METL – Project
Morphological inflection

14.04.2022
Deadline 13.05.22 by midnight

In this project you will explore the use of seq2seq models for morphological inflection. You can work and submit this project in pairs (in which case the same grade will be given to both students), or individually. The task is based on the shared task of the ACL Special Interest Group on Computational Morphology and Phonology (SIGMORPHON) in 2016, shared task 1. For all the materials you will be redirected to their github repository.

Morphological inflection (from Cotterell et al., 2016)[1]:

Many languages use systems of rich overt morphological marking in the form of affixes (i.e. suffixes, prefixes, and infixes) to convey syntactic and semantic distinctions. For example, each English count noun has both singular and plural forms (e.g. robot/robots, process/processes), and these are known as the inflected forms of the noun.
While English has relatively little inflectional morphology, Russian nouns, for example, can have a total of 10 distinct word forms for any given lemma and 30 for an imperfective verb.

The fact that inflected forms are systematically related to each other, as shown in Figure 1, is what allows humans to generate and analyze words despite this level of morphological complexity.

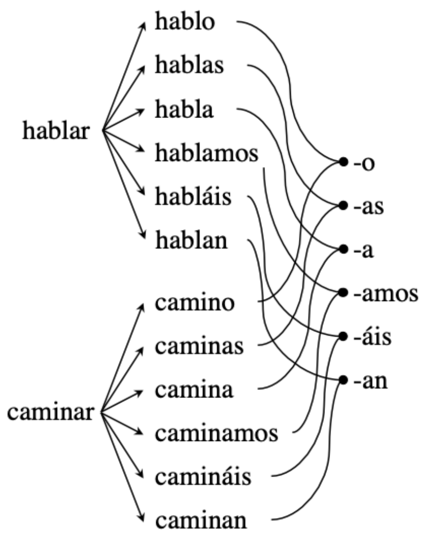

Figure 1: The relatedness of inflected forms, such as the present indicative paradigm of the Spanish verbs *hablar* 'speak' and *caminar* 'walk,' allows generalizations about the shape and affixal content of the paradigm to be extracted.

[1] Cotterell, R., Kirov, C., Sylak-Glassman, J., Yarowsky, D., Eisner, J., & Hulden, M. (2016, August). The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology* (pp. 10-22).

Developing systems that can accurately learn and capture these mappings, overt affixes, and the principles that govern how those affixes combine is crucial to maximizing the crosslinguistic capabilities of most human language technology.

The task:

Given a **lemma** (the dictionary form of a word) with its part-of-speech, generate a target inflected form.

English example -
Source lemma: *run*   Target tag: *Present participle*   Output: *running*

From SIGMORPHON 2016 github repository, download the following data files from the data folder:

- Finnish-task1-train
- Finnish-task1-dev
- Finnish-task1-test

In each data file there are three columns of information as in the example from English above – a lemma, the target morpho-syntactic description (MSD) provided as a set feature/value pairs, and the target form. The fields on a line are TAB-separated.

Line 1 in the train data:

```
ääkköstää	pos=V,polar=POS,mood=IMP,tense=PRS,per=3,num=SG	ääkköstäköön
```

The input to the model should include the first two columns and the output should be as close as possible to the form in column 3.

Use the train dataset to train the model, the dev to check if further changes are needed in your model or encoding method and the test set for your final results.

Encoding – Choose how to encode the input sequences given to the model. Describe the way you encoded the data in the project report.

The model – you can train and test a sequence-to-sequence model for this task (or you can suggest a different model and justify your choice). Describe in the final report the architecture and specifications of the model.

Evaluation –

Base line model-
You will compare your model performance to a baseline model, found here.
Run the baseline model on the same data and in the final project, describe the baseline model.

Your system should predict a single string for each test example. Optionally, you may also produce a ranked list of up to 20 predictions for each test example.

You can find the evaluation script here. The script reports:
Accuracy = fraction of correctly predicted forms
Average Levenshtein distance between the prediction and the truth
Mean reciprocal rank of the truth (if your system produces ranked lists)

Compare the performance of your model with the baseline model and present the results in the final report. Results should be presented in a table together with a paragraph with your interpretation of the results.

Bonus points: visualise the results together with a description of the graph.

Deliverables:

You need to submit you code for the model and a final report.
Final report – the final report should be up to three pages long and should include a description of the task, the method – encoding and the model you used, a description of the base line model and a result and conclusion section.

Code – keep the code commented and clear for an external reader (me) to understand.

Grading:

Although your project is not going to be graded according to your model's performance, you are expected to think of that aspect when making choices in the encoding phase and in the model you're using with its specifications. These choices need to be mentioned in the final report with your motivation in making them.

Your project will be assessed according to the grading scheme below.
A coherent short presentation of the task – 5 points.
Description of the data encoding process -5 points
Description of the model used for the task – 15 points
Description of the baseline model – 5 points
Description of the evaluation process – 10 points.
Presentation of the results – 15 points (5 extra points for good visual presentation).
Discussing the results and conclusions – 5 points.

Max total – 60 points