M1 Report

February 20, 2022

1 Code for constructing the index

~~~
```python
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import stopwords
from bs4 import BeautifulSoup
import sqlalchemy
import nltk
import re
import os
from tqdm import tqdm
import multiprocessing


def get_wordnet_pos(treebank_tag):
    """
        Converts treebank tags to Wordnet POS names.

        Whole function (modified): Suzana
        https://stackoverflow.com/questions/15586721/wordnet-lemmatization-and-pos-tagging-in-

        RESOURCE
        All possible tags : https://stackoverflow.com/questions/15388831/what-are-all-possible
    """

    if treebank_tag.startswith('J'):
        return 'a'
    elif treebank_tag.startswith('V'):
        return 'v'
    elif treebank_tag.startswith('N'):
        return 'n'
    elif treebank_tag.startswith('R'):
        return 'r'
    else:
        return 'n'  # treat token as noun.
~~~

```
def removeStopwords(word_list):
    """
        Removes English stopwords from a list of words.
        List comprehension done by : Daren Thomas
        https://stackoverflow.com/questions/5486337/how-to-remove-stop-words-using-nltk-or-pyt
    """

    return [word for word in word_list if word not in stopwords.words('english')]


def lemmatize(word_list):
    """
        Lemmatizes a list of two tuples with token and a part of speech pairs.
    """

    lemmatizer = WordNetLemmatizer()
    return [lemmatizer.lemmatize(word, pos=tag) for word, tag in word_list]


def clean(word_list):
    """
        Cleans a list of words by removing stop words and applying lemmatization to each word.
    """
    word_list = removeStopwords(word_list)  # Removing stop words
    treebank_tags = nltk.pos_tag(word_list)  # Identify parts of speech, need to simplify tags

    wordnet_tags = []  # New list with simplified tags, not sure why this takes two steps from

    for token, tag in treebank_tags:
        wordnet_tags.append((token, get_wordnet_pos(tag)))

    return lemmatize(wordnet_tags)  # Return lemmatized words


def getKeywords(soup):
    """
        Gets all keywords (headers, bolded, titles, listed items) from an HTML document, these
        carry more weight than the words found in the text.

        Returns a list of keywords, empty list if there aren't any.
            Could be enhanced by giving weight to each word, e.g. Title tokens have more weigh

        Regex found by user phd,
        https://stackoverflow.com/questions/45062534/how-to-grab-all-headers-from-a-website-us

        HTML stripping : Jorge Galvis
        https://medium.com/@jorlugaqui/how-to-strip-html-tags-from-a-string-in-python-7cb81a2b
        Puncuation stripping : rmalouf
        https://stackoverflow.com/questions/15547409/how-to-get-rid-of-punctuation-using-nltk-t
    """

    raw_keywords = soup.find_all(
        [re.compile('^h[1-6]$'), 'b', 'title', 'li'])  # Regex to find all keywords, see citati

    keywords = []

    # Stripping keyword tags
    for raw_keyword in raw_keywords:
        keyword = raw_keyword.get_text(" ", strip=True).lower().replace('\n', " ")

        tokenizer = nltk.tokenize.RegexpTokenizer(
            r'\w+\'?\w*')  # Getting rid of unnecessary punctuation (preserving apostrophes)
        keywords.extend(tokenizer.tokenize(keyword))

    return clean(keywords)


def getTokens(soup):
    """
        Gets all tokens from html, removes stop words and lemmatizes. Returns list of tokens.
    """

    text = soup.get_text(" ", strip=True).lower().replace('\n', " ")

    tokenizer = nltk.tokenize.RegexpTokenizer(
        r'\w+\'?\w*')  # Getting rid of unnecessary punctuation (preserving apostrophes)
    tokens = tokenizer.tokenize(text)

    return clean(tokens)


def mult_dir(dir):
    root_directory = 'C:/Users/aKost/Desktop/2021-2022/WINTER 2022/CS 121 - Information Retriev
    engine = sqlalchemy.create_engine('postgresql+psycopg2://postgres:qrT90!xvnpc@localhost/cs1
    for file in tqdm(os.listdir(os.path.join(root_directory, dir))):  # Iterating over files
        with open(os.path.join(root_directory, dir, file), 'r', encoding="utf8") as f:  # Openi
            f = f.read()
            soup = BeautifulSoup(f, 'html.parser')

