Buscador de texto en Wikipedia por palabras claves usando VSM y similitud coseno

Yenner Robayo

Miguel Pedraza

Used technology

python

git

GitHub

VS Code

jupyter

spyder

ANACONDA®

Data set

500 1000

WIKIPEDIA

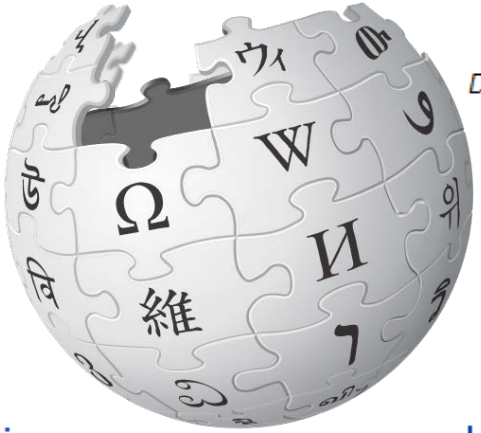
English
The Free Encyclopedia
4 465 000+ articles

Español
La enciclopedia libre
1 086 000+ artículos

日本語
フリー百科事典
898 000+ 記事

Deutsch
Die freie Enzyklopädie
1 694 000+ Artikel

Русский
Свободная энциклопедия
1 094 000+ статей



Français
L'encyclopédie libre
1 482 000+ articles

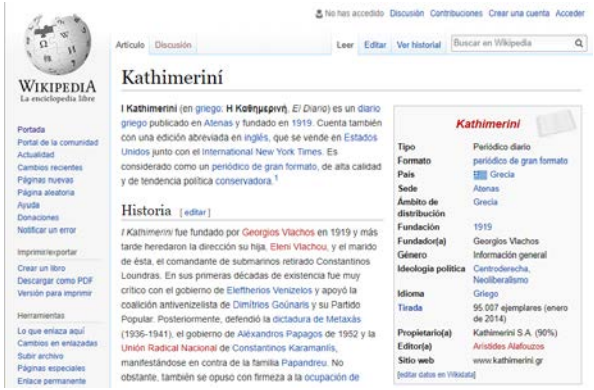
Italiano
L'enciclopedia libera
1 104 000+ voci

Português
A enciclopédia livre
821 000+ artigos

Polski
Wolna encyklopedia
1 032 000+ haseł

中文
自由的百科全書
754 000+ 條目

Wikipedia API and web scraping



wikipedia 1.4.0

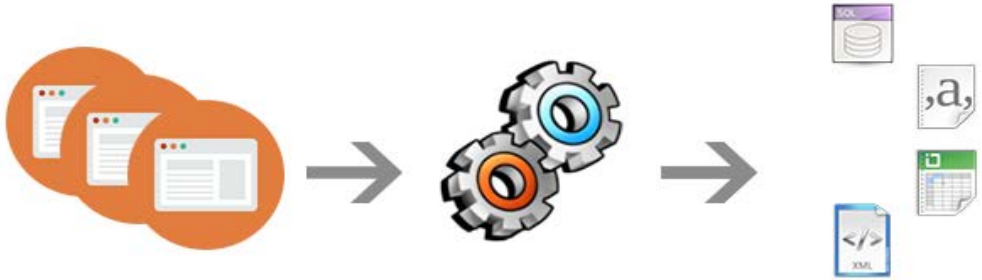
Wikipedia API for Python

{ api }

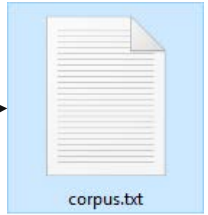

```
['Kathimeriní', 'Poetas andaluces de ahora', 'Blue
[0]: 'Kathimeriní'
[1]: 'Poetas andaluces de ahora'
[2]: 'Blue Mound (Illinois)'
[3]: 'Desarrollo web'
[4]: 'Duracell'
[5]: 'Minshutō'
[6]: "Ain't Life Grand"
[7]: 'Temporada 2016 del Campeonato Mundial
[8]: 'Daniela Bousso'
[9]: 'Conjetura de Artin sobre raíces primi
[10]: 'Aftermath (película de 2017)'
[11]: 'Vectorscopio'
[12]: 'Teresa de Cofrentes'
[13]: 'Sumner (Misisipi)'
[14]: 'Renacer de Arauco'
[15]: 'Fido (futbolista)'
[16]: 'Wenslingen'
[17]: 'Municipio de Shrewsbury (condado de
```

