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.

The ZIP file must not be encrypted and it must be compatible with to ZIP format version 2.0 as defined by PKWare application note 6.2.0: http://www.pkware.com/support/application-note-archives.

1.1 File naming convention

The default file name for the zip file is:

```
LOCALE-spl.zhfst
```

where

LOCALE
specifies the locale for which the speller lexicon is meant to be used for, following BCP 47; this would also include the use of private use subtags for specifying e.g. domains for specialised spellers, e.g `-x-legal`.

-spl
indicates that this is an archive for spelling purposes

zhfst
indicates a zipped collection of (mostly) hfst files

Examples:

- `se-spl.zhfst` (North Sámi speller lexicon, all geographies)
- `fi-x-legal-spl.zhfst` (Finnish legal terms speller lexicon, all geographies)
- `fi-SE-spl.zhfst` (speller lexicon for Finnish as used in Sweden)

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. The filenaming convention is only a recommendation, but care should be taken to make sure the filename is reasonably informative within the limits of external requirements, and in one way or another convey the information in the three parts above (ie locale, that it is a speller, and that the content is hfst-based).

2 Zip file content - file naming conentions of contained files

The zip archive should contain files as follows (not all required, see below):

```
index.xml
acceptor.default.hfst
acceptor.DESCR.hfst
errmodel.default.hfst
errmodel.ocr.hfst
errmodel.DESCR.hfst
```

where

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

`acceptor.DESCR.hfst`
is one or more acceptor transducers. A 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 provide an analysing speller transducer without stating so in the `index.xml` file.

The filename `acceptor.default.hfst` is reserved for the default acceptor in case there are multiple acceptors (for implementations

or situations where the acceptor to be used can not be determined at runtime).

Additional acceptor files can be analysing acceptors (if the default one is non-analysing), or they can be additional specialised terminology, or whatever seems appropriate in each use case. **DESCR** should be a short, preferably one-word description of the lexicon. If the archive is a distribution of a specialised terminology speller, this transducer may be the only transducer in the archive.

It would also be possible to indicate if an acceptor is analysing by using an `anl.` infix before the filename suffix, like `acceptor.default.anl.hfst`. This option is left open at the moment, and subject to feedback from implementors.

At least one acceptor is REQUIRED.

`errmodel.DESCR.hfst`
is zero or more error model transducers containing spelling error corrections ("error models") for the speller. The reserved filename `errmodel.default.hfst` is to be used as the fallback error model if an implementation is not able to otherwise choose which one to use. OPTIONAL.

The basic idea is that by just using the defaults and the file naming convention described above (ie conveying the intended locale), it would be possible to perform spell checking for the requested language without further work, e.g. without the XML machinery needed to process the `index.xml` file (but that file is still required for an archive to be in accordance with this specification).

The absolute minimal content in a zhfst archive is thus:

```
index.xml
acceptor.hfst -OR- acceptor.DESCR.hfst
```

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 file `se.zhfst` (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" hfstversion="3">
    <info>
        <locale>se</locale>
        <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 type="general" transtype="analyzing" id="acceptor.default.hfst">
        <title>[North Sámi name of acceptor]</title>
        <title xml:lang="nn">Standard stavekontrolleksikon for nordsamisk</title>
        <description>[North Sámi description here]</description>
        <description xml:lang="nn">Eit stavekontrolleksikon som inneheld alle
        vanlege ord og namn. Leksikonet skal dekkja behovet for korrekturlesing
        for vanleg kontorbruk.</description>
    </acceptor>
    <errmodel>
        <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 for tekst typisk
        skrive i tekstbehandlingssystem. Modellen er balansert mellom kravet
        til rask respons og relevante forslag.</description>
        <description xml:lang="en">A correction for text typically entered in
        a text editor. The model is balanced between fast response
        times and more relevant correction suggestions.</description>
        <type type="default"/>
        <model>errormodel.default.hfst</model>
    </errmodel>
    <errmodel>
        <title>[North Sámi name of error model]</title>
        <title xml:lang="nn">OCR-korrigering</title>
        <description>[North Sámi description here]</description>
        <description xml:lang="nn">Ein korrigeringsmodell tilpassa OCR-lesne
        tekstar og typiske OCR-feil. Kan òg brukast på handskriven
        tekst.</description>
        <type type="ocr"/>
        <type type="handwriting"/>
        <model>errormodel.ocr.hfst</model>
        <model>errormodel2.hfst</model>
    </errmodel>
</hfstspeller>
```

This corresponds to the following DTD:

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

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

<!ELEMENT locale (#PCDATA)>

<!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 (title+, description+)>
<!ATTLIST acceptor
  type   ( general | medical | legal | other )   "general"
  trtype (single|analyzing)   "single"
  id     CDATA #REQUIRED
  >

<!ELEMENT errmodel (title+, description+, type+, model+)>
<!ELEMENT type EMPTY>
<!ATTLIST type
  type   ( default | dyslectic | fast | ocr | handwriting ) "default"
  >
<!ELEMENT model (#PCDATA)>
```

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

@hfstversion of hfstspeller
this is the version of the hfst header in the transducer files, can be used for compatibility testing during initialisation. The only version accepted at the moment is version 3, the latest major version of the HFST tools.

locale
the locale of the transducer(s) in the archive, following BCP47. If there is any conflict between locale in the filename of the archive and the value of this element, the value in this element takes precedence.

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.

vcsrev

the revision number in the producer's version control system

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

@type of acceptor
specifies one of a limited number of possible lexicon types. `general` is the default value, `other` should be used for specialised lexicons not covered by other categories in the DTD.

@trtype of acceptor
transducer type, either `single` for one-level transducers (default), or `analyzing` for analysing, two-level transducers.

errmodel
optional; each specified corresponds to a transducer with the filename given in the `model` element.

4 Acceptance transducers

The default acceptance transducer is a weighted, one-level automaton. 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.

There can be more than one accepting transducer in the archive. Usually there should be one general-language transducer, and possibly one or more specialised transducers, but if an archive is meant to just complement an existing installation, only specialised transducers could be included.

The acceptor transducers must include a HFST 3 metadata header.

5 Error model transducers

There can be zero or more error models in the zhfst archive. There is one reserved error model filename: `errmodel.default.hfst` (see further below). The behaviour when there are multiple error models included is at present undefined, and subject to feedback from implementors. Possible options are at least:

- only apply one at a time, at the user's choice through a user configuration interface

- apply a subset of the error models, at the user's choice; all selected error models are applied together, as if they were joined (in the fst sense of the word)
- 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 default error model named `errmodel.default.hfst`, and silently ignore the other error models when opening a speller archive containing multiple error models.

The error model transducers must include a HFST 3 metadata header.