Sangita Gupta

Project 4: Semantic Search

Objective:

The objective of this assignment is to engineer a novel wikipedia search engine using what you've learned about data collection, infrastructure, and natural language processing.

The task has two required sections:

1. Data collection
2. Search algorithm development

Part 1 - Data Collection:

I query the wikipedia API and try to collect all of the articles under the following wikipedia categories and their subcategories:

- Machine Learning
- Business Software

Both these categories contain a hierarchy of nested sub-categories.
For ***Machine Learning***, I was able to pull all the pages, which total 1620 pages.
For ***Business Software***, the recursive process for acquiring all pages of all subcategories, seemed to spin off into an infinite loop. To avoid this, I chose to limit the recursive subcategory search to a depth of two levels. I pulled 4075 pages total for Business Software.

Storage:

The raw page text, its category and page id, were stored in a dictionary. Each page was represented by a dictionary and each category was represented by a list of dictionaries. The information was then written to a 'raw data' collection on a Mongo server, running on a dedicated AWS instance. I retrieved this 'raw data' collection, cleaned its text to remove html and newline tags, and stored it in a 'clean data' collection on the Mongo server.

Reference: jupyter notebook "data_acquisition_NB1"

```
# fetch wikipedia's 'category: Business software' page content.
# (do this for pages in its categories and subcategories).
category = "Category:Business_software"
entire_category_data_list = wiki.get_wiki_full_category_content(category, tree_depth=2)
```

```
100%|██████████| 4075/4075 [18:17<00:00,  3.71it/s]
```

```
# create a collection in the Mongo database for the BS content, retrieved from wikipedia.
mongo.mongoDB_create_collection('wiki_database', 'wiki_BS_collection', entire_category_data
```

```
100%|██████████| 4075/4075 [02:45<00:00, 24.56it/s]
```

```
# clean the retrieved text.
BS_clean_text_content_list = gu.clean_text(entire_category_data_list)
```

```
100%|██████████| 4075/4075 [00:00<00:00, 21952.04it/s]
```

```
# create a collection in the Mongo database for the cleaned BS content.
mongo.mongoDB_create_collection('wiki_database', 'wiki_BS_clean_collection', BS_clean_text_c
```

```
100%|██████████| 4075/4075 [02:43<00:00, 24.99it/s]
```

```
# get collection names on specified mongo db.
mongoDB_get_collection_names('wiki_database')
```

```
['wiki_BS_collection',
 'wiki_ML_collection',
 'wiki_ML_clean_collection',
 'wiki_BS_clean_collection']
```

Part 2 – Search:

Developed a search engine, using Latent Semantic Analysis, to match a “user query” with its most similar wiki articles. I developed an interactive engine, which takes a query from the user and returns a list with its top 5 wiki article hyperlinks.

I used singular vector decomposition (SVD) and cosine similarity, to find the most similar wiki articles.

Reference: jupyter notebooks “semantic_analysis_NB2” and “final_query_app_NB3”

```
Type wiki query(q to exit): stochastic time-series forecasting

Here are the top 5 wiki page links for your query:

Doubly stochastic model

Stochastic neural network

Entropy rate

Stochastic cellular automaton

Stochastic matrix


Enter wiki query(q to exit): machine learning decision tree

Here are the top 5 wiki page links for your query:

Decision tree

Incremental decision tree

Grafting (decision trees)

Decision tree learning

Outline of machine learning


Enter wiki query(q to exit): business software
```

Optional: I made the query app also run via a python script.

```
Type wiki query(q to exit): stochastic time-series forecasting

***** Here are the top 5 wiki page links for your query: *****

'https://www.wikipedia.org/wiki/Doubly_stochastic_model'
'https://www.wikipedia.org/wiki/Stochastic_neural_network'
'https://www.wikipedia.org/wiki/Entropy_rate'
'https://www.wikipedia.org/wiki/Stochastic_cellular_automaton'
'https://www.wikipedia.org/wiki/Stochastic_matrix'

Enter wiki query(q to exit): Random Forest boosting

***** Here are the top 5 wiki page links for your query: *****

'https://www.wikipedia.org/wiki/Random_indexing'
'https://www.wikipedia.org/wiki/Boosting_(machine_learning)'
'https://www.wikipedia.org/wiki/Random_subspace_method'
'https://www.wikipedia.org/wiki/Random_projection'
'https://www.wikipedia.org/wiki/Clustering_illusion'

Enter wiki query(q to exit): q
```