Data mining default project report

Albu Alexandru

General remarks

The python script will heavily rely on two libraries, namely whoosh and nltk.

In order to solve this problem, I have split this into two sub tasks. The first task involves the index creation:

1. Create an index using whoosh. Store the title of the Wikipedia page and the content
2. Do some filtering on the page before storing it in the index
3. Save this index for later use as it takes a while to create it

The second task should take care of trying to answer the questions found in questions.txt:

1. Load up the index if it exists in the local filesystem
2. Read the questions
3. Filter the questions
4. Make an OR query with the category and the entire clue
5. Check if there are any results

Filtering

The content of the Wikipedia pages requires some filtration to have a more reliable retrieval system. Firstly, I used the nltk library to tokenize the content into individual words. Afterwards, using the same library I applied porter stemming to reduce the words to their base form. Finally, I've excluded stop words from the index with the help of the same library.

Another point worth mentioning is the fact that the given dataset contains multiple Wikipedia pages inside a single text file. In order to separate each individual page with its' own content, I have used a regex so that my index contains a title and content for an individual Wikipedia page, not for the entire text file.

Answering a query

After creating the index with the entire dataset, it is time to query it using a combination of the question category and clue. I have decided to OR the clue and the category so that I can get as many hits as possible even though the results might not be the best. Using the whoosh QueryParser I interrogated the index for every clue-category query. If the first retrieved result matches the answer from the questions.txt file, we have a success.

Performance

In order to measure the performance of the system, I decided to use the precision at 1 metric. I believe this is a good metric due to the nature of the problem. We need a single answer for a question and ideally this answer should be the first retrieval, the top result.

To have this metric implemented, I simply increased a counter whenever I got a match for the top ranked result (the Wikipedia page title matched the answer). When all the questions were answered, I just divided the counter with the total number of questions obtaining the metric.

Initial results

Running the program yielded an initial result of 20 correctly answered queries out of 100:

```
Trying to answer question number 100...
Correct answers so far: 20
Got 20 right answers out of 100 questions
P@1: 0.2

Process finished with exit code 0
```

Error analysis

The causes for not finding the right answer to a question are multiple.

The program might not correctly recognize synonyms or idioms. For example a clue might contain the syntagm “once in a blue moon” and it will not get matches for the term “rarely”.

The program cannot process context. For example, we have the following category and clue:

Category: COMPLETE DOM-INATION(Alex: Not "domination.")

Clue: This New Orleans venue reopened Sept. 25, 2006

Without context, it is quite hard to grasp that this is about The Superdome. Also, the pun-like category might induce additional errors in the program.

Improving the output

In order to check if natural language processing can yield better results, I slightly altered my current algorithm so that not only the best ranked result can be a match, but the entire result set. This will showcase that in theory, an improvement can be made to get a better top ranked result.

```
Trying to answer question number 100...
Correct answers so far: 37
Got 37 right answers out of 100 questions
P@1: 0.37

Process finished with exit code 0
```

This result means that with the current index, we have the answers for 37 questions. The improvement part is about making sure that those results will be the first to pop up. One solution would be to ask chat-gpt which result he would pick for a given category and clue out of the result list.