XGTagger, an open-source interface dealing with XML contents.

Xavier Tannier, Jean-Jacques Girardot and Mihaela Mathieu
Ecole Nationale Supérieure des Mines
158, cours Fauriel
42023 Saint-Etienne FRANCE
tannier, girardot, mathieu@emse.fr

Abstract

This article presents an open-source interface dealing with XML contents and simplifying their analysis. This tool, called XGTagger, allows to use any existing system developed for text only, for any purpose. It takes an XML document in input and creates a new one, adding information brought by the system. We also present the concept of "reading contexts" and show how our tool deals with them.

1. Introduction

XGTagger[1] is a generic interface dealing with text contained by XML documents. It does not perform any analysis by itself, but uses any system S that analyse textual data. It provides S with a text only input. This input is composed of the textual content of the document, taking *reading contexts* into account.

A *reading context* is a part of text, syntactically and semantically self-sufficient, that a person can read in a go, without any interruption [3]. Document-centric XML contents does not necessary reproduce reading contexts in a linear way.

Within this context, we can distinguish three kinds of tags [1]:

- *Soft tags* identify significant parts of a text (mostly emphasis tags, like bold or italic text) but are transparent when reading the text (they do not interrupt the reading context);
- *Jump tags* are used to represent particular elements (margin notes, glosses, etc.). They are detached from the surrounding text and create a new reading context inserted into the existing one.
- Finally *hard tags* are structural tags, they break the linearity of the text (chapters, paragraphs...).

[1] `http://www.emse.fr/~tannier/en/xgtagger.html`

2. General principle

Figure 1 depicts the general functioning scheme of XGTagger. Input XML document is processed and a text is given to the user's system S. After execution of S, a post-processing is performed in order to build a new XML document.

2.1. Input

As shown by figure 1, if a list of soft and jump tags is given by the user, XGTagger recovers the reading contexts, gathers them (separated by dots) and gives the text T to the system S. In the following example `sc` (small capitals) and `bold` are soft tags, since `footnote` is a jump tag.

(1)
```
<article>
    <title>Visit I<sc>stanbul</sc> and M<sc>armara</sc> region</title>
    <par>
        This former capital of three empires<footnote>Istanbul has successively been the capital of Roman, Byzantine and Ottoman empires</footnote> is now the economic capital of <bold>Turkey</bold>
    </par>
</article>
```

Considering soft, jump and hard tags allows XGTagger to recognize terms "Istanbul" and "Marmara", but to distinguish "empires" and "Istanbul" (not separated by a blank character). The text infered is:

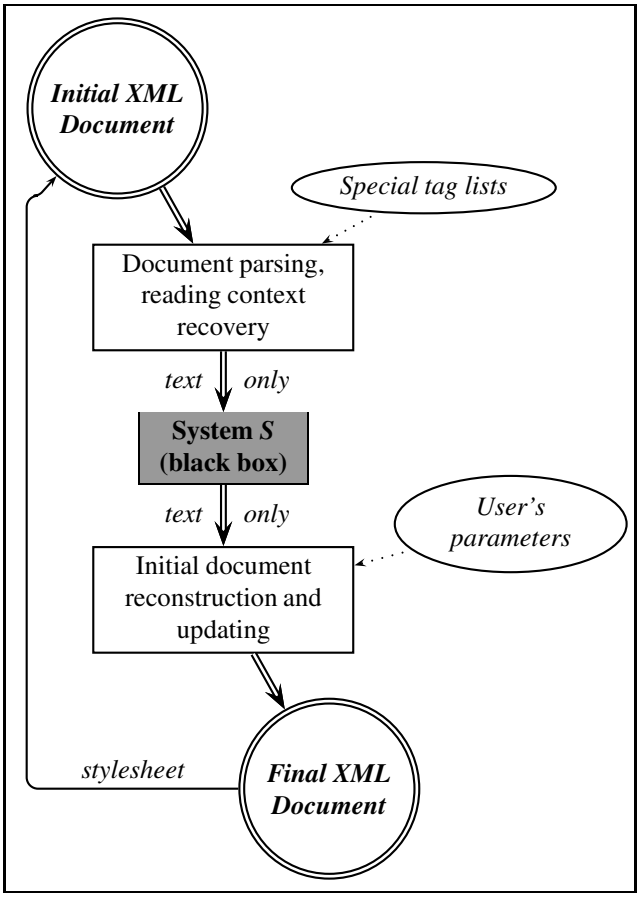

Figure 1. XGTagger general fonctioning scheme.

```
Visit Istanbul and Marmara region .  This
former capital of three empires is now the
economic capital of Turkey .  Istanbul has
successively been the capital of Roman,
Byzantine and Ottoman empires
```

It is not necessary to take care of soft and jump tags if the document or the application do not impose it. If nothing is specified, all tags are considered as hard (in this example, "I" and "stanbul" would have been separated, as well as "M" and "armara" and the footnote would have stayed in the middle of the paragraph). Nevertheless, in applications like natural language processing or indexing, this classification can be very useful.