Get corpus from wikipedia pages

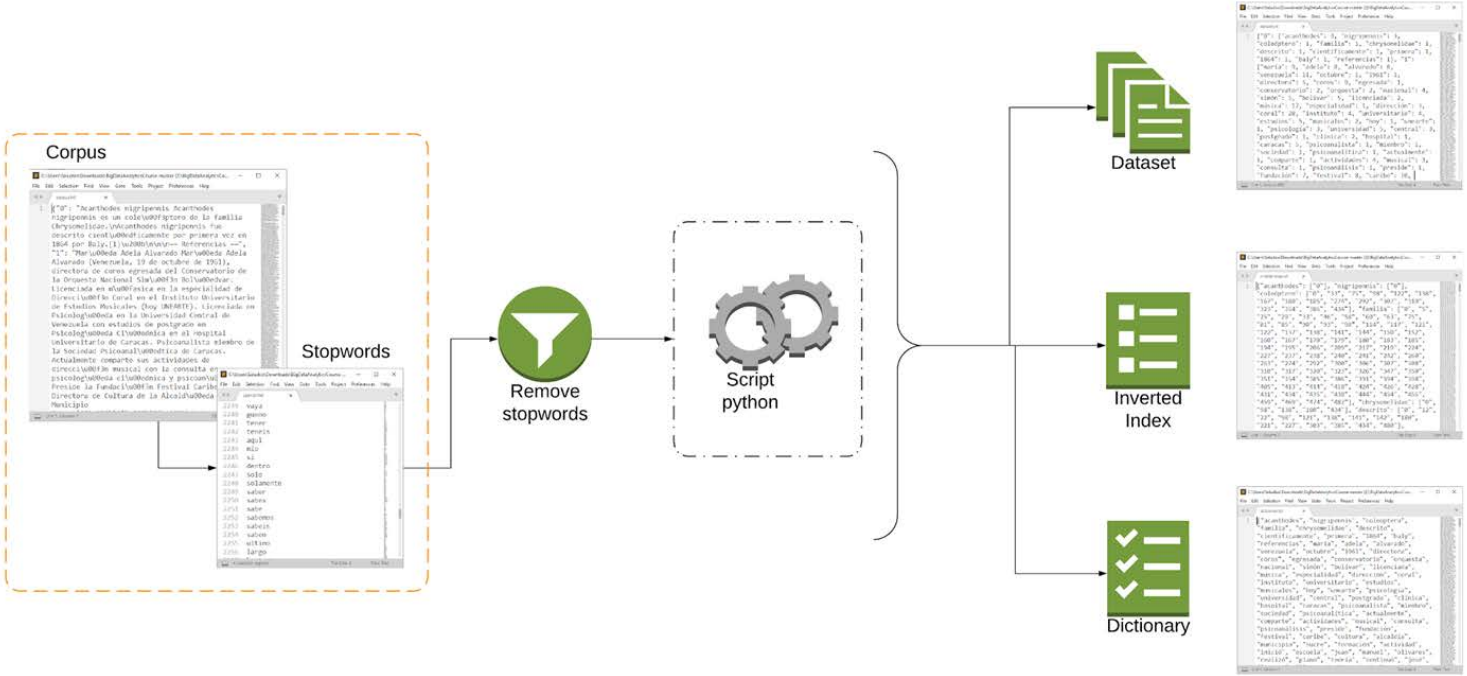
```
wikipedia.set_lang("es")
pagesList = wikipedia.random(n_len)

for i, page in enumerate(pagesList):
    print("Processing page...", end=' ')
    try:
        foundPage = wikipedia.page(page)
    except:
        corpus[i] = ""
        print("page %i is void" % i)
        continue

    corpus[i] = foundPage.title + " " + foundPage.content
    print("page %i added" % i)
    sleep(0.1)  # evita ser bloquedo por multiples peticiones
```



```
Processing page... page 0 added, page is Seren Gibson
Processing page... page 1 added, page is Íñigo López de Mendoza y Mendoza,
Processing page... page 2 added, page is Kepler-191b
Processing page... page 3 added, page is EusLinux
Processing page... page 4 added, page is Dominios franceses de Santa Elena
Finish
```



