Opdracht 2 Letterfrequenties

Laurens Lancel – 1782607

Github - https://github.com/Lancelot1106/letterfrequenties

Om de opdracht te beginnen heb ik het kopje “procedure” compleet genegeerd en heb ik de mapreduce in python gemaakt. Dit was een relatief simpel process door de code aan de hand van volgend voorbeeld te schrijven:

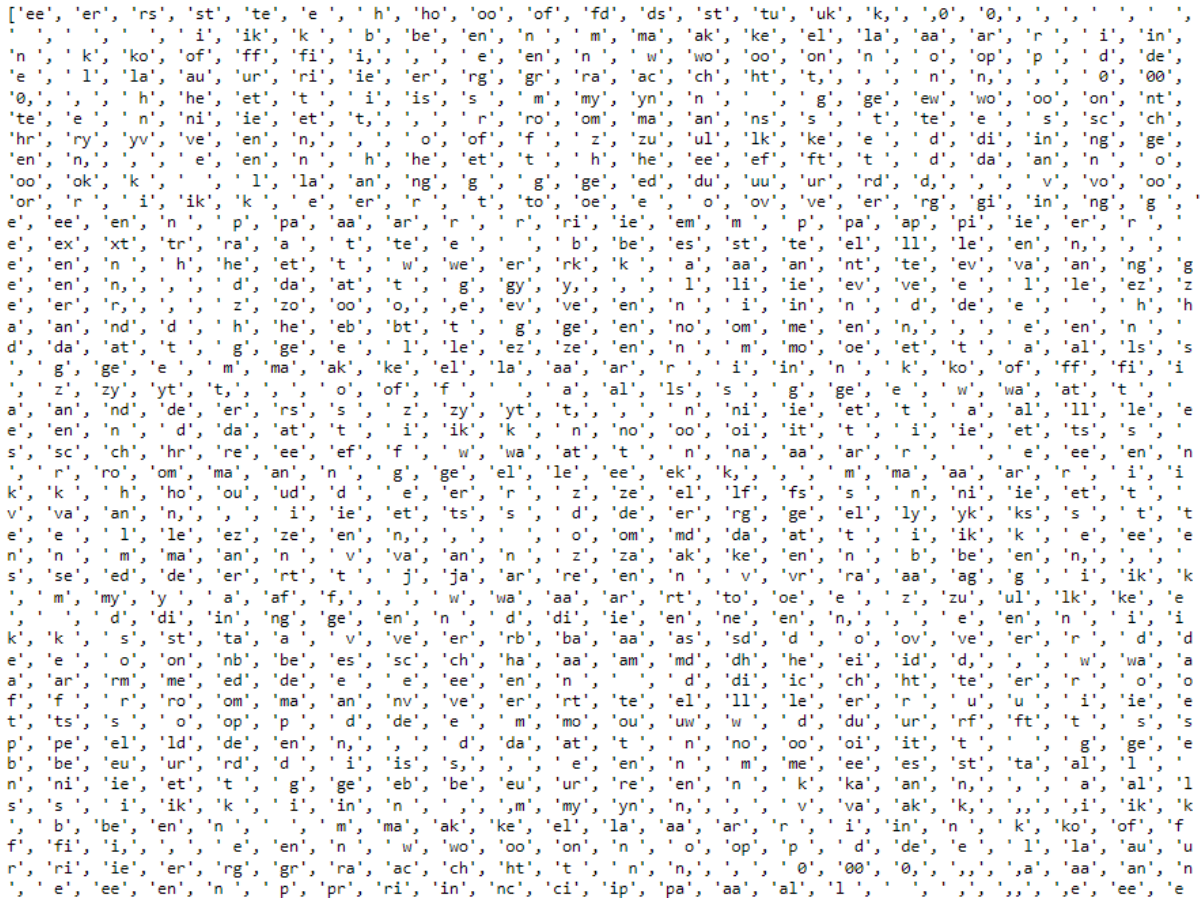
https://medium.com/geekculture/mapreduce-with-python-5d12a772d5b3