2.2. Output

This output of the system S must contain (among any other information) the repetition of the input text. If we take the example of POS tagging[2], with TreeTagger [2] standing for the system S, the first field of the output is the initial text. Considering our example, words are separated:

Visit	VV	visit
Istanbul	NP	Istanbul
and	CC	and
Marmara	NP	Marmara
Region	NN	region
.	SENT	.
...	...	...

The user describes S output with parameters[3], allowing XGTagger to compose back the initial XML structure and to represent additional information generated by S with XML attributes. In our running example, parameters should specify that fields are separated by tabulations, that the first field represents the initial word, the second field stands for the part-of-speech (`pos`) and the third one is the lemma (`lem`). XGTagger treats these parameters and S output and returns the following final XML document:

```
<article>
  <title>
    <w id="1" pos="VV" lem="visit">Visit</w>
    <w id="2" pos="NP" lem="Istanbul">I</w>
    <sc>
      <w id="2" pos="NP" lem="Istanbul">
      stanbul</w>
    </sc>
    <w id="3" pos="CC" lem="and">and</w>
    <w id="4" pos="NP" lem="Marmara">M</w>
    <sc>
      <w id="4" pos="NP" lem="Marmara">
      armara</w>
    </sc>
    <w id="5" pos="NN" lem="region">region</w>
  </title>
  <par>
    <w id="7" pos="DT" lem="this">This</w>
    <w id="8" pos="JJ" lem="former">former</w>
    <w id="9" pos="NN" lem="capital">capital</w>
    <w id="10" pos="IN" lem="of">of</w>
    <w id="11" pos="CD" lem="three">three</w>
    <w id="12" pos="NNS" lem="empire">empires</w>
    <footnote>
      <w id="21" pos="NP" lem="Istanbul">
      Istanbul</w>
      <w id="22" pos="VHZ" lem="have">has</w>
      <w id="23" pos="RB"
      lem="successively">successively</w>
      ...
      <w id="32" pos="NP" lem="Ottoman">
      Ottoman</w>
      <w id="33" pos="NNS" lem="empire">
      empires</w>
```

[2] A part-of-speech (POS), or word class, is the role played by a word in the sentence (*e.g.:* noun, verb, adjective...). POS tagging is the process of marking up words in a text with their corresponding roles.

[3] These parameters can be specified either through a configuration file or Unix or DOS-like options (the program is written is Java).

</footnote>
<w id="13" pos="VBZ" lem="be">**is**</w>
<w id="14" pos="RB" lem="now">**now**</w>
<w id="15" pos="DT" lem="the">**the**</w>
<w id="16" pos="JJ" lem="economi">**economic**</w>
<w id="17" pos="NN" lem="capital">**capital**</w>
<w id="18" pos="IN" lem="of">**of**</w>
<bold>
<w id="19" pos="NP" lem="Turkey">**Turkey**</w>
</bold>
</par>
</article>

Note that the identifier `id` allows to keep the reading contexts (see ids 2 and 4, 12 and 13) without any loss of structural information. The initial XML document can be converted back with a simple stylesheet (except for blank characters that S could have added).

More details about XGTagger use and functioning can be found in [4] and in the user manual [5].

3. Examples of uses

The first example was part-of-speech tagging, but any kind of treatments can be performed by system S.

N.B.: Recall that an important constraint of XGTagger is that at least one field of the user system output must contain the initial text (blank characters excepted).

3.1. POS tagging upgrading: locution handling

If the system S is able to detect locutions, XGTagger can deal with that feature, with a special option (called `special separator`). With this option the user can specify that a sequence of characters represents a separation between words.

