Semantic Text Indexing in PostgreSQL

Problem Statement: Searching documents containing words in a query as well as words which are similar to query words.

Dataset: We use a dataset of 5736 documents from the medical domain.

Approaches: We implemented 6 approaches in which we create synonym dictionaries as follows:

1. We use synsets from Wordnet to populate the synonym dictionary. (completely integrated with postgresql)
2. We use GloVe to learn word embeddings from our dataset. We hypothesize that similar words should have the characteristics of synonyms. We use similar words to populate the synonym dictionary. (completely integrated with postgresql)
3. We use FastText to learn word embeddings from our dataset. We populate the synonym dictionary in a similar manner. (completely integrated with postgresql)
4. We use k-means clustering on word vectors learned using FastText. We hypothesize that words belonging to the same cluster have the characteristics of synonyms. (completely integrated with postgresql)
5. We use Locality Sensitive Hashing (LSH) for similarity-based retrieval. (implemented separately, integration with postgresql is remaining)
6. We also use LDA which is a topic modeling method for retrieving similar documents. (partially implemented)

We observe that k-means clustering, when used on word vectors, learned using FastText perform the best.

Future Work:

- We plan on using PL/Python to integrate LSH based approach with PostgreSQL.
- We plan on completing our LDA based approach.
- Currently, we compared our approaches based on qualitative evaluation. We plan on evaluating our approaches using precision, recall, and f1-score.

Contribution: We did pair programming for this project and contributed equally.

Github Project Link:
https://github.com/dracarys09/Semantic_Text_Indexing_With_PostgreSQL

Steps to run the project:

1. Create table documents:
   a. `create table documents (title varchar(20), description TEXT );`
2. Insert data into table:
   a. `python data/create_large_dataset.py documents.txt`
3. For similarity methods: Preprocess data and create vocabulary with [method:wordnet/glove/fasttext]:
   a. `python create_vocab.py [method]`
4. For similarity methods: Create dictionary with [method:wordnet/glove/fasttext]:
   a. `python create_dictionary.py [method]`
5. For k-means-clustering method:
   a. `python kmeans_create_vocab.py`
6. synonym dictionary is formed in [name].syn format: e.g. *synonym_sample_[method].syn.*
   Location of this file: /home/user/postgresql/src/install/share/tsearch_data This file is used by pgadm3 in the following steps.
7. From pgadm3 interface, create text search dictionary as follows:
   a. `CREATE TEXT SEARCH DICTIONARY document_dict ( template = synonym, synonyms = synonym_sample );`
   b. To delete dictionary for recreating it for various methods: `DROP TEXT SEARCH DICTIONARY document_dict CASCADE;`
8. From pgadm3 interface,
   a. `CREATE TEXT SEARCH CONFIGURATION document_config (copy=simple);`
9. Map text search configuration to the text search dictionary:
   a. `ALTER TEXT SEARCH CONFIGURATION document_config ALTER MAPPING FOR asciiword, asciihword, hword_asciipart, word, hword, hword_part WITH document_dict;`
10. Try searching with a query word:

a. `SELECT title from documents where to_tsvector('document_config_ft', description) @@ to_tsquery('document_config_ft','anthropologists');`
    - This should fetch titles of documents whose description column contains the query word or any other word which is similar to query word as per the mapping in the dictionary configuration currently used (document_config).

Example Results:
We provide an example for k-means clustering method**:**

Query:
SELECT title from documents where to_tsvector('document_config_ft', description) @@ to_tsquery('document_config_ft','actor');

Results:
Time(Without Indexing): 2.2 secs
Time(With Indexing - GIN index): 1.4 secs
11 rows retrieved.

Rows retrieved:
"Document133"
"Document761"
"Document1503"
"Document1890"
"Document2181"
"Document3797"
"Document3801"
"Document3802"
"Document4098"
"Document5199"
"Document5222"

Rows from syn dictionary corresponding to 'actor':
actor actor
IGH actor
RDT actor

Explanation: Documents fetched contain either or more of the {actor, IGH, RDT} words.

References

1. https://data.mendeley.com/datasets/9rw3vkcfy4/6
2. https://towardsdatascience.com/understanding-locality-sensitive-hashing-49f6d1f6134
3. https://github.com/facebookresearch/fastText
4. https://pypi.org/project/gensim/
5. https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24
6. https://www.postgresql.org/