MTAT.03.311
Loomuliku keele töötlus *Python*is
(6 EAP)

Raul Sirel

01.09.2015

Korralduslikku

- Toimub 2x nädalas:
  - T 12.15 Liivi 2-205 (praktikum)
  - T 14.15 Liivi 2-611 (loeng)

- Õppejõud:
  - Raul Sirel (rsirel@ut.ee)
  - Karl-Oskar Masing (kom@ut.ee)

- Kursuse veebileht: https://courses.cs.ut.ee/2015/pynlp/fall

Käsitletavad teemad

- Numpy
- Keeleandmete tehniline eeltöötlus
- Keeleandmete lingvistiline eeltöötlus
- Keeleandmete kogumine veebist
- Masinõpe ja selle kasutamine loomuliku keele töötluses
- Tunnuste eraldamine
- Informatsiooni ekstraheerimine
- Loomuliku keele vektormudelid
- Klasterdamine
- Jne

- Täpne nimekiri selgub kursuse vältel

Hinde kujunemine

- Praktikumiülesannete lahendused (75%)
- Kirjalik projektiraport (20%)
- Projekti ettekanne seminaris (5%)

- Praktikumiülesanded peavad olema esitatud 75% ulatuses

Praktikumiülesanded

- Iga praktikumi tarbeks on digitaalne käsileht
- Praktikumiülesannete lahendused tuleb esitada 2 nädala jooksul alates praktikumi toimumisest
- Nt 1. septembri praktikumi lahenduses tuleb esitada hiljemalt 14. septembriks 23:59
- Ülesannete lahendused tuleb esitada aine veebilehel: https://courses.cs.ut.ee/2015/pynlp/fall

Projekt

- Projekti eesmärk on kinnistada kursuse vältel kogutud oskusi
- Sõltuvalt osalejate arvust saab projekti läbi viia individuaalselt või kaksi
- Projekti tulemused vormistatakse kirjaliku raportina ning kantakse ette seminaris

Projekti tähtaeg

- Projekti ettekanded toimuvad semestri viimasel nädalal (15.12.2015)

- Projekti kirjaliku raporti tähtaeg on 15.01.2016

NumPy

Numpy – mis ja milleks?

- Numeric Python – teek andmemassiivide haldamiseks ja manipuleerimiseks

- Puhas Python on suurte andmestike jaoks aeglane, mistõttu kasutab Numpy C-keelt

Andmemassiiv ehk *array*

- Numpy toob kasutajani uue andmestruktuuri – *array*
- Array ehk andmemassiiv sarnaneb Pythonisse sisseehitatud järjendi (*list)* andmestruktuurile
- Andmemassiiv on võrreldes järjendiga hõlpsamini hallatav ning selle töötlemiseks leidub hulk sisseehitatud meetodeid
- Andmemassiiv võib olla *n*-dimensionaalne (nt 2D massiiv on järjendite järjend ehk *list of lists*)

Andmemassiivi loomine

- 1D massiivi loomine Pythoni järjendi alusel:

```
>>> import numpy
>>> l = [1,2,3]
>>> arr = numpy.array(l)
>>> arr
array([1, 2, 3])
```

- Levinud viga:

```
>>> arr = numpy.array(1,2,3)
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    numpy.array(1,2,3)
ValueError: only 2 non-keyword arguments
accepted
```

- Sama 2D massiiviga:

```
>>> l = [[1,2,3],[4,5,6]]
>>> arr = numpy.array(l)
>>> arr
array([[1, 2, 3],
       [4, 5, 6]])
```

Andmemassiivi loomine (2)

- Tühja massiivi loomine:

```
>>> arr = numpy.zeros((2,3))
>>> arr
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
```

- Juhusliku massiivi loomine:

```
>>> arr = numpy.random.uniform(0,1,(2,3))
>>> arr
array([[ 0.48810202,  0.88288271,  0.75692414],
       [ 0.81976378,  0.44220499,  0.4118117 ]])
```