Aangezien de meeste MapReduce structuren echter gebaseerd zijn op woorden heb ik deze aangepast naar de functies in hun huidige staat. De mapper neemt een lijst van character pairs en vormt deze om naar een lijst van tuples met het pair en een 1, zoals in bovenstaande afbeelding. Hierna neemt de shuffle_sort deze lijst en zet deze om naar een unsorted dictionary, gevolgd door het sorteren van de keys (en daarmee ook gelijk de values bij die keys). Voor deze sorted dictionary zet de reducer de lijsten van de values om naar een enkel getal, namelijk de som van de lijst.

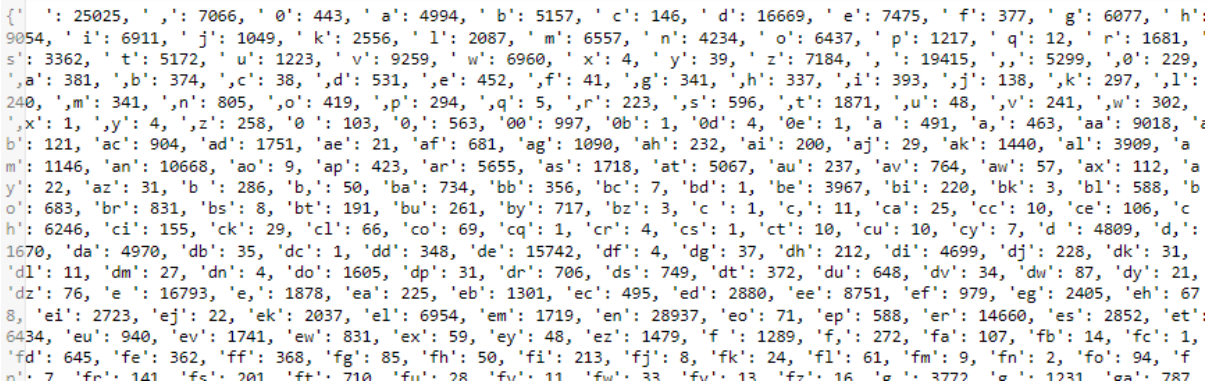
```python
def mapper(char_list:list):
    mapped = []
    for char_pair in char_list:
        mapped.append((char_pair, 1))
    return mapped


def shuffle_sort(mapped_list:list):
    shuffled = {}
    for mapped in mapped_list:
        if mapped[0] in shuffled.keys():
            shuffled[mapped[0]].append(1)
        else:
            shuffled[mapped[0]] = [mapped[1]]

    sort = {}
    keys = list(shuffled.keys())
    keys.sort()
    for key in keys:
        sort[key] = shuffled[key]

    return sort

def reducer(shuffled_dict):
    reduced = {}
    for key, value in shuffled_dict.items():
        reduced[key] = sum(value)
    return reduced
```

Hierna is het systeem om de matrix op te bouwen gemaakt. In eerste instantie waren dit losse functies, wat ik later om heb gezet naar een klasse. Het systeem werkt als volgt (de afbeeldingen zijn van de classifier_nl):

1. Een zin wordt opgesplitst in paren van het huidige karakter en het volgende karakter, die in een lijst worden gestopt. Bij speciale karakters worden deze vervangen door generic karakters



2. Deze lijst wordt door de MapReduce gehaald en omgevormd tot de sorted dictionary



3. De klasse wordt geïnstantieerd
4. De matrix binnen de klasse wordt opgevuld met de dictionary uit de MapReduce door te loopen door alle mogelijke charpairs en te controleren welke value bij de key hoort, gevolgd door deze invoegen in de matrix

