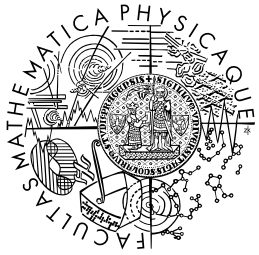

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Name Surname

Thesis title

Name of the department

Supervisor of the bachelor thesis: Supervisor's Name
Study programme: study programme
Study branch: study branch

Prague YEAR

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Dedication.

Title: Thesis title

Author: Name Surname

Department: Name of the department

Supervisor: Supervisor's Name, department

Abstract: Abstract.

Keywords: key words

Contents

Introduction

1. Implementation

Topic: Automatic inflection of Czech OOV nouns

1.1 The documentation of the Individual Software Project

The goal of this work is to implement a library for automatic inflection of Czech OOV nouns. The inflection model should provide a method `inflect(lemma: str)`, which receives a lemma of an OOV noun (the noun in the base form) and tries to "guess" and generate all the inflected forms of the lemma. In the Czech language, there are 7 possible cases and 2 numbers (singular, plural), therefore the number of returned forms is always 14 (and some of them can be unknown, i.e. for proper names which have only singular or pluralia tantum (Wikipedia [2022]), which have only plural.

In this project we have implemented several components for development of the inflection models, and also one inflection model, the simplest one: a retrograde model based on the MorfFlex dictionary (Hajič et al. [2020]), which for known lemmas contains the inflected forms.

We provide a list of the main components developed in this project (all source code can be found in directory `src/czech_inflection/`).

1.1.1 Inflection model interface proposal

`/models/model.py`

The inflection model should be a class, which during initialization loads everything it needs during prediction (e.g. trained NN or filled trie etc.). When we have an instance of a particular model, we should be able to call its method `inflect(lemma: str)`. It returns a list of inflected forms (strings).

1.1.2 MorfFlex processing

`/morfflex/`

Since we focus on inflection of OOV nouns, we need a vocabulary. And that is MorfFlex 2.0.

We use its processed form (extracted proper lemmas only) during the building of development data to determine whether a given noun is OOV (`lexicon.py`). We use the implementation of a Python set containing all the strings representing the lemmas (and we ask the set whether it contains a given lemma), which is fast even for the large size of MorfFlex.

In the retrograde model we use the train data extracted from the raw MorfFlex file using `Makefile` and `transform_morfflex.py`. In the `Makefile` we extract nouns only from the MorfFlex and shuffle the lemmas randomly (to be able to use a random part of the vocabulary). In the `transform_morfflex.py` we extract only basic variants of inflected forms (according to variant, style, abbreviation and also negation) and then remove the lemmas we consider faulty (those whose number of inflected forms is not 0, 1, 2, 7 nor 14, which we consider standard in

Czech - 0, 1 or 2 for indeclinable nouns, 7 for proper names or pluralia tantum and 14 for any standard noun with 7 cases in singular and 7 cases in plural).

1.1.3 Building of development data and automation of this process

`/dev_testing/build_data/`

As the source of the development data we use a part of the database of the web `https://cestina20.cz/` (specifically words beginning with the letter "e" and "j"), which was provided to us by the authors of the web. It contains neologisms with explanations.

In `cestina20reader.py` we extract the relevant parts of the database (lemma and explanations) and in `build_data.py` we process the data more.

The first part of the script (called by `python3 build_data.py --inflect`) removes word phrases and selects OOV words only, shuffles the lemmas randomly and then uses the rule-based baseline model (based on `sklonuj.cz`) to automatically inflect the lemmas. The inflected lemmas with explanations are pretty-printed to a text file to be checked manually.

Then we are expected to perform the manual check: correct the wrongly inflected forms, add multiple options of correctly inflected forms if there are more than a single one, mark the faulty lemmas and mark the end of the checked part of the file.

After the manual check, the second part of the script can be called by `python3 build_data.py --load_checked`. It loads the data from the manually edited text file, removes the lemmas marked as faulty and remove the explanations to obtain the development data, which are then printed to a specified file.

1.1.4 Model evaluation

`/dev_testing/eval_models/main.py`

Running `python3 main.py` without any arguments runs the evaluation of all available models on the development data.

We consider an inflected forms to be correct if it is one of the possible forms (if there are more of them) present in the development data. We use two different accuracies to evaluate the models. First, accuracy per inflected forms, is the percentage of correctly inflected forms in all the inflected forms.

The second one, accuracy per lemma, is the percentage of lemmas with all correctly inflected forms.

In both accuracies, we completely ignore the inflected forms which are marked as unknown in the development data.

1.1.5 Retrograde inflection model

`/models/retrograde_model/retrograde_model.py`

The Retrograde model is based on the simple idea used in the system ASIMUT (Králíková and Panevová [1990]), which aimed at information retrieval and one of which components was a simple inflection model using a retrograde dictionary (Slavíčková [1975]).

If we want to obtain the inflected forms of an unknown lemma, we find the closest known lemma (the longer the longest common suffix of two lemmas is, the closer they are considered) and then inflect the desired lemma according to the pattern of the known lemma. It reflects the fact that in Czech language, most of the inflection affects only the end of the word (sometimes even only the suffix) and is independent of the beginning of the word.

Our retrograde model uses a retrograde trie (`retrograde_trie.py`) to store the known lemmas. It is a wrapper over a standard trie (the keys are reversed). The trie is filled with a part of inflected lemmas from the train data (in one possible setting 100k lemmas). It supports iterating over all contained lemmas that are most similar to a given unknown lemma (where similarity is measured by the length of the longest common suffix).