- Andmetüübi defineerimine:

```
>>> numpy.sctypes
{'int': [<type 'numpy.int8'>, <type 'numpy.int16'>, <type 'numpy.int32'>, <type
'numpy.int64'>], 'float': [<type 'numpy.float16'>, <type 'numpy.float32'>, <type
'numpy.float64'>], 'uint': [<type 'numpy.uint8'>, <type 'numpy.uint16'>, <type
'numpy.uint32'>, <type 'numpy.uint64'>], 'complex': [<type 'numpy.complex64'>, <type
'numpy.complex128'>], 'others': [<type 'bool'>, <type 'object'>, <type 'str'>, <type
'unicode'>, <type 'numpy.void'>]}

>>> l = [[1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15]]
>>> arr = numpy.array(l,dtype="float32")
>>> arr
array([[  1.,   2.,   3.,   4.,   5.],
       [  6.,   7.,   8.,   9.,  10.],
       [ 11.,  12.,  13.,  14.,  15.]], dtype=float32)
```

Väärtuste poole pöördumine

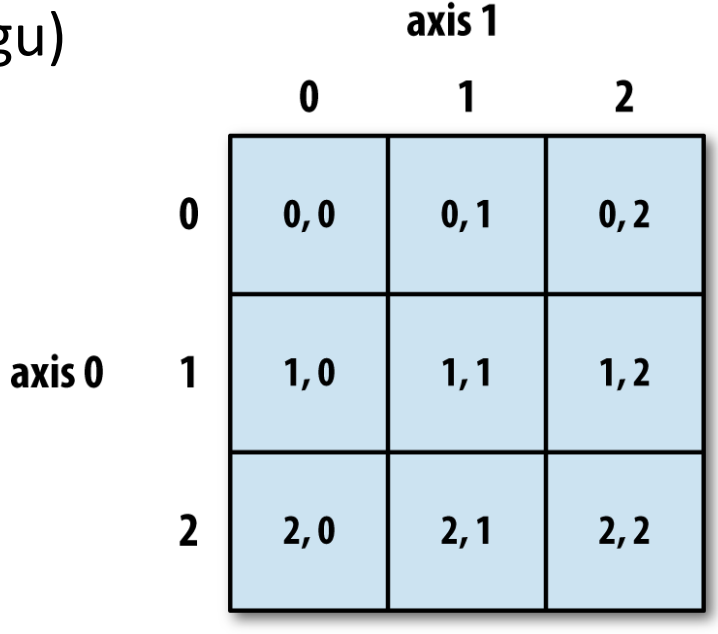
- Väärtuste poole pöördumine toimub nagu mitmedimensioonilistes järjenditeski
- Nt 2D massiivis (ehk maatriksis) on tarvis teada kahte koordinaati (rida ja veergu)

```
>>> l = [[1, 2, 3],[4,5,6]]
>>> arr = numpy.array(l)
>>> arr
array([[1, 2, 3],
       [4, 5, 6]])

>>> arr[1]
array([4, 5, 6])

>>> arr[1][0]
4

>>> arr[-1][-1]
6
```

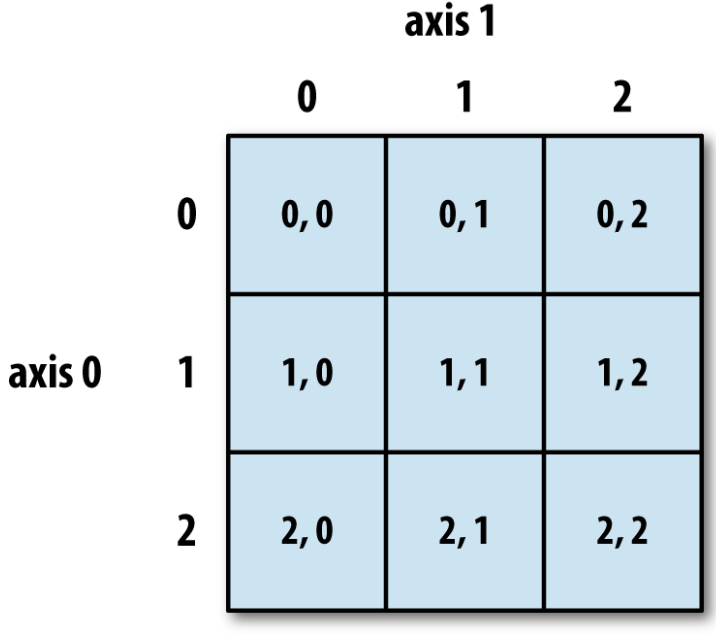


Väärtuste defineerimine

```
>>> arr = numpy.zeros((2,3))
>>> arr
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])

>>> arr[1][2] = 42
>>> arr
array([[  0.,   0.,   0.],
       [  0.,   0.,  42.]])

>>> arr[-2][1] = 7
>>> arr
array([[  0.,   7.,   0.],
       [  0.,   0.,  42.]])
```



