Deep Learning in NLP

Homework P4

Solution to be sent (pdf, zip) to `waszczuk@phil.hhu.de` and `cwurm@phil.hhu.de` by 12.11.2019 (included).

Preparation

Start from the code we implemented today during the practical session (05.11.2019). You can find it at `https://github.com/kawu/hhu-dl-materials/tree/master/lang-rec/code`. You can use `http://kinolien.github.io/gitzip/` to download it as a zip.

Exercise 1

The goal of the first exercise is to improve the current implementation.

1a

Propose a method to determine adequate values of the hyper-parameters: the embedding size and the size of the hidden layer of the scoring FFN. You can describe the method in words, provide a formula or a pseudocode, or implement it.

Updates:

- **11/11**: This is a „food for thought" exercise. So don't worry too much about your solution not being the correct one.

1b (optional)

Recall that there is a bug in the `embedding.py` module. If you find it and correct it, you may be able to significantly improve the performance of the network.

Exercise 2

The current solution seems a bit weak in that it completely ignores the order of the characters in the given name (e.g., *leon* and *noel* will both get the same scores). This is because:

- It uses *unigram* character-level features, therefore local ordering information is lost.
- It uses CBOW to represent names, which also completely ignores both local and non-local ordering information.

We will later see how to improve on the second point. The goal of this exercise is try to improve on the first one. The task is to:

- Implement a *bigram* model in which each pair of adjacent characters is embedded separately. For instance, *leon* should be first split into three bigrams *le*, *eo*, and *on*, and each of these bigrams should receive its own vector embedding.
- To obtain the name representation, CBOW can be used (sum of the vector embeddings of the bigrams in the given name).

Does the bigram-based input representation allow to improve the performance (decrease loss, improve accuracy) on the dev set (`dev20.csv`)? If not, try to determine the reasons why.

Updates:

- **11/11**: The places where you should update the code are not marked with `TODO`s, identifying them is part of the exercise. Nevertheless, the only thing that really changes in the bigram-based model is the implementation of the `LangRec` class (you can of course, and are encouraged to do so, implement the bigram-based model in a seprate class, based on `LangRec`).