Lab 5: Implementation of Query Expansion based on synonyms using Wordnet

Aim: Implement the query expansion using wordnet for the query entered by user. Your program should suggest the Expansion words for the original query entered by user.

Theory:

Query expansion (QE) is the process of reformulating a given query to improve retrieval performance in information retrieval operations, particularly in the context of query understanding.

Query expansion involves evaluating a user's input (what words were typed into the search query area) and expanding the search query to match additional documents. Query expansion involves techniques such as:

- Finding synonyms of words, and searching for the synonyms as well
- Finding semantically related words (e.g. antonyms, meronyms, hyponyms, hypernyms)
- Finding all the various morphological forms of words by stemming each word in the search query
- Fixing spelling errors and automatically searching for the corrected form or suggesting it in the results
- Re-weighting the terms in the original query

For example, given the query “bike”, we can identify “bicycle” as a synonym and add it to the query, matching both bikes and bicycles in the corpus. This process of adding new words to the original query is called query expansion. The added words are called expansion words.

Global and Local analysis of documents

Various statistical approaches have been proposed to build custom thesauri using data from the corpus. The idea of leveraging the whole corpus for query expansion is called Global analysis.

In contrast, if we only use a select few documents from corpus for query expansion, it is called Local analysis.

In this practical session we are going to enter a search query and expanding this query based on synonym and antonyms using wordnet corpus from NLTK.

```
from nltk.corpus import wordnet
from nltk.tokenize import word_tokenize
from collections import defaultdict
import pprint
```

```
syns = wordnet.synsets("hesitate")
print(syns[0].name())
print(syns[0].lemmas()[0].name())
print(syns[0].definition())
print(syns[0].examples())
```

We will get following result

hesitate.v.01
hesitate
pause or hold back in uncertainty or unwillingness
['Authorities hesitate to quote exact figures']

Now will print synonyms and antonyms of single word using wordnet

```
synonyms = []
antonyms = []
for syn_set in wordnet.synsets("allow"):
 for l in syn_set.lemmas():
  synonyms.append(l.name())
  if l.antonyms():
   antonyms.append(l.antonyms()[0].name())
print(set(synonyms))
print(set(antonyms))
```

Results as follows
{'leave', 'admit', 'grant', 'reserve', 'allow_for', 'set_aside', 'give_up', 'appropriate', 'permit', 'allow', 'earmark', 'let', 'tolerate', 'provide', 'take_into_account', 'countenance'}
{'forbid', 'deny', 'prevent', 'disallow'}

With def text_parser_synonym_antonym_finder() Function we will expand a query in the form of sentence using wordnet.

```
antonyms = []
for syn_set in wordnet.synsets("hesitate"):
 for l in syn_set.lemmas():
  synonyms.append(l.name())
  if l.antonyms():
   antonyms.append(l.antonyms()[0].name())
```

```
print(set(synonyms))
print(set(antonyms))

def text_parser_synonym_antonym_finder(text):

    tokens = word_tokenize(text)
    synonyms = defaultdict(list)
    antonyms = defaultdict(list)
    for token in tokens:
        for syn in wordnet.synsets(token):
            for i in syn.lemmas():
                # synonyms.append(i.name())
                 #print(f'{token} synonyms are: {i.name()}')
                 synonyms[token].append(i.name())
                 if i.antonyms():
                    # antonyms.append(i.antonyms()[0].name())
                     #print(f'{token} antonyms are: {i.antonyms()[0].name()}')
                     antonyms[token].append(i.antonyms()[0].name())
    pprint.pprint(dict(synonyms))
    pprint.pprint(dict(synonyms))
    synonym_output = pprint.pformat((dict(synonyms)))
    antonyms_output = pprint.pformat((dict(antonyms)))
    with open(str(text[:5]) + ".txt", "a") as f:
        f.write("Starting of Synonyms of the Words from the Sentences: " + synonym_output + "\n")
        f.write("Starting of Antonyms of the Words from the Sentences: " + antonyms_output + "\n")
        f.close()
```

Pass any string to function you wish to expand. For the query **"Slow Cable Internet Problem"** the result as follows with multiple expansion words.

{'cable': ['cable',

'cablegram',

'overseas_telegram',

'cable',

'line',

'transmission_line',

'cable',

'cable',

'cable_length',

''cable's_length'',

'cable_television',

'cable',

'cable',

'cable_television',

'cable_system',

'cable_television_service',

'cable',

'telegraph',

'wire',

'cable'],

'internet': ['internet', 'net', 'cyberspace'],

'problem': ['problem', 'job', 'problem', 'trouble', 'problem'],

'slow': ['decelerate',

'slow',

'slow_down',

'slow_up',

'retard',

'slow',

'slow_down',

'slow_up',

'slack',

'slacken',

'slow',

'slow_down',

'slow_up',

'slow',

'slow',

'dense',

'dim',

'dull',

'dumb',

'obtuse',

'slow',

'slow',

'boring',

'deadening',

'dull',

'ho-hum',

'irksome',

'slow',

'tedious',

'tiresome',

'wearisome',

'dull',

'slow',

'sluggish',

'slowly',

'slow',

'easy',

'tardily',

'behind',

'slow']}

{'slow': ['accelerate', 'fast', 'fast', 'fast', 'quickly']}