Massiivide tükeldamine

```
>>> l = [[1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15]]
>>> arr = numpy.array(l)
>>> arr
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15]])
```

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

- Massiive tükeldatakse ridade, veergude ja nende vahemike vahemike defineerimise abil:

```
>>> arr[0,1:4]
array([2, 3, 4])
```

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

Massiivide tükeldamine(2)

```
>>> arr[0:2,0:2]
array([[1, 2],
       [6, 7]])

>>> arr[:2,:2]
array([[1, 2],
       [6, 7]])

>>> arr[:-1,:2]
array([[1, 2],
       [6, 7]])
```

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

```
>>> arr[:,1:2]
array([[ 2],
       [ 7],
       [12]])

>>> arr[,1:2]
array([[ 2],
       [ 7],
       [12]])
```

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

Massiivide tükeldamine(3)

```
>>> arr[:,:3]
array([[ 1,  2,  3],
       [ 6,  7,  8],
       [11, 12, 13]])
```

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

```
>>> arr[:2,1]
array([2, 7])
```

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

```
>>> arr[:,-2:]
array([[ 4,  5],
       [ 9, 10],
       [14, 15]])
>>>
```

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

Massiivide kombineerimine

- Massiivide konkatenatsioon:

```
>>> arr1 = numpy.array([[1,2,3]])
>>> arr2 = numpy.array([[4,5,6]])

>>> numpy.concatenate((arr1,arr2),axis=1)
array([[1, 2, 3, 4, 5, 6]])

>>> numpy.concatenate((arr1,arr2),axis=0)
array([[1, 2, 3],
       [4, 5, 6]])
```

- Veergude kuhjamine:

```
>>> arr1 = numpy.array([1,2,3])
>>> arr2 = numpy.array([4,5,6])
>>> arr3 = numpy.array([7,8,9])

>>> numpy.column_stack((arr1,arr2,arr3))
array([[1, 4, 7],
       [2, 5, 8],
       [3, 6, 9]])
```

Tehted massiividega

```
>>> arr1 = numpy.array([1,2,3])
>>> arr2 = numpy.array([4,5,6])
```

- Liitmine ja lahutamine:

```
>>> arr1+arr2
array([5, 7, 9])

>>> numpy.add(arr1+arr2)
array([5, 7, 9])

>>> arr2-arr1
array([3, 3, 3])

>>> numpy.subtract(arr2-arr1)
array([3, 3, 3])
```

- Korrutamine ja jagamine:

```
>>> arr1*arr2
array([ 4, 10, 18])

>>> numpy.multiply(arr1,arr2)
array([ 4, 10, 18])

>>> arr1/arr2
array([0, 0, 0])

>>> arr1 = numpy.array([1.0,2.0,3.0])
>>> arr1/arr2
array([ 0.25,  0.4 ,  0.5 ])

>>> numpy.divide(arr1,arr2)
array([ 0.25,  0.4 ,  0.5 ])
```

Tehted massiividega (2)

```
>>> arr = numpy.array([1,2,3,4,5])
```

- numpy.sum():

```
>>> numpy.sum(arr)
15

>>> numpy.sum(arr[2:4])
7
```

- numpy.power():

```
>>> numpy.power(arr,2)
array([ 1,  4,  9, 16, 25])

>>> numpy.power(arr,4)
array([  1,  16,  81, 256, 625])
```

- numpy.max():

```
>>> numpy.max(arr)
5
```

- numpy.min():

```
>>> numpy.max(arr)
1
```

Olulisemaid massiivide meetodeid

```
>>> arr = numpy.array([[1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15]])
>>> arr
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15]])
```

- Informatsioon massiivi kohta:

```
>>> arr.shape
(3L, 5L)

>>> arr.dtype
dtype('int32')
```

- Pööritamine

```
>>> arr.transpose()
array([[ 1,  6, 11],
       [ 2,  7, 12],
       [ 3,  8, 13],
       [ 4,  9, 14],
       [ 5, 10, 15]])

>>> numpy.swapaxes(arr,0,1)
array([[ 1,  6, 11],
       [ 2,  7, 12],
       [ 3,  8, 13],
       [ 4,  9, 14],
       [ 5, 10, 15]])
```

Massiivide talletamine kettal

```
>>> arr = numpy.array([[1,2,3],[4,5,6]])
>>> arr
array([[1, 2, 3],
       [4, 5, 6]])
```