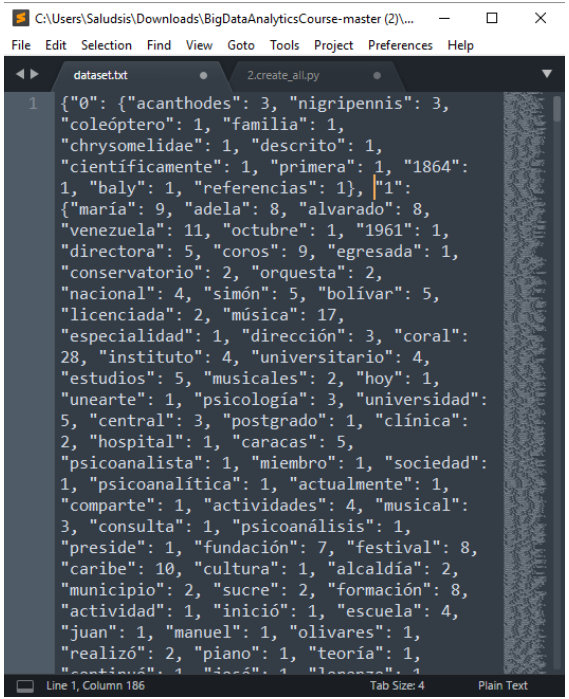
Create dataset, inverted index and dictionary



Generate dataset with frequency

{Documento n: { “término”: *frecuencia*, ……..} }

```
for key, doc in corpus.items():
    doc_listwords = []
    for word in RemoveSymbols(doc.lower()).split():
        if word not in stopwords and isNotEmpty(word):
            doc_listwords.append(word)
            if word in inverterIndex:
                if key not in inverterIndex[word]:
                    inverterIndex[word].append(key)
            else:
                inverterIndex[word] = [key]
    dataset[key] = Counter(doc_listwords)
```




```
{"0": {"acanthodes": 3, "nigripennis": 3,
"coleóptero": 1, "familia": 1,
"chrysomelidae": 1, "descrito": 1,
"científicamente": 1, "primera": 1, "1864":
1, "baly": 1, "referencias": 1}, "1":
{"maría": 9, "adela": 8, "alvarado": 8,
"venezuela": 11, "octubre": 1, "1961": 1,
"directora": 5, "coros": 9, "egresada": 1,
"conservatorio": 2, "orquesta": 2,
"nacional": 4, "simón": 5, "bolívar": 5,
"licenciada": 2, "música": 17,
"especialidad": 1, "dirección": 3, "coral":
28, "instituto": 4, "universitario": 4,
"estudios": 5, "musicales": 2, "hoy": 1,
"unearte": 1, "psicología": 3, "universidad":
5, "central": 3, "postgrado": 1, "clínica":
2, "hospital": 1, "caracas": 5,
"psicoanalista": 1, "miembro": 1, "sociedad":
1, "psicoanalítica": 1, "actualmente": 1,
"comparte": 1, "actividades": 4, "musical":
3, "consulta": 1, "psicoanálisis": 1,
"preside": 1, "fundación": 7, "festival": 8,
"caribe": 10, "cultura": 1, "alcaldía": 2,
"municipio": 2, "sucre": 2, "formación": 8,
"actividad": 1, "inició": 1, "escuela": 4,
"juan": 1, "manuel": 1, "olivares": 1,
"realizó": 2, "piano": 1, "teoría": 1,
```

Generate Inverted index

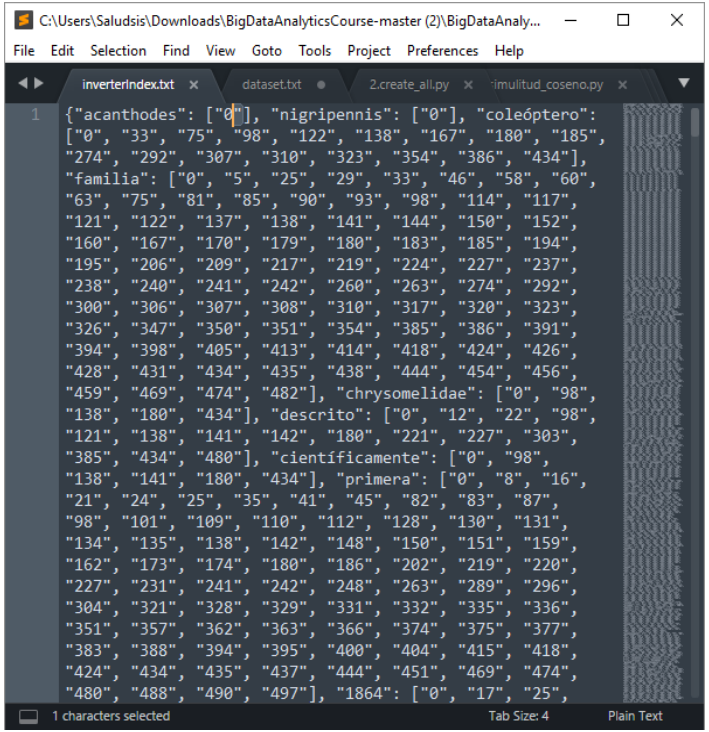
{término n: [“doc 1”, “doc 2”, “doc 3”, …, “doc m”] }