- Let's take the following XML element:
  <sentence>I did it in order to clarify matters</sentence>
- XGTagger will input the following text into the system:
  `I did it in order to clarify matters`
- With the special separator '`///`', S can return:

I	PP
did	VVD
it	PP
in///order///to	LOC
clarify	VV
matters	NNS

- With appropriate options, XGTagger final output is:
  <sentence>
  <w id="1" pos="PP" t="I">**I**</w>
  <w id="2" pos="VVD" t="do">**did**</w>
  <w id="3" pos="PP" t="it">**it**</w>
  <w id="4" pos="LOC" t="in///order///to">**in**</w>
  <w id="4" pos="LOC" t="in///order///to">**order**</w>
  <w id="4" pos="LOC" t="in///order///to">**to**</w>
  <w id="5" pos="VV" t="clarify">**clarify**</w>
  <w id="6" pos="NNS" t="matter">**matters**</w>
  </sentence>

Note that the three words composing the locution get the same identifier.

3.2. Syntactic analysis

With the same `special separator` option, a syntactic analysis can be performed. Suppose that S groups together noun phrases of the form "`NOUN PREPOSITION NOUN`".

- For the following XML element:
  <english_sentence>He has a taste<gloss>Taste: preference, a strong liking</gloss>
  for danger</english_sentence>
- ...XGTagger will give this text into the system (considering that `'gloss'` is a jump tag):
  `He has a taste for danger . Taste: preference, a strong liking .`
- S can perform a simple syntactic analysis and return, by example:
  `He has a taste_for_danger/NP . Taste: preference, a strong liking .`
- With XGTagger options `-i -w 1 -2 pos -f "/" -d " " -e "_"`, the final output is:
  <english_sentence>
  <w id="1">**He**</w>
  <w id="2">**has**</w>
  <w id="3">**a**</w>
  <w id="4" pos="NP">**taste**</w>
  <gloss>
  <w id="6">**Taste:**</w>
  <w id="7">**preference,**</w>
  ...
  <w id="10">**liking**</w>

</gloss>
<w id="4" pos="NP">**for**</w>
<w id="4" pos="NP">**danger**</w>

</english_sentence>

3.3. Lexical enrichment

The user's system can also return any information about words. For example, a translation of each noun:

- XML Input:
  <sentence>I had a conversation with my brother</sentence>
- S output (suggestion):

```
I
had
a
conversation/entretien/Gespräch
with
my
brother/frère/Bruder
```

- Options: second field is French, third field is German;
  Output:
  <sentence>

<w>**I**</w>
<w>**had**</w>
<w>**a**</w>
<w french="entretien" german="Gespräch">**conversation**</w>
<w>**with**</w>
<w>**my**</w>
<w french="frère" german="Bruder">**brother**</w>

</sentence>

3.4. Reading Contexts finding

Finally, S can just repeat the input text (possibly with a simple separation of punctuation). The result is that words are enclosed between tags, reading contexts are brought together (by ids) and cut words are reassembled. This operation can be particularly interesting for traditional information retrieval; it can represent a first step before indexing XML documents[4] or operating researchs taking logical proximity [3] into account.

- XML Input:
  <title>U<sc>nited</sc> S<sc>tates</sc>
  E<sc>lections</sc></title>

[4] An option of XGTagger adds the path of each element as one of its attribute.

- S output (same as the input):
  `United States Elections`
- Possible final output:
  <title>

<w id="1" rc="United">**U**</w>
<sc>
<w id="1" rc="United">**nited**</w>
</sc>
<w id="2" rc="States">**S**</w>
<sc>
<w id="2" rc="States">**tates**</w>
</sc>
<w id="3" rc="Elections">**E**</w>
<sc>
<w id="3" rc="Elections">**lections**
</w>
</sc>

</title>

4. Conclusion

We have presented XGTagger, a simple software system aimed at simplifying the handling of semi-structured XML documents.

XGTagger allows any tool developed for text-only documents, either in the domain of information retrieval, natural language processing or any document engineering field, to be applied to XML documents.

References

[1] L. Lini, D. Lombardini, M. Paoli, D. Colazzo, and C. Sartiani. XTReSy: A Text Retrieval System for XML documents. In D. Buzzetti, H. Short, and G. Pancalddella, editors, *Augmenting Comprehension: Digital Tools for the History of Ideas*. Office for Humanities Communication Publications, King's College, London, 2001.

[2] H. Schmid. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *International Conference on New Methods in Language Processing*, Sept. 1994.

[3] X. Tannier. Dealing with XML structure through "Reading Contexts". Technical Report 2005-400-007, Ecole Nationale Supérieure des Mines de Saint-Etienne, Apr. 2005.

[4] X. Tannier. XGTagger, a generic interface for analysing XML content. Technical Report 2005-400-008, Ecole Nationale Supérieure des Mines de Saint-Etienne, July 2005.

[5] X. Tannier. XGTagger User Manual. http://www.emse.fr/~tannier/XGTagger/Manual/, June 2005.