Bingbing

Assignment 1 - Relevance Feedback

Team Members

- Nikhil Mitra (nm2868)
- Prakhar Srivastav (ps2894)

File List

```
|-- README.md
|-- transcripts
|   |-- musk-output.txt
|   |-- gates-output.txt
|   |-- taj-mahal-output.txt
|-- requirements.txt
|-- run.sh
|-- src
|   |-- __init__.py
|   |-- corpora.py
|   |-- config.py
|   |-- bing.py
|    -- starter.py
|-- tests
    |-- __init__.py
    |-- corpora_test.py
```

Running the program

On CLIC

To run on the *CLIC* machines, just run the following set of commands.

```
$ cd /home/ps2894/bingbling
$ ./run.sh
```

Build it locally

In order to build the project on your computer, just install the python packages the program depends on. A list of these packges is provided in the the `requirements.txt` file. Since this program uses `nltk`, nltk data also needs to be downloaded.

```
$ pip install -r requirements.txt
$ python -m nltk.downloader all
$ ./run.sh
```

Bing Account Key

The bing account key is `Byygq1zI2KKyssKp8UvVe3DV/v6Aa0FEsKrE+pqDa0s`

Internal Design

The program is divided primarily into four files.

- `config.py`: Contains key-value pairs for configuring the program. This includes the account keys, the output format and configuration parameters for the underlying algorithm.
- `bing.py`: Contains the implementation specific to the bing search engine which is responsible for making the API call, parsing and organizing the results, and then passing those results off to the main program. This has been separated so that another search engine can be added via a drop-in replacement.
- `starter.py`: This file is the main driver program. Its job is to get the results from the search engine module (bing in this case), make changes to the results (such as adding the query in the dataset) and call the public methods exposed by the `corpora` module. `starter.py` takes care of all the user-interaction logic such as taking input from user, calculating the precision, changing the query and lastly, deciding when to stop.
- `corpora.py`: Corpora.py is the primary workhorse of the program. It implements the **Rocchio's Algorithm** for modifying the query. It relies on external libraries such as NLTK to help with stopwords filtering and word tokenization. As a part of Rochhio's algorithm, the code in this file does the following -

  1. Creates a vocabulary
  2. Uses the vocabulary to compute tf-idf for each word
  3. Creates the document vector for each vector
  4. Applies the Rocchio's query update formula
  5. Returns a set of (word, score) tuples sorted in the descending order of scores

Program Flow

hello world

Query Modification

Folder Structure

The folder *outputs* contains sample outputs of the algorithm. The folder *src* contains the main source code.