ID	Text
1	Baseball is played during summer months.
2	Summer is the time for picnics here.
3	Months later we found out why.
4	Why is summer so hot here

↑ Sample document data

Term	Freq	Document ids
baseball	1	[1]
during	1	[1]
found	1	[3]
here	2	[2], [4]
hot	1	[4]
is	3	[1], [2], [4]
months	2	[1], [3]
summer	3	[1], [2], [4]
the	1	[2]
why	2	[3], [4]

Dictionary and posting lists →




```
{"acanthodes": ["0"], "nigripennis": ["0"], "coleóptero":
["0", "33", "75", "98", "122", "138", "167", "180", "185",
"274", "292", "307", "310", "323", "354", "386", "434"],
"familia": ["0", "5", "25", "29", "33", "46", "58", "60",
"63", "75", "81", "85", "90", "93", "98", "114", "117",
"121", "122", "137", "138", "141", "144", "150", "152",
"160", "167", "170", "179", "180", "183", "185", "194",
"195", "206", "209", "217", "219", "224", "227", "237",
"238", "240", "241", "242", "260", "263", "274", "292",
"300", "306", "307", "308", "310", "317", "320", "323",
"326", "347", "350", "351", "354", "385", "386", "391",
"394", "398", "405", "413", "414", "418", "424", "426",
"428", "431", "434", "435", "438", "444", "454", "456",
"459", "469", "474", "482"], "chrysomelidae": ["0", "98",
"138", "180", "434"], "descrito": ["0", "12", "22", "98",
"121", "138", "141", "142", "180", "221", "227", "303",
"385", "434", "480"], "científicamente": ["0", "98",
"138", "141", "180", "434"], "primera": ["0", "8", "16",
"21", "24", "25", "35", "41", "45", "82", "83", "87",
"98", "101", "109", "110", "112", "128", "130", "131",
"134", "135", "138", "142", "148", "150", "151", "159",
"162", "173", "174", "180", "186", "202", "219", "220",
"227", "231", "241", "242", "248", "263", "289", "296",
"304", "321", "328", "329", "331", "332", "335", "336",
"351", "357", "362", "363", "366", "374", "375", "377",
"383", "388", "394", "395", "400", "404", "415", "418",
"424", "434", "435", "437", "444", "451", "469", "474",
"480", "488", "490", "497"], "1864": ["0", "17", "25",
```

Generate dictionary

```
dictionary =  list(inverterIndex.keys())
```

```
with open('dictionary.txt', 'w', encoding='utf-8') as file:
    json.dump(dictionary, file, ensure_ascii=False)
```

C:\Users\Saludsis\Downloads\BigDataAnalyticsCourse-master (2)\...

File Edit Selection Find View Goto Tools Project Preferences Help

dictionary.txt | inverterIndex.txt | dataset.txt | all.py


```
["acanthodes", "nigripennis", "coleóptero",
"familia", "chrysomelidae", "descrito",
"científicamente", "primera", "1864", "baly",
"referencias", "maría", "adela", "alvarado",
"venezuela", "octubre", "1961", "directora",
"coros", "egresada", "conservatorio", "orquesta",
"nacional", "simón", "bolívar", "licenciada",
"música", "especialidad", "dirección", "coral",
"instituto", "universitario", "estudios",
"musicales", "hoy", "unearte", "psicología",
"universidad", "central", "postgrado", "clínica",
"hospital", "caracas", "psicoanalista", "miembro",
"sociedad", "psicoanalítica", "actualmente",
"comparte", "actividades", "musical", "consulta",
"psicoanálisis", "preside", "fundación",
"festival", "caribe", "cultura", "alcaldía",
"municipio", "sucre", "formación", "actividad",
"inició", "escuela", "juan", "manuel", "olivares",
"realizó", "piano", "teoría", "continuó", "josé",
"lorenzo", "llamozas", "privados", "beatriz",
"lockhart", "modesta", "bor", "harriet", "serr",
"obtuvo", "título", "posteriormente",
"licenciatura", "tutoría", "maestro", "alberto",
"grau", "coralista", "cantoría", "schola",
"cantorum", "dirigidas", "guinand",
"respectivamente", "participado", "talleres",
"cursos", "directores", "músicos", "talla",
"carlos", "pinto", "fonseca", "brasil", "luigi",
```


