This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644402.

Health in my Language

HimL

D3.2: Common Morphology Interface description

Author(s): Michal Kašpar, Ondřej Bojar, Alexander Fraser, Barry Haddow

Dissemination Level: Public

Date: February 1st 2016

Version 1.0

Grant agreement no.	644402
Project acronym	HimL
Project full title	Health in my Language
Funding Scheme	Innovation Action
Coordinator	Barry Haddow (UEDIN)
Start date, duration	1 February 2015, 36 months
Distribution	Public
Contractual date of delivery	February 1st 2016
Actual date of delivery	February 1st 2016
Deliverable number	D3.2
Deliverable title	Common Morphology Interface description
Type	Report
Status and version	1.0
Number of pages	14
Contributing partners	LINGEA,CUNI,LMU-MUENCHEN,UEDIN
WP leader	LMU-MUENCHEN
Task leader	LMU-MUENCHEN
Authors	Michal Kašpar, Ondřej Bojar, Alexander Fraser, Barry Haddow
EC project officer	Martina Eydner
The Partners in HimL are:	The University of Edinburgh (UEDIN), United Kingdom
	Univerzita Karlova V Praze (CUNI), Czech Republic
	Ludwig-Maximilians-Universitaet Muenchen (LMU-MUENCHEN), Germany
	Lingea SRO (LINGEA), Czech Republic
	NHS 24 (Scotland) (NHS24), United Kingdom
	Cochrane (COCHRANE), United Kingdom

For copies or reports, updates on project activities and other HimL-related information, contact:

Barry Haddow bhaddow@staffmail.ed.ac.uk
University of Edinburgh Phone: +44 (0) 131 651 3173

Contents

1 Introduction

Two of the research work packages of HimL (WP2 Semantics and WP3 Morphology) rely on morphological processing of input sentences (English) and more importantly of the target side (Czech, German, Polish, Romanian). Training data in both the source and the target languages have to be morphologically analyzed prior to the extraction of translation equivalents and some of our system configurations need also morphological generation for the target language. LMU and CUNI have long experience and custom tools for processing German and Czech, respectively. Lingea provides its morphological processing for all HimL languages.

Techniques for improving translation quality will be developed on translations from English to German and Czech and then ported to Romanian and Polish. To simplify this transfer, we decided to use a common morphology interface, which abstracts from the implementation details of particular morphology toolkits and provides morphology information in a consistent way for any processed language. For now, we are aiming at English, German, Czech, Polish and Romanian, but during further exploitation after the project end, we should aim at a much wider range of languages. This should be possible because we have representatives of Germanic, Slavic, and Romance languages covered. If there were many languages with many tools with different interfaces, one would have to simultaneously change the interface used to get morphology information and the language dependent way how this information is used, which would probably lead to many errors caused by the complexity of such a modification. Using common morphology interface effectively splits this task into two easier and well defined subtasks – (1) use the new morphology engine and (2) adapt the language-specific behaviour to a new language – while the interface remains the same.

To enable further consistent use for new languages and purposes, we base our Common morphology interface on a formalism that is capable of processing much finer morphological information and for many more languages than we currently need, the Interset (Zeman, 2008).

To avoid the unnecessary burden of implementation of all Interset features, we selected only a subset of morphology features, which we believe is important for our task, and also may be expected from a reasonably well developed morphology engine if a new language were added. Anyone who needs some of the remaining Interset features can find them in the upstream formalism, implement and use them without breaking any interface compatibility for others.

By publishing our interface description under a non-restrictive open license, we also want to encourage its broader use in the development of new morphology-enabled technologies and their further porting to other natural languages. The interface description is based on a feature set developed at CUNI for translating tagsets among languages and annotation styles, so we can suppose that the base formalism already has well-designed features for an accurate representation of morphological information of many languages and many existing tagsets and it also lends itself to extensions when necessary. The interface was reviewed at Lingea, who have the resources to implement it for more than 20 languages at least to the level of ambiguous lemmatization, ambiguous tagging and surface form generation and provide such tools even for commercial solutions. The interface can be freely reimplemented by anyone without requiring anyone's permission and it is designed to make this really simple. We also consider this a step towards the goals of the Cracking the language barrier initiative[1] in which our project takes part.

1.1 Data format over API

The three main contributors to this task, Lingea, LMU and CUNI, have discussed their best practices with handling morphologically annotated data. The partners are also supposed to use the morphology interface in both experimental and final HimL systems.

