Corpus Query System Bonito
Recent Development

Vojtěch Kovář

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
xkovar3@fi.muni.cz

Abstract. This paper presents two of the new features of corpus query system Bonito: 1. Saving the outputs of the system in the XML format and 2. Localization mechanism used to enable easy translation of the system into different languages. In both cases, the developing process is described and examples of the new functionality are given. In the first sections, we also outline the general system functionality and features.

1 Introduction

At the present time, large text corpora form an important source of liguistic information. They are used for a wide variety of tasks, e.g. language learning and teaching, testing of automatic text processing tools, discovering of real words behaviour and many more lingustic research purposes. As the corpus linguistics becomes more and more popular, there is a need of good corpus query systems (CQS) that enable people to work with large text data comfortably. According to the variety of users needs, there are more and more features and functions of these CQSs needed.

At Masaryk University in Brno, a corpus manager Manatee/Bonito [1] is being developed, that is able to perform wide variety of tasks including e.g. fast searching in big corpora, computing word sketches, thesaurus and many more statistical characteristics. The system is used by researchers and lexicographers from all over the world. In order to fulfill different users needs, we continually extend the system by adding new functions.

In this paper, two of these new functions are discussed. Firstly, we briefly describe the Manatee/Bonito system in general. In the next sections the new features – saving outputs in the XML format and localization mechanism of the system – are introduced. We describe the development process of both new features and show examples of the new functionality.

2 The Manatee/Bonito System

The Manatee/Bonito corpus query system consists of two parts.

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2007, pp. 71–76, 2007.

Manatee provides low-level access to corpus data, it integrates fast searching in corpora and evaluation of complex queries implementing a powerful corpus query language. It also functions as a corpus management server.

Bonito serves as an interface between low-level Manatee functions and the user. Version 1 is a standard multiplatform application that connects to the Manatee server and mediates most of its functions in a user-friendly way. The newer version Bonito 2 (see Figure 1) is completely web-based. Web pages are generated on the server (using CGI), with a standard web browser serving as the corpus client.

Bonito 2 is written in Python, object-oriented and very transparent programming language. It enables the system to be well maintained and easily extensible. For generating web pages, a templating engine is used which enables easy changes in web pages appearance.

The functions desribed bellow were implemented within the scope of the Bonito 2 system.

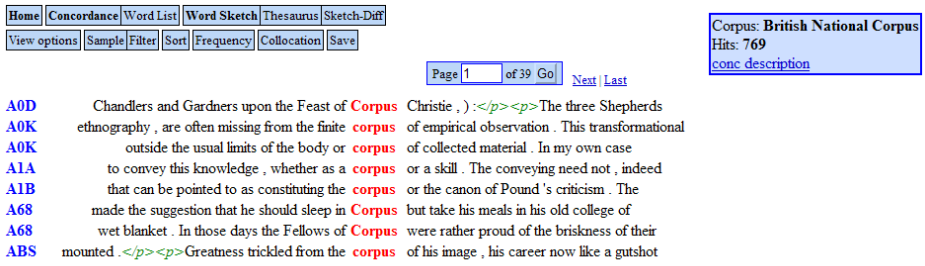

Fig. 1. Illustration of the Bonito 2 user interface

3 Saving Outputs in XML

In the Bonito 2 system, all possible outputs (concordances, word sketches, thesaurus) were in the form of HTML pages. This format is very suitable for viewing the search results but it is not very comfortable e.g. for saving and further processing of the results.

For this reason, we decided to implement two saving possibilities – plain text (in form of columns delimited by the tab character) and XML, that is currently very popular and suitable for further straightforward processing.