Line 1, Column 1 Tab Size: 4 Plain Text

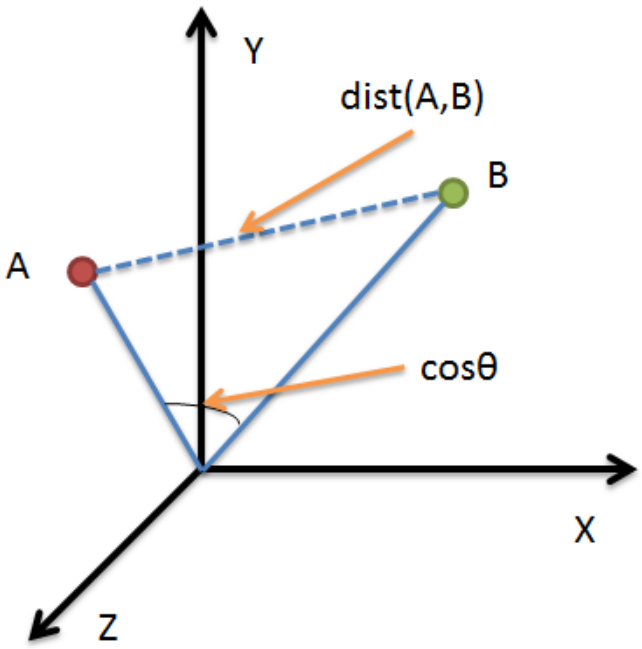
Generate TF-IDF

$$\underbrace{TF\text{-}IDF_{(n,d)}}_{\text{Peso de un término (n) en un documento (d)}} = \underbrace{TF_{(n,d)}}_{\text{Frecuencia de aparición de un término (n) en un documento (d)}} \times \underbrace{IDF_{(n)}}_{\text{Factor IDF de un término (n)}}$$

$$\boldsymbol{TF - IDF} = TF_t \times \log \frac{N}{DF_t}$$

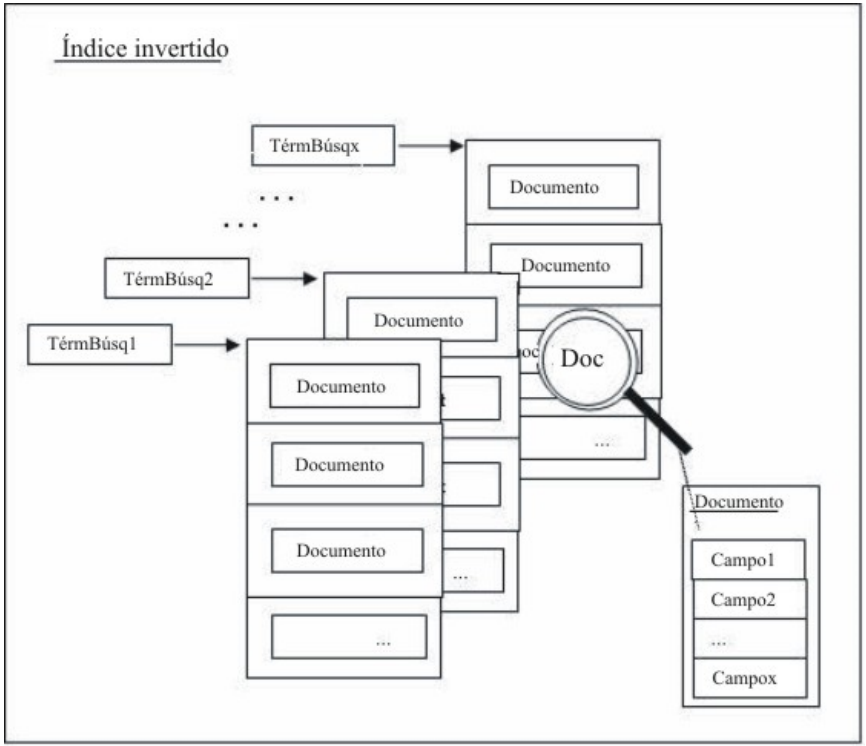
```python
doc_lenght = len(dataset)

for word, docs in inverterIndex.items():
    inv_frec = log(doc_lenght/len(docs))
    for doc_id, doc_words in dataset.items():
        eq = doc_words.get(word, 0)*inv_frec
        if doc_id in tfidf:
            tfidf[doc_id].append(eq)
        else:
            tfidf[doc_id] = [eq]
    inv_frec_vector.append(inv_frec)

dictionary =  list(inverterIndex.keys())
```