            path = os.path.join(root_directory, dir, file)

            keywords = getKeywords(soup)  # Getting keywords
            tokens = getTokens(soup)  # Getting all text tokens

            insertTokens(tokens, path, dir, file, engine)
```

~~~
            insertTokens(keywords, path, dir, file, engine, isKeyword=True)

def constructIndex(root_directory):
    """
        Gets all the html in the specified path to extract tokens and stores them in a Postgre
    """
    engine = sqlalchemy.create_engine('postgresql+psycopg2://postgres:qrT90!xvnpc@localhost/cs

    for subdir, dirs, files in os.walk(root_directory):  # Iterating over folders in WEBPAGES_
        a_pool = multiprocessing.Pool()
        a_pool.map(mult_dir, dirs)

    # engine.execute("CREATE INDEX inv_idx ON Tokens USING gin(token)")


def insertTokens(tokens, full_path, dir, file, engine, isKeyword=False, frequency=1):
    for token in tokens:

        token_info = (
            token,
            isKeyword,  # isKeyword
            full_path,
            dir,
            file,
            frequency
        )

        engine.execute(
            "INSERT INTO Tokens VALUES (%s, %s, %s, %s, %s, %s) ON CONFLICT (token, fullPath)
            token_info)


if __name__ == "__main__":

    rootdir = 'C:/Users/aKost/Desktop/2021-2022/WINTER 2022/CS 121 - Information Retrieval/pro

    constructIndex(rootdir)
~~~

2 SQL setup

~~~
```sql
SET CLIENT_ENCODING = "utf8";

DROP TABLE IF EXISTS Tokens;
CREATE TABLE IF NOT EXISTS Tokens
(
~~~

````
    token         TEXT,
    isKeyword    BOOLEAN,
    fullPath    TEXT,
    dir            VARCHAR(5),
    file        VARCHAR(5),
    frequency    INTEGER,
    PRIMARY KEY (Token, fullPath)
);

CREATE EXTENSION IF NOT EXISTS pg_trgm ;
CREATE INDEX IF NOT EXISTS tkn_trgm_idx ON Tokens USING gin(token gin_trgm_ops);
````

3 SQL Querying

~~~
```sql

-- NUMBER OF DOCUMENTS
SELECT COUNT(DISTINCT t.fullPath)
FROM Tokens t;

-- NUMBER OF UNIQUE WORDS

SELECT COUNT(DISTINCT t.token)
FROM Tokens t;

--- INDEX SIZE ---
select pg_indexes_size('Tokens');

-- QUERIES --

-- "Informatics" query

SELECT *
FROM Tokens t
WHERE t.token = 'informatics'
ORDER BY t.Frequency DESC
LIMIT 20;

-- COUNT
SELECT COUNT(t.token)
FROM Tokens t
WHERE t.token = 'informatics'
GROUP BY t.token;


-- "Mondego" query
~~~

```
SELECT *
FROM Tokens t
WHERE t.token = 'mondego'
ORDER BY t.Frequency DESC
LIMIT 20;


-- COUNT
SELECT COUNT(t.token)
FROM Tokens t
WHERE t.token = 'mondego'
GROUP BY t.token;


-- "Irvine" query

SELECT *
FROM Tokens t
WHERE t.token = 'irvine'
ORDER BY t.Frequency DESC
LIMIT 20;

    -- COUNT
SELECT COUNT(t.token)
FROM Tokens t
WHERE t.token = 'irvine'
GROUP BY t.token;


-- "artificial intelligence" query

SELECT *
FROM Tokens t
WHERE t.token = 'artificial' OR t.token = 'intelligence'
ORDER BY t.frequency DESC
LIMIT 20;


-- COUNT
SELECT COUNT(t.token)
FROM Tokens t
WHERE t.token = 'artificial' OR t.token = 'intelligence'
GROUP BY t.token;


-- "computer science" query
```

```
SELECT *
FROM Tokens t
WHERE t.token = 'computer' OR t.token = 'science'
ORDER BY t.frequency DESC
LIMIT 20;


-- COUNT
SELECT COUNT(t.token)
FROM Tokens t
WHERE t.token = 'computer' OR t.token = 'science'
GROUP BY t.token;
```