The model inflects the given lemma according to all the most similar lemmas from the trie and then combines the inflected forms (for every desired form it lets the predicted forms to vote).

The inflection according to a pattern is almost as simple as replacing the pattern stem with the lemma stem in all inflected forms of the pattern.

1.1.6 Rule-based baseline model wrapper (sklonuj.cz)

`/models/hardcoded/sklonuj_cz.py`

The script `sklonuj_cz.py` provides a wrapper over the rule-based inflection model implemented in PHP, which is publically available on web (Sedlák [2013]).

1.1.7 Comparison of the accuracies

In the Table 1.1 we can see that the retrograde model performs better than the rule-based model measured by both accuracies. It inflects completely correctly 77 of 100 lemmas, whereas the in rule-based model it is only 55.

Tested model	Acc. by all inflected forms	Acc. per lemma
Sklonuj.cz	0.86	0.55
Retrograde model[size=100K]	0.92	0.77

Table 1.1: Comparison of accuracies of rule-based model and implemented Retrograde model

When inspecting the results (predicted inflected forms compared with the gold forms), it seems that the rule-based model usually incorrectly assumes that the unknown lemma is of the masculine inanimate gender (and inflects it according to such pattern). For examples see Table 1.2.

On the contrary, the Retrograde model seems to usually incorrectly assume the unknown lemmas to be of the masculine animate gender (see Table 1.3). With no thorough inspection of the train data composition we can only guess that it can be caused by the masculine animate being the most common gender in the train data.

Lemma: eurobubák	predicted form	gold form
S2 (bez)	eurobubáku	(eurobubáka)
S4 (vidím)	eurobubák	(eurobubáka)
P1 (ti/ty/ta)	eurobubáky	(eurobubáci/eurobubákové)
P5 (volám)	eurobubáky	(eurobubáci/eurobubákové)
Lemma: europosranec	predicted form	gold form
S4 (vidím)	europosranec	(europosrance)
P1 (ti/ty/ta)	europosrance	(europosranci/europosrancové)
P5 (volám)	europosrance	(europosranci/europosrancové)

Table 1.2: Example of incorrect predictions of the Sklonuj.cz rule-based model

Lemma: elpaso	predicted form	gold form
S3 (ke/k)	elpasovi	(elpasu)
S4 (vidím)	elpasa	(elpaso)
S6 (o)	elpasovi	(elpasu)
P1 (ti/ty/ta)	elpasové	(elpasa)
P2 (bez)	elpasů	(elpas)
P4 (vidím)	elpasy	(elpasa)
P5 (volám)	elpasové	(elpasa)
Lemma: Jágrov	predicted form	gold form
S3 (ke/k)	Jágrovovi	(Jágrovu)
S4 (vidím)	Jágrova	(Jágrov)
S6 (o)	Jágrovovi	(Jágrovu)

Table 1.3: Example of incorrect predictions of the implemented Retrograde model

1.1.8 Conclusion

The overall result is satisfying: we have created an inflection model that performs better than the rule-based one.

2. Literature from the BP task

List of literature:

Flect (Dušek and Jurčíček [2013])
MorfFlex tags (Mikulová et al. [2020])
MorfFlex (Hajič et al. [2020])
MorphoDiTa(Straková et al. [2014])
Morphological inflection generation (Zhou and Neubig [2017])

2.1 Example of including source code

```
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```

Conclusion

Bibliography

Ondřej Dušek and Filip Jurčíček. Robust multilingual statistical morphological generation models. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 158–164, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://aclanthology.org/P13-3023.

Jan Hajič, Jaroslava Hlaváčová, Marie Mikulová, Milan Straka, and Barbora Štěpánková. Morfflex cz 2.0, 2020. URL http://hdl.handle.net/11234/1-3186. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (
'UFAL), Faculty of Mathematics and Physics, Charles University.

Květoslava Králíková and Jarmila Panevová. ASIMUT - a method for automatic information retrieval from full texts. *Explizite Beschreibung der Sprache und automatische Textbearbeitung*, XVII, 1990.

Marie Mikulová, Daniel Zeman, Barbora Vidová Hladká, Barbora Štěpánková, Emil Jeřábek, Jaroslava Hlaváčová, Hana Hanová, Jiří Hana, and Jan Hajič. *Manual for Morphological Annotation. Revision for Prague Dependency Treebank – Consolidated 2020 release*, volume ÚFAL TR-2020-64 of *The ÚFAL/CKL Technical Report Series (ISSN 1214-5521)*. Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University, 2020. URL https://ufal.mff.cuni.cz/techrep/tr64.pdf.

Pavel Sedlák, 2013. URL https://sklonuj.cz/.

Eleonora Slavíčková. *Retrográdní morfematický slovní češtiny.* Academia, 1. edition, 1975.

Jana Straková, Milan Straka, and Jan Hajič. Open-source tools for morphology, lemmatization, POS tagging and named entity recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-5003. URL https://aclanthology.org/P14-5003.

Wikipedia. Plurale tantum, Aug 2022. URL https://en.wikipedia.org/wiki/Plurale_tantum.

Chunting Zhou and Graham Neubig. Morphological inflection generation with multi-space variational encoder-decoders. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 58–65, Vancouver, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/K17-2005. URL https://aclanthology.org/K17-2005.

List of Figures

List of Tables

List of Abbreviations

A. Attachments

A.1 First Attachment