As mentioned in the Introduction, the system performs wide variety of tasks. Most important of them are concordance view, word lists, frequency lists, word sketches, thesaurus and collocation candidates computation. For each of these functions, there is an output in form of HTML page, realized by particular template.

```
<concordance>
  <heading>
    <corpus>bnc</corpus>
    <hits>769</hits>
    <query>lc,[word="corpus"|lemma="corpus"] 769 </query>
  </heading>
  <lines>
    <line>
      <ref>A0D</ref>
      <left_context>Chandlers and Gardners upon the Feast of </left_context>
      <kwic> Corpus </kwic>
      <right_context> Christie , ) : The three Shepherds </right_context>
    </line>
    <line>
      <ref>A0K</ref>
      <left_context>ethnography , are often missing from the finite </left_context>
      <kwic> corpus </kwic>
      <right_context> of empirical observation . This transformational </right_context>
    </line>
```

Fig. 2. Concordance – XML structure example

For saving options, we created a new set of templates that is used for text and XML output instead of HTML pages. For each output type, we designed different XML structure. The tags used in the structures are quite simple and self-explaining but they also provide a good structuralization of the data (see Figure 2). For each function, we also created a web form that enables users to modify saving options. An example of "Save Concordance" form can be seen in the Figure 3.

4 Localization Mechanism

The second function we have implemented is the localization mechanism. The main motivation for this step was the fact that the program is used worldwide and for many people the default English version can be non-intuitive and confusing. In the following, we describe all steps leading to well working localization mechanism.

4.1 Templating Engine

As a first step, we changed the used templating engine. The old templating engine was very tiny and simple, but it was not very fast and it also provided only small support for `gettext` utilities that we planned to use for implementing the localization mechanism (see below).

For this reason, we switched to the Cheetah Templating Engine [2], a robust templating engine based on the Python language. It has also quite intuitive

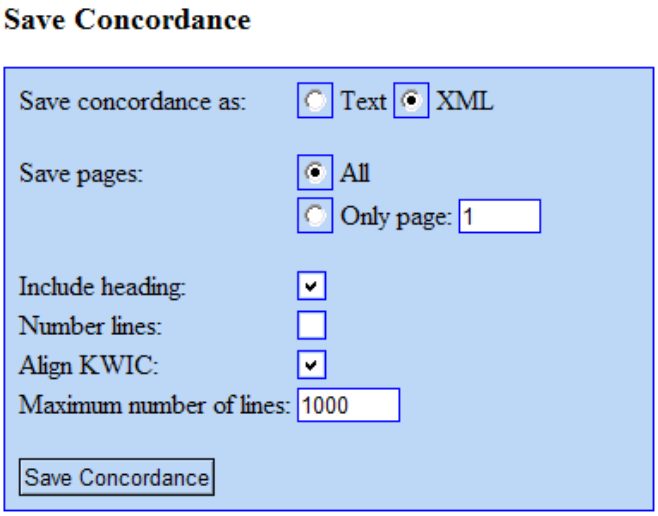

Fig. 3. Save Concordance form

syntax similar to common Python code. All templates used in the system were translated into the Cheetah language.

4.2 The Translation

For the localization itself, we used the `gettext` services integrated in the Python language[1].

In the template files, all translatable strings were replaced by `gettext` statements. By the `gettext` tools, we can now extract all translatable strings from the templates and add next localization language only by adding one file (containing traslated strings) into the system. This is very flexible, so that the system is now easily extensible.

4.3 Language Selection

Another question was how to select the correct user interface (UI) language for particuar user.We solved it by defining two possible ways of how to do that.

By default, the UI language is set according to the preferred language in the user's web browser (we got it by parsing the "HTTP_ACCEPT_LANGUAGE" parameter sent by the browser). The second possibility for the users is to associate their user name with a particular UI language. Currently, both ways are implemented.

[1] `http://docs.python.org/lib/module-gettext.html`

4.4 Input and Output Encoding

When working with corpora in different languages and system with different localizations, there is a question: In what encoding should be the results presented? So far, encoding of the currently selected corpus was used in all system outputs. However, this is useless when using different localizations, e.g. Czech localization could not be used at the same time as an English corpus in ISO Latin 1 encoding.