Cosine similarity




```python
def cosine_similarity(vectorSpace1, vectorSpace2):
    numerator = 0
    sumxx, sumyy = 0, 0
    for i in range(len(vectorSpace1)):
        x = vectorSpace1[i]
        y = vectorSpace2[i]
        sumxx += x*x
        sumyy += y*y
        numerator += x*y
    return numerator/math.sqrt(sumxx*sumyy)
```

Search with the inverted index method




```python
def search(query):
    dataset = {}
    tfidf = [0] * len(inverdIndex)
    cos_sim = {}

    doc_listwords = []
    for word in utils.removeSymbols(query.lower()).split():
        if word not in stopwords and utils.isNotEmpty(word):
            doc_listwords.append(word)
    dataset = Counter(doc_listwords)

    for id, word in enumerate(inverdIndex.keys()):
        if word in dataset:
            tfidf[id] = dataset.get(word, 0)*inv_frec_vector[id]

    for word in dataset.keys():
        if word in inverdIndex:  # si la palabra esta en el index invertido
            for key in inverdIndex.get(word):
                if key not in cos_sim:
                    cos_sim[key] = utils.cosineSimilarity(tfidf, allTfidf[key])
    return cos_sim
```

Demonstration 500 vs 1000

2.create_all.py

```python
print("start algorithm")
t0 = time()
for id_doc, document in corpus.items():
    doc_listwords = []
    for word in utils.removeSymbols(document.lower()).split():
        if word not in stopwords and utils.isNotEmpty(word):
            doc_listwords.append(word)
            if word in inverterIndex:
                if id_doc not in inverterIndex[word]:
                    inverterIndex[word].append(id_doc)
            else:
                inverterIndex[word] = [id_doc]
    dataset[id_doc] = Counter(doc_listwords)

doc_lenght = len(dataset)

for word, docs in inverterIndex.items():
    inv_frec = log(doc_lenght/len(docs))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
start algorithm
done in 18.653s.
```

18.653s

2.create_all.py

```python
for word, docs in inverterIndex.items():
    inv_frec = log(doc_lenght/len(docs))
    for doc_id, doc_words in dataset.items():
        eq = doc_words.get(word, 0)*inv_frec
        if doc_id in tfidf:
            tfidf[doc_id].append(eq)
        else:
            tfidf[doc_id] = [eq]
    inv_frec_vector.append(inv_frec)

print("done in %0.3fs." % (time() - t0))

dictionary =  list(inverterIndex.keys())

with open('dataset.txt', 'w', encoding='utf-8') as file:
    json.dump(dataset, file, ensure_ascii=False)

with open('dictionary.txt', 'w', encoding='utf-8') as file:
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
start algorithm
done in 43.074s.
```

43.074s

Result of the query

500

```python
""" main """
# abrir documentos con los documentos procesados
inverdIndex, stopwords, allTfidf, corpus, inv_frec_vector = opendDocuments()

query = "universidad de los llanos"

# obtiene el tiempo actual
t0 = time()

# realiza la consulta
docs = search(query)

