Tarea 1 - Recuperaci?n de tweets basado en contenidos

May 21, 2019

1. Funcion de limpieza

[1]:
```
import re
sw = open("stopwords.txt", 'r').read().split("\n")
def cleanSW(text):
    global sw
    text = re.sub(r'[^[a-zñA-ZÑáéíóúÁÉÍÓÚüÜ ]]*', "", text).replace("_"," ").
 ↪lower()
    clean = []
    for i in text.split(" "):
        if i not in sw:
            clean.append(i)
    return clean
```

2. Índice invertido

[2]:
```
import json
from pprint import pprint
with open("tweets.json") as f:
    data = json.load(f)
n_doc = len(data)
def invertedIndex(data):
    inv = {}
    for i in data:
        text = cleanSW(i["text"])
        for word in text:
            if word in inv:
                #frec_words[word] = frec_words[word]+1
                if i["id"] in inv[word]:
                    inv[word][i["id"]] = inv[word][i["id"]]+1
                else:
                    inv[word][i["id"]] = 1
            else:
                inv[word] = {}
                inv[word][i["id"]] = 1
    for key in inv:
        inv[key]["idf"] = len(inv[key])
    return inv
inv = invertedIndex(data)
```

3. Implementación de TF-IDF

[3]:
```
import math
def tf_idf(n_doc,word, inv,_id):
    if word not in inv:
        return 0
    return (math.log10(n_doc/inv[word]["idf"])+1)*inv[word][_id]
```

4. Implementación de la similitud de cosenos

[4]:
```
def Normalize(vector):
    norm = 0
    for v in vector:
        norm += vector[v]*vector[v]
    norm = math.sqrt(norm)
    for i in vector:
        vector[i] /= norm
    return vector
def cosineScore(Q,inv):
    global n_doc
    Q = cleanSW(Q)
    table = {}
    for word in Q:
        for i in inv[word]:
            if i not in table:
                table[i] = {}
            table[i][word] = tf_idf(n_doc,word,inv,i)

    for doc in table:
        vector = table[doc]
        vector = Normalize(vector)

    query = {}
    for i in Q:
        if i not in query:
            query[i] = 1
        else:
            query[i] += 1
    query = Normalize(query)
    coss = {}
    for i in query:
        for _id in table:
            coss[_id] = 0
            for word in table[_id]:
                if i == word:
                    coss[_id] += query[i]*table[_id][word]
    return coss
```

0.1 Consultas

[5]:
```
def score(query, n):
    global inv
    score1 = cosineScore(Q1,inv)
    scores = {}
    for i in score1:
        if score1[i]:
            scores[i]= score1[i]
    top = sorted(scores, key=lambda x: scores[x], reverse=True)[:n]
    topn = {}
    for i in top:
        topn[i] = scores[i]
    return topn
```

1. "Las pruestas de Muñoz"

[6]:
```
Q1 = "Las propuestas de Muñoz"
score(Q1,10)
```

[6]:
```
{1046263372675788800: 0.7071067811865475,
 1046263792840126464: 0.7071067811865475,
 1046263910540738565: 0.7071067811865475,
 1046263969118408705: 0.7071067811865475,
 1046264126513844224: 0.7071067811865475,
 1046264204745986048: 0.7071067811865475,
 1046264347918577665: 0.7071067811865475,
 1046264584712192000: 0.7071067811865475,
 1046264599341912064: 0.7071067811865475,
 1046264647433834496: 0.7071067811865475}
```

2. "Daniel Urresti y Muñoz"

[15]:
```
Q2 = "Daniel Urresti y Muñoz"
score(Q2, 10)
```

[15]:
```
{1046263372675788800: 0.7071067811865475,
 1046263792840126464: 0.7071067811865475,
 1046263910540738565: 0.7071067811865475,
 1046263969118408705: 0.7071067811865475,
 1046264126513844224: 0.7071067811865475,
 1046264204745986048: 0.7071067811865475,
 1046264347918577665: 0.7071067811865475,
 1046264584712192000: 0.7071067811865475,
 1046264599341912064: 0.7071067811865475,
 1046264647433834496: 0.7071067811865475}
```

3. “Mentiras y sicosociales”

[18]:
```
Q3 = "Mentiras y sicosociales"
score(Q3,10)
```

[18]:
```
{1046263372675788800: 0.7071067811865475,
 1046263792840126464: 0.7071067811865475,
 1046263910540738565: 0.7071067811865475,
 1046263969118408705: 0.7071067811865475,
 1046264126513844224: 0.7071067811865475,
 1046264204745986048: 0.7071067811865475,
 1046264347918577665: 0.7071067811865475,
 1046264584712192000: 0.7071067811865475,
 1046264599341912064: 0.7071067811865475,
 1046264647433834496: 0.7071067811865475}
```