The only possible solution seems to be using UTF 8 encoding for all outputs. This step brings particular complications, such as recoding of all outputs from selected corpus encoding into UTF 8 and all inputs from UTF 8 to the corpus encoding, but it is the only possibility to assure that the localization will work correctly.

By the input recoding, we also have to handle unknown characters (e.g. when recoding "ž" from UTF 8 into ISO Latin 1). We solved this problem by replacing unknown characters by the fullstop that matches any character in regular expressions used in the corpus query language.

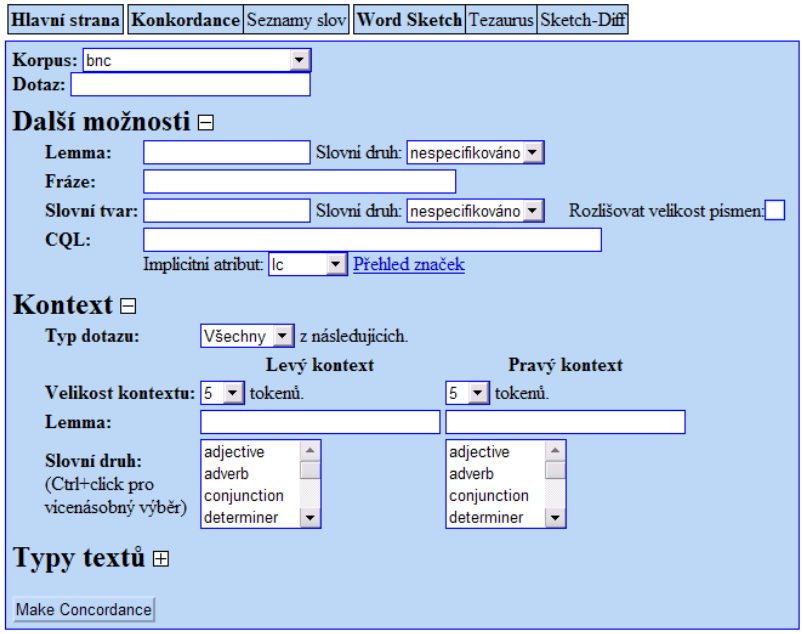

Fig. 4. The main input form in the Czech localization

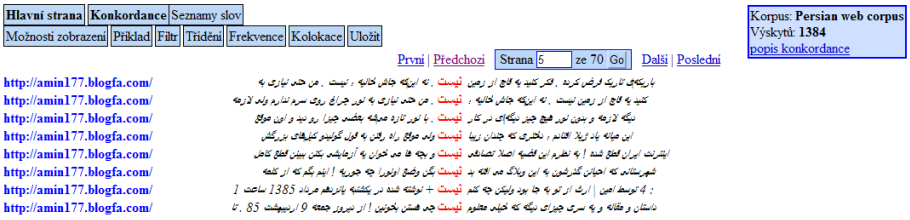

Fig. 5. Persian corpus and Czech localization

4.5 The Czech Localization

A a sample, we created a Czech localization of the system. The user interface in Czech is illustrated in the Figure 4. In the Figure 5, the corpus of Persian is shown within the system with Czech localization.

5 Conclusions and Future Directions

In the paper, we have presented two of recently added features in the Manatee/Bonito corpus query system. We described motivations, development process and some problems related to the implementation as well as their solutions.

In the future development, we want to add more features to enable more comfortable work with the system. Corpora are very valuable source of linguistic information and we want users from all over the world to be able to exploit their benefits.

Acknowledgements

This work has been partly supported by the Academy of Sciences of Czech Republic under the project T100300414.

References

1. Rychlý, P., Smrž, P.: Manatee, Bonito and Word Sketches for Czech. In: Proceedings of the Second International Conference on Corpus Linguisitcs, Saint-Petersburg, Saint-Petersburg State University Press (2004) 124–132.
2. Rudd, T., Orr, M., Bicking, I.: Cheetah: The python-powered template engine (2004) `http://www.cheetahtemplate.org/Py10.html`.