python intro exercises

kuan yu[1]

<2017-04-18 Tue>

[1] kuan.yu@student.uni-tuebingen.de

exercise 1

```
ptb_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 "
```

the variable *ptb_tags* is a single string (this is how you break a string on multiple lines), containing 36 part-of-speech tags in the penn-treebank tagset.

encode categorical values

a *list* represents a mapping from integers (the indices) to its elements, and a *dict* represents a mapping from its keys to values.

exercise 1.1 *idx2tag* : *int* $\rightarrow$ *str*

create a *list* named *idx2tag* with the tags in *ptb_tags*, such that:

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

exercise 1.2 *tag2idx* : *str* $\rightarrow$ *int*

create a *dict* named *tag2idx*, which inverses *idx2tag*, such that:

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

exercise 2

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

sent is a delexicalized sentence: the words are forgotten, and only their postags remain.

encode *sent*

encode *sent* as a list of integers with *tag2idx*; beware that *sent* may contain some tags not found in *ptb_tags*, in which case, update *idx2tag* and *tag2idx* to accommodate the new tags.

exercise 3

learn about one-hot encoding:
https://en.wikipedia.org/wiki/One-hot

one_hot

define a function named *one_hot* which takes two integers *idx* and *dim* as arguments, and returns a one-hot vector, such that:

```
vec = one_hot(idx, dim)
assert dim == len(vec)
assert 1 == vec[idx]
assert 1 == sum(vec)
```

exercise 4

```
dim = len(tag2idx)
mat = [one_hot(tag2idx[tag], dim) for tag in sent]
```

mat is a matrix of *len(sent)* rows and *dim* columns. the rows represent the words, and the columns represent the location of this sentence in the hyperspace of postags.

transpose

transpose *mat* into a matrix of *dim* rows and *len(sent)* columns, so that the meanings of its rows and its columns are exchanged.

exercise 5

find out how to read & write files in python.

exercise 5.1: *save_mat*

define a function *save_mat* for saving matrics in *csv* format.

exercise 5.2: *load_mat*

define a function *load_mat* for loading matrics from *csv* files.

such that:

```
path = "mat.csv"
assert mat == load_mat(save_mat(mat, path))
```

exercise 6

count the frequencies of each tag in *sent*, store the results in a *dict* named *freq*, such that:

```
all(freq[tag] == sent.count(tag) for tag in sent)
```

this might help

https://docs.python.org/3/library/collections.html