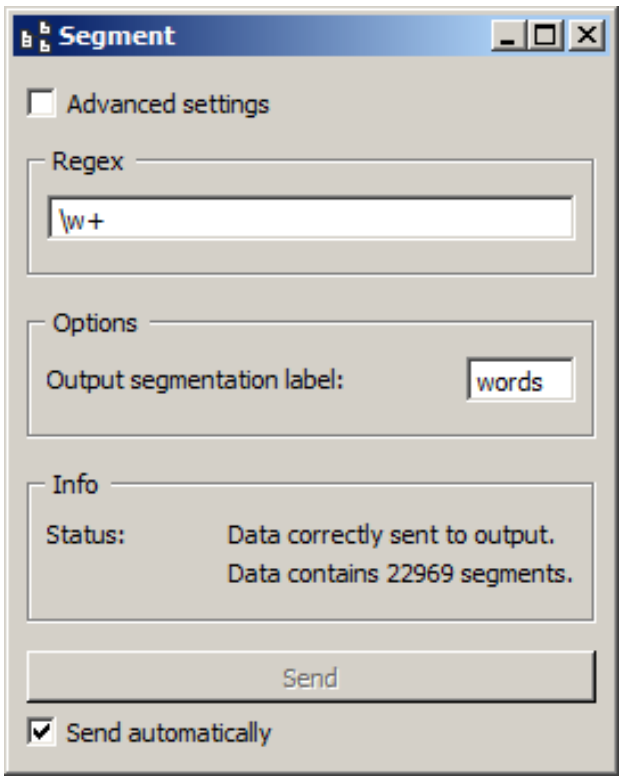
VISUAL PROGRAMMING INTERFACE

The user builds a ”visual program” (called a *schema*) made of interconnected computational units:



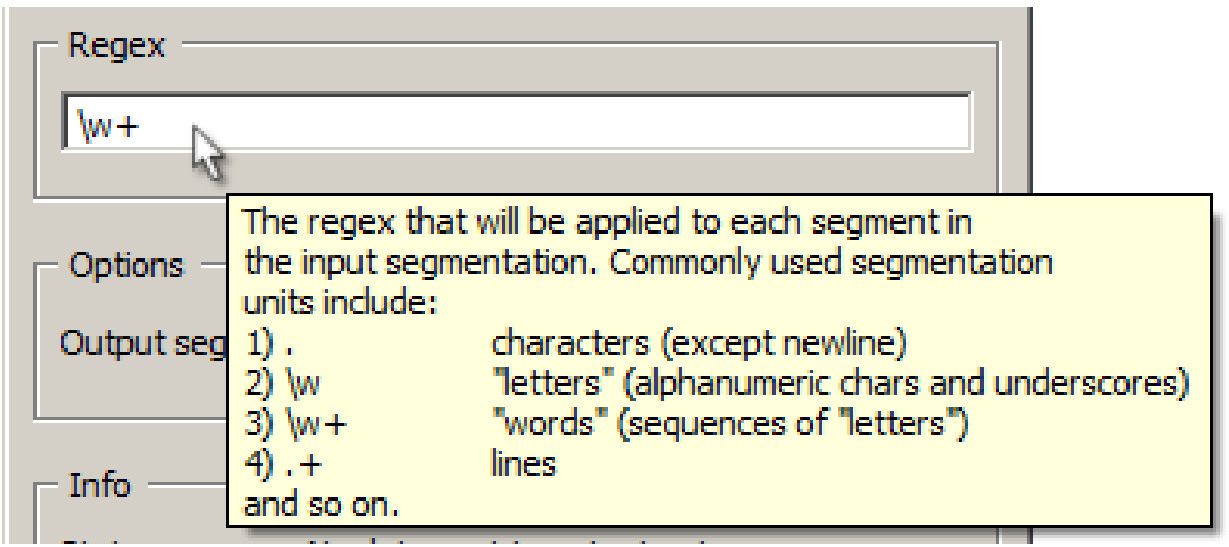
WIDGET CONFIGURATION

Each widget instance can be configured individually, e.g. the *Segment* widget uses a regular expression to specify the units that it attempts to retrieve:



THE MINIMAL AMOUNT OF CODING: REGEXES

- Regular expressions (regexes) are used in the configuration of several widgets (*Segment*, *Recode*, *Select*, ...)
- A very small subset of regular expressions covers most basic needs, as indicated by tooltips:



Looking back on teaching with Textable

- Main tool used for semesters 1 and 2 since 2012–2013
- Actively used by almost all students for their projects
- Projects relying on *Textable* have usually been more ambitious and conducted in a more autonomous fashion
- Students evoke somewhat tedious beginnings until "something clicks" and usage becomes productive and stimulating
- *Textable* has turned out to be very useful as a communication support (for teacher and students alike)

PERSPECTIVES

- How to adapt the 2nd year of my courses given that:
  - *Textable* can't be as powerful than a general-purpose programming language
  - *Textable* makes some results much easier to obtain than with a programming language
- *Textable* is programmed in Python and *Orange Canvas* has a widget for Python scripting
- Re-organize 2nd year around Python scripting within *Orange Canvas* for extending *Textable*

Thank you for your attention!

http://langtech.ch/textable