```

```python
# muestra resultados
print()
print("Tiempo total de la busqueda %0.3fs." % totalTime)
print("Total de documentos encontrados: %d" % len(docs))
print("La consulta es: " + query)
print()

i = 0
for key in sorted(docs, key=docs.get, reverse=True):
    print("Documento encontrado: #%s, cs: %f" % (i, docs[key]))
    print('Documento #%s: %.500s' % (key, corpus[key]))
    print()
    i += 1

    if (i > 10):
        break
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Tiempo total de la busqueda 0.767s.
Total de documentos encontrados: 55
La consulta es: universidad de los llanos

Documento encontrado: #0, cs: 0.040172
Documento #76: José Antonio Balseiro José Antonio Balseiro (Córdoba, 29 de marzo de 1919 - Bariloche, 26 de marzo de 1962) fue un imp
ortante físico argentino.
José Antonio Balseiro nació en la ciudad de Córdoba el 29 de marzo de 1919, cuarto hijo de Antonio Balseiro, quien había emigrado de
 España en su adolescencia, y de Victoria Lahore, argentina de origen francés.
En 1933 ingresa al Colegio Nacional de Monserrat dependiente de la Universidad de Córdoba, de donde egresa con el título de bachiller
 en 1938.
```

Result of the query

1000

3.search.py

```python
    for word in dataset.keys():
        if word in inverdIndex:  # si la palabra esta en el index invertido
            for key in inverdIndex.get(word):
                if key not in cos_sim:
                    cos_sim[key] = utils.cosineSimilarity(tfidf, allTfidf[key])
    return cos_sim


""" main """
# abrir documentos con los documentos procesados
inverdIndex, stopwords, allTfidf, corpus, inv_frec_vector = opendDocuments()

query = "universidad de los llanos"

# obtiene el tiempo actual
t0 = time()

# realiza la consulta
docs = search(query)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Tiempo total de la busqueda 2.044s.
Total de documentos encontrados: 111
La consulta es: universidad de los llanos

Documento encontrado: #0, cs: 0.069000
Documento #559: José Santos Degollado El general José Nemesio Francisco Degollado Sánchez, mejor conocido como Santos Degollado (Gua
najuato, México, 30 de octubre de 1811 - Llanos de Salazar, Estado de México, 15 de junio de 1861) se le conoce como el Héroe de las
 Derrotas, aunque tenía la rara habilidad de poder convocar posterior a sus derrotas, nuevos ejércitos a su mando. Fue un militar y
 político mexicano que se dedicó además, a la geografía, filosofía, física, gramática, matemáticas, jurisprudencia, histo
```

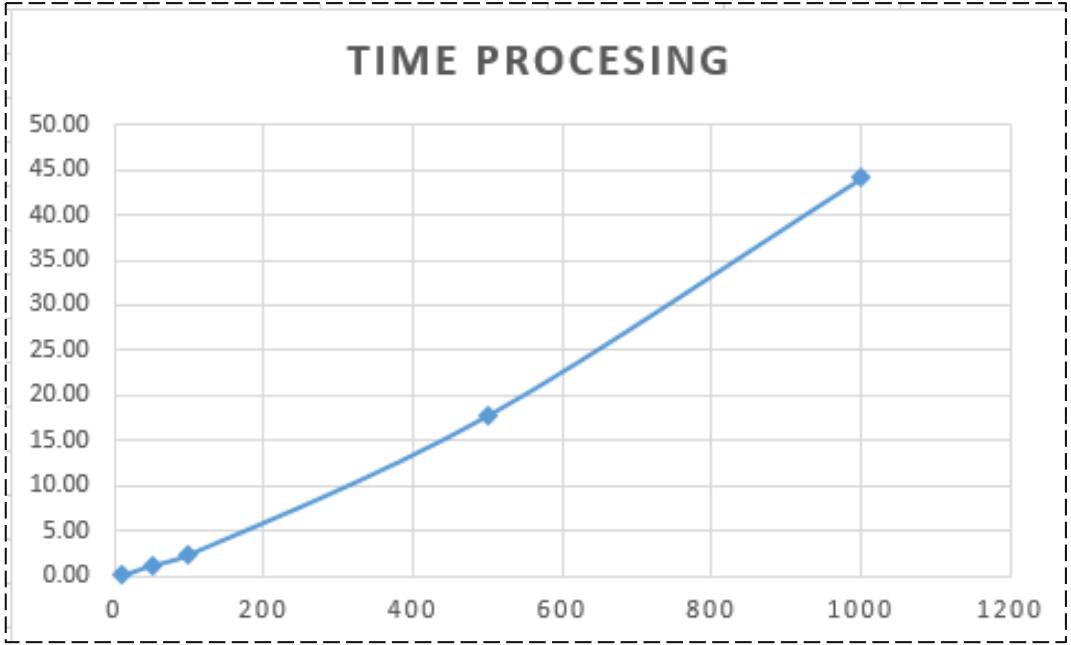
Results comparison

Query = "universidad de los llanos"			
Corpus size	**Execution time**	**Search time**	**Documents found**
500	18.653s	0.767s	55
1000	43.074s	2.044s	111

