Publication and Management System (PUMAS) CSI 605

Ofentse Jabari, Boipelo Mosetho, Ontiretse Ishmael

UNIVERSITY OF BOTSWANA
Department
Of
Computer Science

Where innovation starts

December 1, 2017

Introduction

Main Objective

Below are the fully implemented and extensively tested modules of PUMAS

1. Document Uploading.
2. Document search (with different search criterias i.e. title, authors, department, type)
3. Downloading of documents.
4. Plagiarism checker
5. User Authentication

Modules/Packages Used

```python
from collections import Counter
from nltk.corpus import stopwords
from docx import Document
from nltk.stem import WordNetLemmatizer
import nltk.tokenize as tk
```

Converting document to plain text

```
document = Document(file_path)
document_paragraphs = []
for p in document.paragraphs:
    document_paragraphs.append(p.text)
```

(Excluding pictures, tables or any other graphical components.)

```
''' A list containing a list of sentences '''
sentences_list = [tk.sent_tokenize(s.lower())
                  for s in document_paragraphs]

''' Remove empty lists (Empty lines) '''
sentences_list = [s for s in sentences_list
                  if len(s) != 0]
```

Converting document to plain text

Example (before removing empty lines):

['01. Introduction', ' ', 'this chapter is all about related literature review ...', 'the literature review covers, the role of music...', ...]

```
        ''' Convert the list into a list of sentences
        (from a list of list of sentences)'''
        new_list = []
        [[new_list.append(s) for s in l] for l in sentences_list]

        ''' Filter out short sentences (e.g. 1.1 purpose of
         the system ). This kind of sentences are normaly
         chapters or heading and don't have that much impact
         in what we are trying to solve. Strip trailing and
         leading white spaces '''
        new_list = [s.strip() for s in new_list if len(s)> 50]
```

Creating sentence tokens

Create sentence lexicon of tokenized (and have variant forms of the same word) sentences.

```
common_words = document_common_words(sentence_lists)
stop_words = stopwords.words('english')
sentence_lexicon = []

for sentences in sentence_lists:
    lexicon = tk.word_tokenize(sentences.lower())
    ''' Remove common words '''
    lexicon = [word for word in lexicon
                    if word not in common_words]
    ''' Remove stop words '''
    lexicon = [word for word in lexicon
                    if word not in stop_words]
    ''' Lemmatize '''
    lemmatizer = WordNetLemmatizer()
    lexicon = [lemmatizer.lemmatize(word)
                    for word in lexicon]
```

Comparing two sentences

sentence = ['chapter', 'related', 'literature', 'review', 'extent', 'music', 'promote', 'social']
sentence2 = ['music', 'chapter', 'field', 'extend', 'associated', 'literature', '.']

sentence_match = [1, 1, 0, 1, 0, 1, 0, 0]

```
                ''' count number of 1's in score_vector '''
                score = (sentence_match.count(1)/len(sentence))*100
                sent_perc_matches.append(score)

        return (max(sent_perc_matches))
```

sent_perc_matches = [45, 10, 0, 1.5, 0, 0, 0, 30]

Comparing two sentences

$$\text{scores} = [45, 50, 40, 15, 70, 55, 10, 30]$$

```
scores = []
for sentence in new_document_lexicon:
    sc = sentence_similarity(existing_document_lexicon,
                             sentence)
    scores.append(sc)

return float('%.1g' % (sum(scores)/len(scores)))
```