It turned out that the processing of translation is done using very heterogenous pieces of software which communicate usualy in one way by means of Posix pipes or files or across the network, sending queries and receiving replies. Keeping this in mind, we decided that best way of integrating morphology tools are standalone tools that use fixed file formats. Moreover, while Lingea has a single and stable codebase (systems are designed for production), which could be easily extended with the needed API, the two research partners have to deal with a relatively large set of programming languages and with tools consisting of several loosely-coupled components. Implementing an API in a single or at least very few programming languages would unnecessarily limit the experimentation options at the two research sites.

The three partners thus agreed to define the common interface in terms of a common data format, which can be easily read and written in all supported languages. In the following, we describe the format in detail.

1.2 Scope of the representation detail

Before designing a representation for morphological information in multiple languages, we have to choose the level of abstraction. A critical question is whether we want to build new words from known parts (word formation or derivation), or just change

[1] http://www.cracking-the-language-barrier.eu/

the morphological features of known words (word inflection).

For now, we do not interfere with creating new words, because it would involve rather substantial language-specific complexity – we would have to know the parts that should be used for formation of new words with desired meaning and the way of combining them.

1.3 Overview of this document

First, we introduce the morphological features expected to be implemented for HimL project use, to have some basis for examples (Section 2). We then continue with the notation for morphological tags (Section 3), define which form of a word is considered to be the lemma (Section 4), and then proceed to data formats (Section 5) used for communication between our systems. At the end, we describe API for C which may be used also in other programming languages (Section 6).

2 Morphological features

In HimL, we need this interface for Romanian, Polish, Czech, German and English, so we also describe the level of information supposed to be processed for these languages not giving any strict requirements to enable employing different toolkits underneath this interface for different purposes. To enable consistent growing support for a broad range of languages from various sources, we use and encourage the use of features described by Interset (Zeman, 2008) wherever possible. The following tables describe the union of subsets of Interset features and their expected values that are supposed to be implemented for individual HimL languages. Some of them are needed just in one language, some of them are needed in multiple languages. For extensions, please refer to full Interset description[2]

2.1 Part of speech – pos

The part of speech is a common feature attributed to all words. It thus appears in all morphological tags.

The value of the part of speech also indicates how the rest of the features should be processed.

Value	Description
noun	noun
adj	adjective
num	numeral (cardinal number)
verb	verb
adv	adverb
adp	adposition (preposition, postposition or circumposition)
conj	conjunction
part	particle
int	interjection
punc	punctuation
sym	symbol

2.2 Pronoun-like words – prontype

For pronouns, our tagset defines a finer classification (a kind of a “subpart of speech”) called “prontype”.

An empty value of prontype means that the feature is not applicable because this is not a pronoun. So for pronouns, this feature is obligatory. When the exact value is unknown, use ‘prn’.

[2] https://wiki.ufal.ms.mff.cuni.cz/user:zeman:interset:features

Value	Description
prn	The word is pronominal (or determiner) but we do not know the exact type.
prs	Personal or possessive pronoun. Possessives are recognizable by the value of their poss feature. Reflexive pronouns are distinguished from normal personal/possessive pronouns by the value of their reflex feature.
art	Article, i.e. determiner bearing only the feature of definiteness or indefinitess and nothing more (English "a", "an", "the", German "der", "die", "das", Portuguese "um", "uma", "o", "a", "os", "as").
int	Interrogative pronoun / determiner / adverb ("who", "what", "which").
rel	Relative pronoun / determiner / adverb. Many interrogative pronouns in many languages can also be used as relative pronouns. However, in some languages there are pronouns that fall in one of the categories but not both (Czech "jenž" is only relative; in Bulgarian, relatives are completely separated from interrogatives). For words that can be both interrogative and relative, "int" is the default value.
dem	Demonstrative pronoun / determiner / adverb ("this", "that"). Being a demonstrative pronoun is not the same as being definite (definiteness=def), although the two feature-values are similar.
neg	Negative pronoun / determiner / adverb ("nobody, nothing, none"). This is not the same as the negativeness feature. Unlike e.g. negative and positive adjectives or verbs, negative pronouns are not complements of some "positive" pronouns. Instead, they usually correspond to zero, nothing.
ind	Indefinite pronoun / determiner / adverb ("somebody", "something", "anybody", "anything"). Being an indefinite pronoun is not the same as being morphologically indefinite (definiteness=ind).
tot	Total (universal) pronoun / determiner / adverb ("everybody", "everything")

2.3 Type of numerals – numtype

Value	Description
card	cardinal number
ord	ordinal number
mult	multiplier number (Czech "pětkrát" – "five times")
frac	fraction ("pětina" – "one fifth")
gen	generic numeral ("twofold", Czech "jedny", "čtvero", "čtverý")
sets	number of sets of things, or of pluralia tantum (Czech "čtvery")

