Linked Data in Translation-Kits

FEISGILTT Dublin June 2014

Yves Savourel

ENLASO Corporation

Slides and code are available at:
https://copy.com/nGbcO13l9YLs

ENLASO®

This presentation was made possible by mylider

The main points

- What T-Kits users need:

  Terms (with definitions, translations), Context information, disambiguation, Translation examples (concordances)

- Technical challenges
  - Coding the information in or near the content
- Usability challenges
  - Where/when to gather the information
  - How to present the information

Where / When

- **Before extraction**: to annotate the source content. Especially available for content in HTML5 and XML (microdata, RDFa, ITS, etc.)
- **At T-Kit creation time**: to add terminology and provide disambiguation.
- **On-demand**: when the users (translator, editor, localization engineer) need it.

Information in original documents

- Schema.org microdata, ITS Text Analysis, ITS Terminology, RDFa, etc.

+ No need for text analysis: Information can be used directly

- Original annotations may makes text handing more complicated (a lot of inline codes) (never underestimate the amount of pain inline codes can bring)

Information in T-Kits

- Associated by scope (i.e. not linked to a specific span of content). Useful for some data like concepts, glossaries, etc. that can be associated with content later by CAT tools.
- As annotations associated with specific spans of content. This saves some processing to the CAT tools, but may lead to repetitions if not using stand-off notations.

Information queried on-demand

+ End user control what s/he wants
+ Avoid having stale data
- Users may miss on some information
- Requires live and good enough connection
- Tools need to be smart (e.g. query in advance the next segments using another thread)
- Some of the information can be in the cached in the T-Kit.

Technical challenges

- Difference between plain text and marked up text. Many tools take only plain text or HTML. → Need to retrofit the annotations in the target format.
- Overlapping annotations (e.g. `<mrk>` in XLIFF v1.2) can create problem.
- Results not in parsing order, may make applying the information difficult.

Examples

XLIFF 2.0 document with 2 `<file>` elements:

- Extract the “concepts” in a given `<file>` using **AlchemyAPI** service.
- Mark up the segments with ITS Terminology annotations using **Yahoo! Content Analysis**.
- Mark up the segments with ITS Text Analysis annotations and create glossary entries using **DBpedia Spotlight**, **Wikidata** services and **BabelNet** data.

- Load the document
- Create a map of JSONArray objects keyed on `<file>` ids
- For each event in the document:
    - If it is a `START_FILE` event:
        - Reset the block of text, store the id of the `<file>`.
    - If it is a `TEXT_UNIT` event:
        - For each `<segment>`:
            - Append the plain-text content of the segment to the block of text.
    - If it is a `END_FILE` event:
        - **Call the TextGetRankedConcepts service** on the block of text.
        - Store the resulting JSONArray and the corresponding `<file>` id in the map.
- For each `<file>` id in the map:
    - Get the document `MID_FILE` node for the given file id
    - Create a new `<cpt:concepts>` element (extension)
    - Create a new `<cpt:concept>` element for each item in the JSONArray.
- Save the document

Using Yahoo! Content Analysis

- For each `<unit>`:
  - For each `<segment>`:
    - **Call Yahoo! Content Analysis service** on the coded text of the segment.
    - For each entity found:
      - Annotate the fragment using ITS Terminology (with confidence score and, if available, the Wikidata link as termInfoRef).

Using DBpedia Spotlight, Wikidata and BabelNet

- For each `<unit>`:
  - Create a new map of Resource objects
  - For each `<segment>`:
    - Create a new list of Occurrence object
    - **Call the DBpedia Spotlight Candidates service** on the text of the segment.
    - For each candidate found:
      - Create a new Occurrence object (1 occurrence = 1 unique URI on 1 unique span of content).
      - If the Resource for the URI doesn't exist yet: create a new Resource.
    - For each new Resource created:
      - **Do a query on DBpedia SPARQL end-point** for the URI to try to get the corresponding Wikidata Q-value.
      - If we get a Q-value:
        - **Do a wbgetentities GET on Wikidata** to try to get a translation.
      - **Call BabelNet API** to try to get translations from BabelNet dataset.
    - Add the new Resource objects to the unit's Resource list.
    - Sort the list of Occurrence objects for this segment by position in the string.
    - For each Occurrence:
      - Annotate the fragment using ITS Text Analysis and adding extra attributes using extension attributes.
  - For each Resource with translations:
    - Add a glossary entry in the `<unit>`.

Summary

- Technically: It is *relatively* easy to apply linked-data in T-Kits
- But:
  - Information should not get in the way of the end-users tasks (translation, edit, etc.)
  - Most tools in the production chain need to be aware of the metadata annotation mechanisms → need to use standard markup
  - Too much information is no information

A few links

- LIDER Project
  http://lider-project.eu/
- BabelNet
  http://www.babelnet.org/
- ITS 2.0 Specification (W3C ITS IG)
  http://www.w3.org/TR/its20/
- Schema.org microdata (W3C Semantic Web IG)
  http://schema.org/