- Tekstifailina:

```
>>> numpy.savetxt(“array.txt”,arr)

>>> arr_new = numpy.loadtxt(“array.txt”)
>>> arr_new
array([[1, 2, 3],
       [4, 5, 6]])
```

- Binaarfailina:

```
>>> numpy.save(“array.npy”,arr)

>>> arr_new = numpy.load(“array_npy”)
>>> arr_new
array([[1, 2, 3],
       [4, 5, 6]])
```

- Väljaeraldaja defineerimine:

```
>>> numpy.savetxt("array.txt", arr,
delimiter="\t")

>>> arr_new = numpy.loadtxt("array.txt",
arr, delimiter="\t")

>>> arr_new
array([[1, 2, 3],
       [4, 5, 6]])
```

Massiivide itereerimine

- Massiivid on sarnaselt järjenditele itereeritavad
- Massiivi kõikide elementide itereerimiseks tuleb kasutada topelt for-tsüklit
- Suurte massiivide itereerimine on aeglane!

```
>>> arr = np.random.uniform(0,1,(3,2))
>>> arr
array([[ 0.67798285,  0.91793478],
       [ 0.1400463 ,  0.6399191 ],
       [ 0.89242515,  0.7387817 ]])
>>> for row in arr:
          print row

[ 0.67798285  0.91793478]
[ 0.1400463  0.6399191]
[ 0.89242515  0.7387817 ]

>>> for row in arr:
          for col in row:
                    print row

[ 0.67798285  0.91793478]
[ 0.67798285  0.91793478]
[ 0.1400463  0.6399191]
[ 0.1400463  0.6399191]
[ 0.89242515  0.7387817 ]
[ 0.89242515  0.7387817 ]
```

Algoritmiline näpuharjutus

Algoritmiline näpuharjutus

- Koostame massiivi, mis sisaldab sõnasagedusi dokumentide lõikes:

- documents.txt
  - https://courses.cs.ut.ee/2015/pynlp/fall/uploads/Main/documents.txt
  - üks dokument real

	$sõna_1$	$sõna_2$	...	$sõna_n$
$dokument_1$				
$dokument_2$				
...				
$dokument_n$				

- Milline on algoritmiline lahendus?
- Mida tuleb arvestada?

Algoritmi põhistruktuur

- Algoritm koosneb kahest põhilisest osast:
  - sõnasageduste kokkulugemine
  - sõnasageduste esitamine massiivina
- Mõlema sammu puhul tuleb itereerida üle
  - dokumentide
  - sõnade
- Kuna teksti itereerimine on aeglane, tasub rohkemaid iteratsioone vältida

Sõnasageduste sõnaraamat

- Esimeses etapis itereerimine ning koostame sõnaraamatu sõnasagedustest dokumentide lõikes

- Kasutame sõnaraamatut, kuna sellest otsimine on kiire

```
freqs = {}
lexicon = {}

with open('documents.txt') as docs:
    for i,doc in enumerate(docs):
        doc = doc.strip().lower()
        freqs[i] = {}
        for word in doc.split(" "):
            if word not in freqs[i]:
                 freqs[i][word] = 0
            freqs[i][word] += 1
            if word not in lexicon:
                lexicon[word] = True
```

Sagedusmaatriks

- Teises etapis loome tühja maatriksi ning täidame selle sõnasagedustega

- Selleks itereerimine üle sõnasageduste sõnaraamatus olevate dokumentide ja sõnade:

```
arr = np.zeros((len(freqs.keys()),len(lexicon.keys())))

for j,document_id in enumerate(freqs.keys()):
    for k,lexicon_word in enumerate(lexicon.keys()):
        if lexicon_word in freqs[document_id]:
            arr[j][k] = freqs[document_id][lexicon_word]
```

Kasulik koodijupp

```
import numpy as np

freqs = {}
lexicon = {}

with open('documents.txt') as docs:
    for i,doc in enumerate(docs):
        doc = doc.strip().lower()
        freqs[i] = {}
        for word in doc.split(" "):
            if word not in freqs[i]:
                freqs[i][word] = 0
            freqs[i][word] += 1
            if word not in lexicon:
                lexicon[word] = True

arr = np.zeros((len(freqs.keys()),len(lexicon.keys())))

for j,document_id in enumerate(freqs.keys()):
    for k,lexicon_word in enumerate(lexicon.keys()):
        if lexicon_word in freqs[document_id]:
            arr[j][k] = freqs[document_id][lexicon_word]
```