Performance

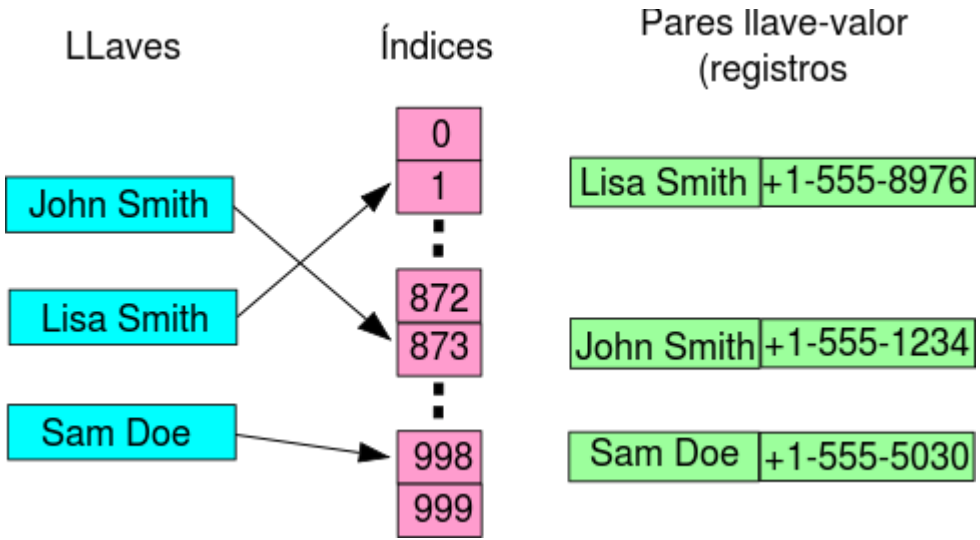
PC Specs	
processor	Intel i5-2410M, 2 cores, 4 Threads
processor speed	Frecuency: 2.30 GHz, Cache: 3 MB
memory	8 ram, 1333 Mhz DDR3
hard disk	1 TB 3 Gb/s 5400 rpm
S.O.	Win 10 64 bits

Corpus size	Time procesing
10	0.06
50	1.22
100	2.42
500	17.81
1000	44.09



Advantage of using hash tables

The main advantage is the speed of access while searching



Demo in Jupyter

```
Ingrese la consulta:
universidad de los llanos
```

```
Ingrese la consulta: universidad de los llanos

Tiempo total de la busqueda 1.986s.
Total de documentos encontrados: 111
La consulta es: universidad de los llanos

Documento encontrado: #0, cs: 0.069000
Documento #559: José Santos Degollado El general José Nemesio F
rancisco Degollado Sánchez, mejor conocido como Santos Degollad
o (Guanajuato, México, 30 de octubre de 1811 - Llanos de Salaza
r, Estado de México, 15 de junio de 1861) se le conoce como el
Héroe de las Derrotas, aunque tenía la rara habilidad de poder
convocar posterior a sus derrotas, nuevos ejércitos a su mando.
Fue un militar y político mexicano que se dedicó además, a la g
eografía, filosofía, física, gramática, matemáticas, jurisprude
ncia, histo

Documento encontrado: #1, cs: 0.058880
Documento #807: Universidad de la medicina y ciencias de la sal
```

jupyter 4.final_project

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

Algoritmo de Búsqueda

In [3]:

```python
query = input("Ingrese la consulta: ")

# obtiene el tiempo actual
t0 = time()

# realiza la consulta
docs = search(query)

# obtiene el tiempo total de la busqueda
totalTime = time()-t0

# muestra resultados
print()
print("Tiempo total de la busqueda %0.3fs." % totalTime)
print("Total de documentos encontrados: %d" % len(docs))
print("La consulta es: " + query)
print()

i = 0
for key in sorted(docs, key=docs.get, reverse=True):
    print("Documento encontrado: #%s, cs: %f" % (i, docs[key]))
    print('Documento #%s: %.500s' % (key, corpus[key]))
    print()
    i += 1
```

Gracias!