2.4 Writing of numerals – numform

Value	Description
word	numeral word ("fourteen")
digit	number written using digits ("14")
roman	number written using Roman numerals ("XIV")

2.5 Types of adverbs – advtype

Value	Description
man	adverb of manner ("how")
loc	adverb of location ("where")
tim	adverb of time ("when")
deg	adverb of quantity or degree ("kolik")
cau	adverb of cause ("why")
mod	adverb of modal nature (Czech "možno", "nutno", "radno", "třeba")
sta	adverb of state (Czech "zima", "volno")

2.6 Types of adpositions – adpostype

Value	Description
prep	preposition ("in", "on", "to", "from")
post	postposition (German "entlang" in "der Strasse entlang")
circ	circumposition (German "von ... an" in "von dieser Stelle an")
voc	vocalized preposition (Czech "ve" as opposed to base form "v")

2.7 Types of conjunctions – conjtype

Value	Description
coor	coordinating conjunction
sub	subordinating conjunction

2.8 Word is posessive adjective or pronoun – poss

If word is not possessive, this feature is not present.

Value	Description
poss	possessive

2.9 Word is reflexive – reflex

If word is not reflexive, this feature is not present.

Value	Description
reflex	reflexive

2.10 Word is negative – negativeness

Value	Description
pos	positive, affirmative
neg	negative

2.11 Definiteness of word – definiteness

Value	Description
ind	indefinite
def	definite

2.12 Gender – gender

Value	Description
masc	masculine
fem	feminine
com	common, utrum
neut	neuter

2.13 Posgender – posgender

Possessor gender if apropriate.

Value	Description
masc	masculine
fem	feminine
com	common, utrum
neut	neuter

2.14 Animateness if known – animateness

Value	Description
anim	animate
nhum	animate but not human
inan	inanimate

2.15 Grammatical number – number

Value	Description
sing	singular
dual	dual
plur	plural
ptan	plurale tantum
coll	collective / mass / singulare tantum

2.16 Number of possessors – possnumber

Value	Description
sing	singular
dual	dual
plur	plural

2.17 Number of possessed items – possednumber

Value	Description
sing	singular
dual	dual
plur	plural

2.18 Grammatical case – case

Value	Name	Description
nom	nominative	cs: dům, budova = a house, building
gen	genitive	cs: domu, budovy = of a house; in Basque, this is possessive genitive (as opposed to locative genitive): diktadorearen erregimena = dictator's regime (diktadore = dictator)
dat	dative	cs: domu, budově = to a house
acc	accusative or oblique	cs: dům, budovu = a house
voc	vocative	cs: dome, budovo = hey, you house!
loc	locative	cs: v domě, budově = in a house; used also for locative genitive (as opposed to possessive genitive) in Basque: talde anarkistako = group of anarchists
ins	instrumental / instructive	cs: domem, budovou = with/through/using/by a house.

2.19 Gradation – degree

Value	Description
pos	positive, first degree (note that although this degree is traditionally called "positive", negative properties can be compared, too)
cmp	comparative, second degree
sup	superlative, third degree
abs	absolute superlative

2.20 Person – person

Used for verbs and possessive pronouns. Means possessors person.

Value	Description
1	first (I, we)
2	second (you)
3	third (he, she, it, they)

2.21 Possessors person – possperson

Not to be used with possessive pronouns.

Value	Description
1	first (I, we)
2	second (you)
3	third (he, she, it, they)

2.22 Politeness forms – politeness

Value	Description
inf	informal (Czech "ty/vy", German "du/ihr", Spanish "tú/vosotros")
pol	polite (Czech "vy", German "Sie", Spanish "usted")

2.23 Type of verbform – verbform

Value	Description
fin	finite
inf	infinitive
part	participle (present ("doing"), past ("done"), passive (Czech "udělán" distinguished from adjective "udělaný" by variant=short)), gerundive
trans	transgressive, adverbial participle (modifies other verbs, behaves like adverb; Czech present "dělaje", past "udělav")
ger	gerund (verbal noun). Latin gerundium: "amare" ⇒ genitive "amandi", dative "amando", accusative "(ad) amandum", ablative "amando".

2.24 Mood of verbs – mood

Value	Description
ind	indicative
imp	imperative
cnd	conditional
sub	subjunctive (conjunctive) (spojovací)
jus	jussive (optative) (přací)

2.25 Tense – tense

Value	Description
past	past
pres	present
fut	future
aor	aorist
imp	imperfect
pqp	pluperfect

2.26 Aspect – aspect

Value	Description
imp	imperfect
perf	perfect
prog	progressive

2.27 Voice – voice

Value	Description
act	active
pass	passive

2.28 Whether word is abbreviated – abbr

Value	Description
abbr	abbreviation

2.29 Form variant – variant

Variants are different surface forms with exactly the same values of every feature except 'variant'. Short and long variants are typicaly used in different contexts. This may occur for example with Czech 'mi'/'mně'. Numbered values are for other purposes.

