Hfst speller files

Table of contents

This document specifies the different parts of a speller lexicon file format to be used together with an hfst-based speller engine. It is based on discussions and feedback from June 2010 from potential stakeholders and interested parties.

The specification is open, and the use of it is encouraged whereever hfst-based proofing tools are implemented.

A specification of a hyphenation lexicon file format is deliberately left out at the moment, but will be added in the future, based on feedback and discussions of this specification.

1 Base file format - zip archive

The base file format is a zip archive containing a number of single files that together comprises the full speller lexicon file package. There are several other, widely used file formats that use the same structure, e.g. OpenOffice and the newer MS Office xml file formats.

This format has several benefits: the deliverable is one binary, it is easily extendable with more content without breaking backwards compatibility (at least with some planning ahead), and support for this format should be easy to find both as open-source and closed-source if one doesn't want to do it oneself.

1.1 File naming convention

The default file name for the zip file is:

```
LOCALE.zhfst
```

where

LOCALE
: specifies the locale for which the speller lexicon is meant to be used for, following BCP 47

zhfst
: indicates a zipped collection of (mostly) hfst files

1.2 Exceptions to file naming conventions

There might be application-specific or even plattform-specific reasons to not follow the locale convention above, cases where the plattform or the host application prescribes a different naming conventions. Only when such naming conventions are mandatory should the naming

convention described in this document be deviated from. In all other cases the file naming convention described above should be followed.

2 Zip file content - file naming conentions of contained files

The zip archive should contain the following files:

```
index.xml
LOCALE.hfst
errmodel1.hfst
errmodel2.hfst
...
errmodeln.hfst
```

where

`index.xml`
is a simple xml file (structure described below) giving details about the rest of the zip file content.

`LOCALE.hfst`
is the acceptance transducer; filename as for the zip file, although there are NO exceptions to the filename convention for this file (ie it MUST follow BCP 47). The transducer can either be a one-level or a two-level transducer; if it is a two-level transducer, the other level must produce analyses of the correct words or the accepted suggestions produced by the error models, and the fact that it is a two-level transducer producing analyses must be expressed in the `index.xml` file's description of this file. It is an error to archive an analysing speller transducer without stating so in the `index.xml` file.

`errmodel1.hfst...errmodeln.hfst`
is one or more transducers containing spelling error corrections ("error models") for the speller.

3 Structure of index.xml

The structure of the `index.xml` file can be exemplified by the following xml document for a North Sámi speller (with a simple BCP 47 locale name of just 'se', the ISO 639-1 code for the language):

```
<?xml version="1.0" encoding="UTF-8"?>
<hfstspeller version="1.0">
    <info>
        <title>Davvisámegiela divvunxxx</title>
        <title xml:lang="nn">Nordsamisk stavekontroll</title>
        <title xml:lang="en">North Sámi speller</title>
        <description>[North Sámi description here]</description>
        <description xml:lang="nn">Ein stavekontroll for nordsamisk
        laga av Divvun-prosjektet i samarbeid med Senter for samisk
        språkteknologi, Universitetet i Tromsø</description>
        <version vcsrev="33459">1.5.73</version>
```

```
        <date>2010-09-23</date>
        <producer>Divvun</producer>
        <contact email="divvun@samediggi.no" website="http://www.divvun.no/"/>
    </info>
    <acceptor analyzing="true" id="se.hfst"/>
    <errmodel id="errmodel1.hfst">
        <title>[North Sámi name of error model]</title>
        <title xml:lang="nn">Standardkorrigering</title>
        <title xml:lang="en">Default corrections</title>
        <description xml:lang="nn">Ein korrigeringsmodell balansert mellom rask
        responstid og fleire relevante retteforslag.</description>
        <description xml:lang="en">A correction model balanced between fast
        response times and more relevant correction suggestions.</description>
    </errmodel>
    <errmodel id="errmodel2.hfst">
        <title>[North Sámi name of error model]</title>
        <description>[North Sámi description here]</description>
        <title xml:lang="nn">Hurtigkorrigering</title>
        <description xml:lang="nn">Ein korrigeringsmodell som gjev rask
        responstid og færre forslag.</description>
    </errmodel>
    <errmodel id="errmodel3.hfst">
        <title>[North Sámi name of error model]</title>
        <title xml:lang="nn">Mange skrivefeil</title>
        <description>[North Sámi description here]</description>
        <description xml:lang="nn">Ein langsamare korrigeringsmodell som prøver
        å korrigera fleire skrivefeil i kvart ord.</description>
    </errmodel>
</hfstspeller>
```

This corresponds to the following DTD:

```
<!ELEMENT hfstspeller (info, acceptor, errmodel+)>
<!ATTLIST hfstspeller
  version  CDATA #IMPLIED
  >

<!ELEMENT info (title+, description+, version, date, producer, contact)>

<!ELEMENT title (#PCDATA)>
<!ATTLIST title
  xml:lang NMTOKEN #IMPLIED>
  >

<!ELEMENT description (#PCDATA)>
<!ATTLIST description
  xml:lang NMTOKEN #IMPLIED>
  >

<!ELEMENT version (#PCDATA)>
<!ATTLIST version
  vcsrev  CDATA #IMPLIED
  >

<!ELEMENT date (#PCDATA)>
<!ELEMENT producer (#PCDATA)>

<!ELEMENT contact EMPTY>
<!ATTLIST contact
  email   CDATA #REQUIRED
  website CDATA #REQUIRED
  >

<!ELEMENT acceptor EMPTY>
```

```
<!ATTLIST acceptor
  analyzing (true|false)   "false"
  id CDATA #REQUIRED
  >

<!ELEMENT errmodel (title+, description+)>
<!ATTLIST errmodel
  id CDATA #REQUIRED
  >
```

Most of the elements in this dtd should be straightforward, but a few points should be mentioned:

@version of hfstspeller
this is the version of the specification, can be used for compatibility testing during initialisation

title, description
at least one is required, which corresponds to the locale of the speller lexicon; no xml:lang attribute is required for this one; an unlimited number of additional elements can be added, each with its own unique and required xml:lang attribute. These elements are intended to be presented to the end users if supported by the host application, f.ex. in a user interface to present the speller, or in an interface element where correction models can be chosen.

errmodel
at least one is required; each specified corresponds to a transducer with the filename given in the `id` attribute.

id
is the name of the transducer file in the zip archive

vcsrev
the revision number in the producer's version control system

4 Acceptance transducer

The default acceptance transducer is a one-level automaton, weighted. All complying implementations must support this.

It is also possible to use a two-level, weighted analysing transducer. The analysis output could be used to support concext-based filtering of suggestions (ie only present suggestions with a POS that fits in the given context). Other usage scenarios are also possible.

Support for analysing transducers is optional. When the use of such a transducer is specified in the index.xml file, an implementation that does not support it may choose to flag an error or silently ignore the speller zip archive.

5 Error model transducers

At least one error model transducer must be present. It is an error for the archive not to contain a transducer file named `errmodel1.hfst`.

The behaviour when there are multiple error models included is at present undefined, and subject to feedback from implementors. Possible options are:

- only apply one at a time, at the users choice through a user configuration interface
- apply all the requested error models together, as if they were joined

Supporting multiple error models is optional. An implementation that does not support more than one error model must use the error model named `errmodel1.hfst`, and silently ignore the other error models when opening a speller file containing multiple error models.