

tasks-schedule

Table of contents

1 Project plan.....	2
2 Tentative division of labour.....	2
3 Schedule.....	3
4 Financing.....	3

1 Project plan

1. As a very first step make HFST transducers work as accepting spellers (i.e., no suggestions) in OpenOffice and MacOS X. OOo integration should be done in Java, building on the Java HFST runtime, whereas the MacOS X integration would be done in C/C++/ObjectiveC, using the C-based runtime as the starting point.
2. Do the same for one other application / interface for each of Java and C. This is done to ensure a certain degree of abstraction and API neutrality in the OLALib design. Possible candidates are: XMLEditor for Java (we need to check the availability of the speller API - is it available in the "free" version, or only in the paid version? other applications could be used instead), and XXX for a C alternative.
3. When the basic accept/reject functionality is in place (as of 1 and 2 above), we can start looking at adding the rest of speller functionality, like suggestions, adding words to the user dictionary, etc.
4. Test the resulting speller, and develop the HFST-based speller technology to ensure that it is of equal or better quality as closed-source alternatives.
5. Do 1-4 for the traditional hyphenator.
6. If/when the resulting speller and hyphenator are good enough, integrate them with MS Office using a company with an established relationship with MS. This will probably result in a second iteration over speller + hyphenator API abstraction, and should make the library more robust for future expansion to other applications and API's.
7. At some point in the future, we want to encapsulate the speller & hyphenator services on the transducer side such that we can abstract away from different implementations and details (i.e. we want both FOMA and HFST as speller backends, and we don't want to exclude any other technology as such, as long as they adhere to this abstraction between the actual language technology components and the OLALib). As part of this step, we also want to define a set of services that is known to the OLALib, and within each service define a set of functionality. Examples of such services can be: traditional speller, hyphenation, text proofing (new style proofing, i.e., both spelling, grammar checking and other types of proofing), grammatical analysis, grammatical editing, etc. This is also the point at which we should start to build out the internal linguistic representation in the OLALib, the so-called LDOM.

2 Tentative division of labour

The Java-based coding could be done by the Java programmer at HU responsible for the Java runtime as well as by Tomi Pieski of the Divvun project. As both are rather occupied at the moment, this coding could be delayed a bit. Another option is to cooperate with the Basque speller project - they are already working on an OOo integration of a FOMA-based speller.

The C-based programming (MacOS X integration + something else) could be the task for Tommi Pirinen, with some support from the Divvun team.

Project coordination and planning could be done by Krister and Sjur, with the kind help of Kimmo and Måns, but this is not yet discussed.

3 Schedule

No detailed schedule is made, but the Divvun goal is to have a working and decent alternative to MS Office spellers (and hyphenators) by the end of 2010, with the main target being tools for OpenOffice.

4 Financing

There might be a possibility to get financing from Langnet. Krister will investigate this.