Value	Description
short	short form
long	long form
0	variant form 0
1	variant form 1
2	variant form 2
3	variant form 3
4	variant form 4
5	variant form 5
6	variant form 6
7	variant form 7
8	variant form 8
9	variant form 9

3 Morphological tags

Morphological information in Interset is like a map from feature names to feature values. For low level purposes and serialization of tags, we will define tags based on Interset as a sequence of pairs, where the first element in the pair is a feature name and the second is the value of the feature. The feature name and feature value are delimited by a colon ':'. A sequence of feature-value pairs is glued together by semicolons ';'. Features are required to be sorted in alphabetical order, to enable their comparison using string comparison. When several alternative tags have to be recorded together, they are delimited by slashes '/', again in alphabetical order.

4 Lemma

The lemma is considered the base form of a word. It however happens sometimes that it is not quite clear which form of a word should get this prominence of being the base form. For instance, we could make several obvious steps to simplify the word form further, and get word form with completely different features, for example with a different part of speech. Sometimes, such a sequence of "natural simplifications" could even become circular.

We take a pragmatic approach to this. Our goal is to process text in a way as simple as possible, so we decided to preserve the part of speech in most cases and otherwise generalize as far as possible. The only exception so far are possesives, which should be generalized to words describing the owner, for example 'matčino' → 'matka'.

5 Data formats

In this section, we define the input and output formats for individual tools we need.

5.1 Tokenization

The expected input of tokenization is plain text which can contain any whitespace between words. The tokenizer should break such text into meaningful tokens which means:

- Preserve newlines. (These may be used as sentence delimiters. If we intend to replace ends of lines by space, we can do this before tokenization.)
- Replace any sequence of non-newline whitespace by a single space.

- Separate punctuation from words except where punctuation is part of the word – e.g. dots of abbreviations.

5.2 Ambiguous lemmatization, ambiguous tagging

For these tasks, the input may be either

- vertical – One token per line, sentences are delimited by empty lines.
- plain text – Tokenization will be done using knowledge of morphology. Each line is expected to contain exactly one sentence.

Disambiguated output may be:

- vertical – Columns containing original form, lemma and tag are tab delimited. Alternative lemmas and tags are delimited by slash '/', n^{th} lemma corresponds to n^{th} tag.
- factored – Newlines from input are preserved. Word, lemma and tag are delimited by pipes: word|lemma|tag. Alternative lemmas and tags are delimited by slash '/', n^{th} lemma corresponds to n^{th} tag – for example: "jedu|jet/jed/jed/jed|aspect:imp;mood:ind;negativeness:pos;number:sing;person:1;pos:verb;tense:pres;voice:act/ case:gen;gender:masc;number:sing;pos:noun/case:dat;gender:masc;number:sing;pos:noun/ case:loc;gender:masc;number:sing;pos:noun" (linebreaks inside tag would not be present in real data)

5.3 Lemmatization, tagging

For these tasks, the input may be either

- vertical – One token per line, sentences are delimited by empty lines. In this format, the tokenization is defined by the user and it will be preserved by our tools at the risk of minor loss in coverage due to a possible incompatibility of the tokenization styles.
- plain text – Each line is expected to contain exactly one sentence. The tokenization will be carried out by our tools, which has the benefit of guaranteed match with the tokenization rules of the underlying morphological dictionaries or processing tools.

The disambiguated output may be:

- vertical – Columns containing the original form and the assigned lemma and tag are delimited with a tab.
- factored – The newlines from the input are preserved. Word, lemma and tag are delimited by the pipe character: word|lemma|tag – for example "jdu|jít|aspect:imp;mood:ind;negativeness:pos;number:sing;person:1;pos:verb;tense:pres;voice:act".

5.4 Surface form generation

The surface form generation expects the information about the lemma or other word form, which should be used as base, and the tag which describes the desired morphological features. This can be encoded as:

- vertical – Columns containing some known form, lemma and tag are tab-delimited.
- factored – Newlines from input are preserved. Known form, lemma and tag are delimited by pipes: word|lemma|tag If either word or lemma is not known, it can be replaced by an underscore '_', but only one of them may be replaced.

