WP5 - Integration

Ragnar Bonk, Copenhagen Business School

November 27, 2012
Outline

• Brief overview over the tasks
  ➔ 5.1 Specification
  ➔ 5.2 Graphical Interface
  ➔ 5.3 E-pen Interaction
  ➔ 5.4 Logging Functions
  ➔ 5.5 MT Server
  ➔ 5.6 Manual Gaze-to-word Alignment
  ➔ 5.8 Replay Mode for User Activity Data
Outline

• Highlight of some major challenges
• Demo
  ➔ Prototype 1
  ➔ Prototype 2
• Questions and discussion
Specification

• Task 5.1:
  ➔ “Specify the interfaces, data formats and programming languages to allow modular development of components.”
Specification

• Client:
  ➔ HTML 5 and JavaScript
  ➔ jQuery with additional plugins
  ➔ Browser extension(s) for eye tracking (C++)
• Server:
  ➔ Apache 2.x running PHP 5
• Database:
  ➔ MySQL 5
Specification

- System components and architecture
- System API
- Data flow
- Data exchange format (Client ↔ Server):
  - JavaScript Object Notation (JSON)
- Persistence layer (Database table structure)
Already implemented
Graphical Interface

• Task 5.2

  ”Specify and implement the CASMACAT editor including the visualisation and editing options.”
Graphical Interface

• Visual and functional resemblance to commercial CAT tools
• Definition and implementation of shortcuts, e.g.
  ➔ Switch to next segment
  ➔ Copy source to target
  ➔ ...
• Align segments
• Avoid scrollbars
• Use full screen space
Graphical Interface

• Regarding eye tracking:
  ➔ Conservative UI approach chosen
  ➔ Virtual screen concept
  ➔ Same UI look over different browsers
  ➔ Bigger fonts

• There will be an area for features, like
  ➔ Dictionaries
  ➔ Instant messaging
  ➔ …
Graphical Interface

Active source segment

Past target view

Target present edit

Feature area

Source view

Future target view

Virtual screen
Graphical Interface
E-pen Interaction

• Task 5.3
  ➔ ”Specify and implement the e-pen interface and e-pen functions to remove, insert or replace a word or to swap groups of words.”
E-pen Interaction

- Implementation of HTR server finished
  - Supports word recognition
  - No gestures yet
  - No context information yet
  - Uses websockets
  - API similar to MT/IMT server
  - Basic HTML 5 test interface
Logging Functions

• Task 5.4
  ➔ “Specify and implement user activity data logging protocol.”
Logging Functions

• Specification of logged events, e.g.
  ➔ Text changes
  ➔ Keystrokes
  ➔ Eye tracking data
  ➔ Mouse clicks (but not movements)
  ➔ Usage of shortcuts
  ➔ E-pen actions
  ➔ UI events, like resizing the window
Logging Functions

• Data needed from a particular event, e.g.
  ➔ Diff of a text change
  ➔ Current cursor position
  ➔ Character looked at
  ➔ Clicked UI element
  ➔ Selected text
Logging Functions

• Storing and transferring the data
  ➔ Log and upload data in chunks:
    · Store events in a list in the browser
    · If the list exceeds a given limit, upload this chunk to the server
  ➔ Very scalable, keeps performance up
  ➔ Doesn't interfere with user's activity
MT Server

• Task 5.5
  ➔ ”Specify and implement the CASMACAT MT server, including API and database backend.”
MT Server

• Extended Google Translate like API
• Modular design (MT/IMT)
  ➔ First MT prototype implemented in Python → next will be a faster C++ implementation
  ➔ Basic IMT prototype also already implemented
• WebSocket support for fast (live) IMT prediction
• JSON format is compatible with Matecat
• Tokenization on server side
• Word-level confidence measures
• Online training
Manual Gaze-to-word Alignment

• Task 5.6
  ➔ ”Implement a tool to manually correct gaze-to-word alignments.”
Manual Gaze-to-word Alignment

- Gaze data can be unprecise
- First prototype implemented in Translog-II
- 20 files have already been manually corrected
- Same functionality inside CASMACAT

→ Knowledge and algorithms will be transferred

The majority of hunter-gatherer societies are nomadic. It is difficult to be settled under such a subsistence system as the resources of one region can quickly become exhausted. Hunter-gatherer societies also tend to have very low population densities as a result of their subsistence system. Agricultural subsistence systems can support population densities 60 to 100 times greater than land left uncultivated, resulting in denser populations. Hunter-gatherer societies also tend to have non-hierarchical social structures, though this is not always the case. Because hunter-gatherers tend to be nomadic, they generally do not have the possibility to store surplus food. As a result, full-time leaders, bureaucrats, or artisans are rarely supported by hunter-gatherer societies.
Replay mode for User Activity Data

• Task 5.8
  ➔ ”Implement a replay mode of translation sessions.”
Replay mode for User Activity Data

• Realized in the first prototype
  ➔ Visualizes editing activities
  ➔ Visualizes eye tracking data
• Allows for seeking forth and back in time
• Some automatic correction algorithms already implemented
Outline

• Highlight of some major challenges
• Demo
  ➔ Prototype 1
  ➔ Prototype 2
• Questions and discussion
Some of the major challenges

- Integrating eye tracking
  - How to communicate between eye tracker and JavaScript?
  - How to do Gaze-to-character mapping?
  - How to replay correctly on screens with different size/resolution?
Some of the major challenges

- Integrating eye tracking
- Line wrapping still unsolved (cross browser problem)
Some of the major challenges

• Key logging
  ➔ How to log typing correctly for all languages?
  • Text changed event
Questions and discussion

• Thank you for your attention!