	a	b	c	d	e	f	g	h	i	j	...	t	u	v	w	x	y	z		0
a	9018.0	121.0	904.0	1751.0	21.0	681.0	1090.0	232.0	200.0	29.0	...	5067.0	237.0	764.0	57.0	112.0	22.0	31.0	491.0	0.0
b	734.0	356.0	7.0	1.0	3967.0	0.0	0.0	0.0	220.0	0.0	...	191.0	261.0	0.0	0.0	0.0	717.0	3.0	286.0	0.0
c	25.0	0.0	10.0	0.0	106.0	0.0	0.0	6246.0	155.0	0.0	...	10.0	10.0	0.0	0.0	0.0	7.0	0.0	1.0	0.0
d	4970.0	35.0	1.0	348.0	15742.0	4.0	37.0	212.0	4699.0	228.0	...	372.0	648.0	34.0	87.0	0.0	21.0	76.0	4809.0	0.0
e	225.0	1301.0	495.0	2880.0	8751.0	979.0	2405.0	678.0	2723.0	22.0	...	6434.0	940.0	1741.0	831.0	59.0	48.0	1479.0	16793.0	0.0
f	107.0	14.0	1.0	645.0	362.0	368.0	85.0	50.0	213.0	8.0	...	710.0	28.0	11.0	33.0	0.0	13.0	16.0	1289.0	0.0
g	787.0	20.0	0.0	456.0	10418.0	0.0	462.0	296.0	475.0	13.0	...	295.0	93.0	32.0	12.0	0.0	65.0	14.0	3772.0	0.0
h	2631.0	2.0	0.0	2.0	6259.0	0.0	1.0	2.0	957.0	2.0	...	2137.0	692.0	1.0	9.0	0.0	1630.0	15.0	1175.0	0.0
i	111.0	27.0	1078.0	1943.0	8442.0	108.0	2606.0	8.0	7.0	91.0	...	2310.0	18.0	75.0	13.0	5.0	1.0	132.0	297.0	0.0
j	823.0	0.0	0.0	4.0	710.0	1.0	0.0	0.0	82.0	0.0	...	0.0	174.0	2.0	0.0	0.0	13.0	4.0	23.0	0.0
k	654.0	90.0	0.0	16.0	3260.0	1.0	9.0	159.0	532.0	47.0	...	898.0	339.0	7.0	308.0	0.0	10.0	41.0	4648.0	0.0
l	2969.0	29.0	2.0	1371.0	3476.0	400.0	281.0	34.0	1829.0	5.0	...	487.0	317.0	59.0	45.0	0.0	1599.0	67.0	2544.0	0.0
m	2478.0	180.0	0.0	432.0	4305.0	6.0	69.0	49.0	821.0	0.0	...	231.0	85.0	8.0	8.0	0.0	1019.0	20.0	1716.0	0.0
n	1504.0	265.0	81.0	5494.0	2619.0	22.0	4303.0	239.0	2934.0	33.0	...	2384.0	372.0	210.0	162.0	0.0	20.0	292.0	28929.0	0.0
o	76.0	32.0	713.0	573.0	4580.0	1376.0	1027.0	12.0	220.0	3.0	...	1130.0	1792.0	1074.0	10.0	2.0	7.0	64.0	598.0	0.0
p	747.0	22.0	0.0	28.0	1270.0	5.0	160.0	47.0	171.0	19.0	...	229.0	138.0	47.0	13.0	0.0	49.0	38.0	1554.0	0.0
q	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	19.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
r	2081.0	597.0	30.0	2578.0	4631.0	111.0	634.0	363.0	1886.0	12.0	...	1367.0	892.0	418.0	271.0	0.0	741.0	409.0	8884.0	0.0
s	338.0	88.0	2988.0	131.0	615.0	2.0	55.0	123.0	531.0	153.0	...	4891.0	130.0	38.0	45.0	0.0	10.0	57.0	6555.0	0.0
t	1325.0	105.0	3.0	96.0	8279.0	3.0	215.0	274.0	1393.0	214.0	...	728.0	632.0	185.0	381.0	0.0	414.0	91.0	15509.0	0.0
u	23.0	74.0	224.0	609.0	40.0	98.0	442.0	26.0	2138.0	1.0	...	217.0	270.0	138.0	869.0	17.0	7.0	35.0	935.0	0.0
v	4179.0	0.0	0.0	0.0	5865.0	0.0	2.0	0.0	459.0	0.0	...	0.0	100.0	0.0	0.0	0.0	49.0	0.0	1.0	0.0
w	4055.0	5.0	0.0	57.0	3335.0	0.0	1.0	14.0	895.0	0.0	...	18.0	9.0	1.0	1.0	0.0	503.0	0.0	309.0	0.0
x	5.0	0.0	29.0	0.0	10.0	0.0	0.0	0.0	9.0	0.0	...	2.0	0.0	1.0	0.0	2.0	0.0	0.0	77.0	0.0
y	14.0	11.0	1.0	538.0	74.0	181.0	111.0	28.0	3.0	0.0	...	87.0	0.0	233.0	8.0	0.0	0.0	242.0	2685.0	0.0
z	1005.0	0.0	0.0	0.0	3414.0	0.0	0.0	0.0	1378.0	0.0	...	3.0	375.0	0.0	174.0	0.0	1757.0	3.0	4.0	0.0
	4994.0	5157.0	146.0	16669.0	7475.0	377.0	6077.0	9054.0	6911.0	1049.0	...	5172.0	1223.0	9259.0	6960.0	4.0	39.0	7184.0	25025.0	443.0
0	0.0	1.0	0.0	4.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	103.0	997.0
,	381.0	374.0	38.0	531.0	452.0	41.0	341.0	337.0	393.0	138.0	...	1871.0	48.0	241.0	302.0	1.0	4.0	258.0	19415.0	229.0

