Statistical Natural Language Processing

Python Refresher I: Exercises

Verena Blaschke[1]

April 20, 2018

[1] Based on slides by Kuan Yu.

Exercises

The 36 part-of-speech tags used in the Penn Treebank Project:[2]

```
penn_pos_tags = "CC CD DT EX FW IN JJ JJR JJS " \
                "LS MD NN NNS NNP NNPS PDT POS PRP " \
                "PRP$ RB RBR RBS RP SYM TO UH VB " \
                "VBD VBG VBN VBP VBZ WDT WP WP$ WRB"
```

[2] You can find an overview with short descriptions at
www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

Exercises

1) idx2tag

Create a list named `idx2tag` with the tags in `penn_pos_tags`, such that:

```
assert 36 == len(idx2tag)
assert 'TO' == idx2tag[24]
assert 24 == idx2tag.index('TO')
```

2) tag2idx

Create a dictionary named `tag2idx` which inverses `idx2tag`, such that:

```
assert all(idx == tag2idx[tag]
           for idx, tag in enumerate(idx2tag))
```

Exercises

3) sent

The string `sent` contains the POS tags corresponding to the words in some sentence.

```
sent = "DT NN PRP MD VBG VBZ RB DT JJ NN , " \
       "CC PRP MD VB DT NN VBN IN NN ."
```

Use `tag2idx` to create the list `sent_int` that encodes `sent` as a list of integers.
Beware that `sent` may contain some tags not found in `penn_pos_tags`, in which case you should update `idx2tag` and `tag2idx`. It should afterwards still be the case that

```
assert all(idx == tag2idx[tag]
           for idx, tag in enumerate(idx2tag))
```

Exercises

4) one-hot encoding

One-hot encoding describes a sequence of bits, all of which are 0 except for a single one that is 1.
Write the body of the function `one_hot`:

```
def one_hot(idx, dim):
    """
    Creates a one-hot vector.

    Arguments:
    idx: An int giving the position of the `1`.
    dim: An int describing the length of the list.

    Returns:
    A list(int) that is a one-hot vector.
    """
```

Exercises

5) matrix

Create a nested list `matrix` that encodes the sentence as a list of one-hot arrays. That is, it should be like `sent_int`, but each integer is replaced with the corresponding one-hot list.
It can be interpreted as a matrix with `len(sent_int)` many rows and `len(tag2idx)` many columns.

```
assert len(matrix) == len(sent_int)
assert len(matrix[0]) == len(tag2idx)
assert matrix[0] == one_hot(tag2idx['DT'], len(tag2idx))
```

Exercises

6) saving and reading files

Define a function `save_matrix` for saving two-dimensional matrices as CSV files.
Define another function `load_matrix` for loading matrices from CSV files created with the first function.

```
path = 'matrix.csv'
assert matrix == load_matrix(save_matrix(matrix, path))
```