Programming and Calssification Summer 2019

Python Basics vol. 2 — 06.03, 2019

Dr hab. inż. Marek Klonowski *Scribe: Krzysztof Agieńczuk*

1 Overview

In the last lecture we talked about basics of python.

In this lecture we continued introduction to python and mentioned basic methods of text analysis.

2 Python basic info

In python strings are similar to arrays, ie. you can access n-th element of string s by `s[n]`. On that note, don't look for array in python. They are called lists. Lists in sense of linked lists are not a part of standard library.
Containers can be added together. Beware though to keep them one type. Tuples can be added to tuples, and so on. Adding list to a tuple is not allowed. In the same manner concatenation of strings is possible. When we talk about strings, take a look at different string method - `sort()`, `remove()`, `split()` can all be very useful when dealing with strings.

We talked about two types of containers - lists and sets. Both can be used in loops, to iterate over them. They differ in fact that sets are unordered and can't contain duplicates (lists can).

Another useful method is `zip()`. It allows for making a list of tuples from two lists.

2.1 Lambda functions

Lambda function, in other words anonymous function is another way to write simple functions. Instead using word `def` and whole method body, you can write `g=lambda x: x**2` to get the square of original argument.

2.2 List transform

It is possible to write `f(x) for x in my_list` to use list transform. It allows for easier iteration over containers.

2.3 Dictionary

Dictionaries are python's version of maps. It has a construction {k,v} where k stands for key, and v stands for value. Access is similar to lists, ie. `dict["key"]=value`. Deleting is possible by using word `del`. Command `list(dict.values())` will return list of values of the dictionary.

Analogically works method `keys()`. Dictionaries can contain elements of different types.
As dictionaries are derived from sets, they cannot be sorted. To achieve it, we have to first make a list from it. Exemplary way

```
sort = sorted(dict.keys(), key=lambda x:  x[1])
list(sort).
```

This command will sort set by values in descending manner.

3 Natural Language Processing

The problem we mentioned is categorising documents. Eg. we want to decide if given text is about soccer. The first idea is to use characteristic words. We would need to create binary classifier. The problem is that characteristic words can occur in different contexts. Also, there are texts about the topic, which don't contain the characteristic words. This means it is not possible to create perfect classifier.

Another way is to decide which words are important in a given document. We use frequency of occurring in text. One of such algorithms is TF.IDF. It's Term Frequency times Inverse Document Frequency.

$$TF_{i,j} = \frac{f_{i,j}}{max_k f_{k,j}} \tag{1}$$

In this algorithm we have N documents. $f_{i,j}$ number of times i-th term appears in the j-th document. TF.IDF+ is one where we remove "stop" words. Sometimes we use sum instead of max function.

Definition Stop words are words filtered out before processing. In English usually words like "and, or, the, is"

The IDF part gets rid of stop words.

$$IDF_i = log_2(\frac{N}{n_i}) \tag{2}$$

Final score is counted as

$$TF.IDF_{i,j} = TF_{i,j} * IDF_i \tag{3}$$

Note At the end prof mentioned Natural Language Toolkit and showed some examples for using it.