29 rows × 29 columns

5. De matrix wordt gefilterd om alle pairs die minder dan 1% van het totaal beslaan eruit te halen

	a	b	c	d	e	f	g	h	i	j	...	t	u	v	w	x	y	z		0	,
a	9018.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
b	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
c	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
d	0.0	0.0	0.0	0.0	15742.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
e	0.0	0.0	0.0	0.0	8751.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	16793.0	0.0	0.0
f	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
g	0.0	0.0	0.0	0.0	10418.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
h	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
i	0.0	0.0	0.0	0.0	8442.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
j	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
k	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
l	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
m	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
n	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	28929.0	0.0	5684.0
o	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
p	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
q	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
r	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8884.0	0.0	0.0
s	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	4891.0	0.0	0.0	0.0	0.0	0.0	0.0	6555.0	0.0	0.0
t	0.0	0.0	0.0	0.0	8279.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	15509.0	0.0	0.0
u	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
v	0.0	0.0	0.0	0.0	5865.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
w	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
x	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
y	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
z	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	4994.0	5157.0	0.0	16669.0	7475.0	0.0	6077.0	9054.0	6911.0	0.0	...	5172.0	0.0	9259.0	6960.0	0.0	0.0	7184.0	25025.0	0.0	7066.0
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
,	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	19415.0	0.0	5299.0

29 rows × 29 columns

6. De uiteindelijke matrix bevat enkel nog waarden in de belangrijke charpairs

Als laatste is de code gemaakt om de regels van de opdracht in te laden en in matrices om te zetten. Hierna kan iedere matrix door een compare worden gehaald en berekend met welke classifier matrix deze het meest overeen komt. Hier kwamen de meeste problemen, gezien een < de antwoorden omdraaide…

```
76 nederlandse zinnen
116 engelse zinnen
```

```
all_matrices = []
all_languages = []
testfile = open('letterfrequentietext.txt', 'r', encoding='utf8')
lines = testfile.read().splitlines()
for line in lines:
    chardict = mapreduce_line(split_line(line))
    line_freq = letterfrequency('null')
    line_freq.fill(chardict)
    all_matrices.append(line_freq.matrix)

for matrix in all_matrices:
    all_languages.append(classifier_nl.compare(matrix, classifier_en))

print(all_languages.count('nl'), 'nederlandse zinnen')
print(all_languages.count('en'), 'engelse zinnen')
```