Documentation for Automatic Video Annotation

Authors:
Amod Agrawal (amodka2)
Mariam Vardishvili (mariamv2)

The main code for the project is divided into three files:
- parser.py
- inverted_index.py
- ranking.py

Software required to run code:
- Python 2.7
- NLTK 3.3 (Natural Language Tookit)
- NLTK English stop-words
- NumPy

---

parse.py

This code is primarily responsible for parsing the subtitles file from the Coursera website and making a time-based index.

Input for this code: *subtitles.vtt* file
Output for this code: *time_index.csv*

This code reads the *subtitles.vtt* file and ignored the lines that are not relevant to creating an index. It creates an index of the following mapping:

doc_id, start_time (timestamp), end_time (timestamp), subtitle_text

The timestamps provided from the file are of form H:M:S.ms.

Once this mapping is created, the code performs preprocessing on the subtitle_text.

Function: List<String> preprocess(Sting)

We use NLTK and its methods for preprocessing.

This function:
- Removes stop words
- Removes all special characters that are part of the text, like: ***“ , # @ [ ]***
- Uses a RegEx based Tokenizer to break sentences to words
- Uses Porter stemmer to stem the words

---

inverted_index.py

This code is primarily responsible for developing an inverted index from the time_index file.

Input for this code: *time_index.csv*
Output for this code: *inverted_index.csv*

We develop a different kind of index because we want to store temporal information as well. Hence, we develop this from scratch instead of using libraries.

Function: List<String> preprocess(Sting)

We use NLTK and its methods for preprocessing.

This function:
- Removes stop words
- Removes all special characters that are part of the text, like: ***“ , # @ [ ]***
- Uses a RegEx based Tokenizer to break sentences to words
- Uses Porter stemmer to stem the words

We use a python dictionary to keep track of words in the index. The inverted index is of form:

{ “word” : [[doc_ID, freq], [doc_ID, freq]] }

Functions *make_inverted_index* and *add_to_inverted_index* are helper functions to construct the index.

---

ranking.py / qa.py

This code is primarily implementation of our algorithm for Video Annotation. It uses the inverted_index along with time_index to return the timestamps for queries.

Function of this program:

- Takes query as user input
- Performs preprocessing as described before on the query to create a vector
- Loads inverted_index and time_index in memory (done to avoid being slowed down by very large inverted indices - if we have to load a part of index)
- Calculate tf-idf and bm_25 scores
- Do temporal analysis to get the best document
- It first finds documents which are most relevant, and then does temporal analysis for those documents to find the most relevant timestamp (explained in the video).
- Present result to the users with mapping data from time_index, slide_index and its relevance information.

Function: float bm_25 (String word, List <String> doc)

Implements the bm25 ranking algorithm, also returns tf-idf score if needed.

Instructions: Running the Tool

Running our program is fairly straightforward. If all dependencies are installed:

Run on Terminal:

> python **qa.py**
> Enter the Coursera Lesson you want to search: *2.4*
> Enter your query: *inverted index using faster search*

> Results:

doc_ID, timestamp, slide_title, relevance_value

Team Member Contributions

Amod and Mariam worked together on this project and divided work hours equally. Most of the time, we worked together to write the code, analyze different algorithms, read literature and discuss ways to improve it.