Sample input in vertical format:

```
šel jít aspect:imp;mood:ind;negativeness:pos;number:sing;person:1;pos:verb;tense:pres;voice:act
domů domů advtype:loc;pos:adv
. . pos:punc

rychle rychle advtype:man;pos:adv
. . pos:punc
```

Sample input in factored format:

```
domů|domů|advtype:loc;pos:adv .|.|pos:punc
rychle|rychle|advtype:man;pos:adv .|.|pos:punc
```

The output of form generation can be:

- simple vertical – One generated word per line, empty lines denote sentence boundaries.
- full vertical – Columns containing generated word, lemma and tag delimited by tabs, empty line denotes sentence boundary.
- tokenized – Tokenized text consisting of the generated words.
- factored – Tokenized text consisting of the generated words, their lemmas and tags. Tokens are delimited by spaces or newlines. The generated form, lemma and tag are glued by pipes.

Sample output in simple vertical format:

```
jdu
domů
.

rychle
.
```

Sample output in full vertical format:

```
jdu jít aspect:imp;mood:ind;negativeness:pos;number:sing;person:1;pos:verb;tense:pres;voice:act
domů domů advtype:loc;pos:adv
. . pos:punc

rychle rychle advtype:man;pos:adv
. . pos:punc
```

Sample output in tokenized format:

```
jdu domů .
rychle .
```

Sample output in factored format:

```
domů|domů|advtype:loc;pos:adv .|.|pos:punc
rychle|rychle|advtype:man;pos:adv .|.|pos:punc
```

6 APIs for programming languages

Common morphology interface should treat each functionality as a separate tool which is initialized and used independently of the other tools. Initialization functions should be used to provide any needed configuration data to the implementation and probably will be dependent on implementation. Functions performing the core work of each functionality have to behave in a uniform consistent way. Calling of cleanup functions should be also implementation independent. Data type used for morphology information should allow arbitrary features and values, so a string to string map is prefect implementation, string with tag is acceptable implementation.

Expected lifecycle of common morphology tool implementing `functionality`:

```
handle = cmi_init_functionality(init_data);
while(!done)
{
  result = cmi_functionality(handle, arguments);
  //use result
}
cmi_free_functionality(handle);
```

In object oriented languages, handle should be an object with method `functionality` and destructor, result may be also an object. Functionality is one of: `tokenize`, `analyze`, `tag`, `generate`. For C language we declare these functions (which should be used from other programming languages):

Parameter `option` may cause implementation dependent different behaviour, setting it to `NULL` causes standard behaviour.

```
long cmi_tokenize(void *handle, const char *text, long startPosition, long endPosition,
        void *options = NULL)
```

Returns first token boundary after `startPosition` and before `endPosition`.

```
int cmi_analyze(void *handle, const char *text, long startPosition, long endPosition,
        char *lemmaBuffer, long lemmaBufferLength, char *tagBuffer, long tagBufferLength,
        void *options = NULL)
```

Returns count of different analyses or -1 if any buffer is too short. Lemmas are copied to `lemmaBuffer` and delimited by zero bytes, tags are copied to `tagBuffer` and delimited by zero bytes. n^{th} lemma corresponds to n^{th} tag.

```
int cmi_tag(void *handle, void **context; const char *text, long startPosition, long endPosition,
        char *lemmaBuffer, long lemmaBufferLength, char *tagBuffer, long tagBufferLength,
        void *options = NULL)
```

Returns count of different analyses or -1 if any buffer is too short. For first call on the same buffer `*context` must be `NULL`. For next calls, `*context` is set by previous call. Lemma is copied to `lemmaBuffer` and ended by zero byte, tag is copied to `tagBuffer` and ended by zero byte.

```
void cmi_free_tag(void **context)
```

Frees whatever was used by tagger in `**context` and sets `*context` to `NULL`.

```
long cmi_generate(void *handle, const char *lemma, const char *otherForm, char *tag,
       char *formBuffer, long formBufferLength, void *options = NULL)
```

Returns count of forms or -1 if any buffer is too short. Either `lemma` or `otherForm` may be `NULL`, othervise `lemma` has to be lemma of `otherForm`. Surface forms are copied to `formBuffer` and delimited by zero bytes.

7 Conclusion

We have proposed a common morphology interface for performing morphology operations.

This will serve as a solution of our needs within the HimL project but we believe it is also a starting point for further use in other projects. There are clear ways of further development and enriching today's tools with further features and values for new languages and use cases.

References

Zeman, Daniel. 2008. "Reusable tagset conversion using tagset drivers." *Proceedings of the Language Resources and Evaluation Conference, LREC 2008. CD full edition + printed Conference Abstracts (ISBN 2-9517408-4-0).* Marrakech, Morocco.