Authors: Urvi Awasthi and Anupam Ojha

CS 410 Final Project Documentation

1) An overview of the function of the code (i.e., what it does and what it can be used for).

We have created an enhanced version of SmartMoocs, an intelligent learning platform with lecture videos for CS 410: Text Information Systems. The existing platform for Smartmoocs segmented lectures by dividing up lectures by the minute. For our project, we explored ways to segment the lecture videos based on topic transitions detected by the program. In turn, this would allow the user of the learning platform to directly jump to the topic at hand in the video, at the time mark specified, without coursing through the rest of the lecture.

Our method for topic segmentation is modeled from the research paper titled "Topic Based Segmentation of Classroom Videos", in which a method of merging segments based on similarity between transition segments is discussed in section IV, Text Based Indexing. We implemented the text-based indexing algorithm discussed in this paper, using cosine similarity as our similarity function. Then, after segmenting the lectures, we used LDA to generate a bag of words representing each segment. The only pitfall in this topic based indexing algorithm was that the user is supposed to supply the number of topics covered in the lecture as an input for the program to segment lectures accordingly. (Also, it might be worthwhile to look into using KD-trees as a data structure in the future, this was a very simplified version of the algorithm, and our code used a dictionary instead).

2) Documentation of how the software is implemented with sufficient detail so that others can have a basic understanding of your code for future extension or any further improvement.

There are three main parts that cover a basic implementation of the code for our project.

I. **CS410-UI folder.** This folder consists of all of the website / platform code for the SmartMoocs 2.0 website that we have made, which is modeled off of the current SmartMoocs website.
   A. **index.html.** The homepage of our website.
   B. **Text-Retrieval-Lectures.html.** Accordion view of all of the text retrieval lectures for the class.
   C. **Text-Analytics-Lectures.html.** Accordion view of all of the text analytics lectures for the class.
   D. **Segment-Lecture-Video.html.** View selected video and its topic segmentation with LDA.
   E. **test.py.** Python web server. Handles logic for calling the LDA / Topic Segmentation functions and sending the response to the front end.

II. **Cosine_Similarity.py.** Given a lecture transcript and a number of topics, segments the lecture based on topic transitions.
   A. **build_vtt_dict(lecture_transcript)**. Given a vtt file of the lecture transcript, creates a dictionary- {start_time: [sentence, end_time, duration]}.
   B. **get_transition_points(sentence_timestamp)**. Given the dictionary with the sentence / timestamp information, finds all sentences that begin with a transition.
   C. **combine_segments(start_time1, start_time2, sentence_timestamp)**. Merges two lecture segments together and returns the updated dictionary.
   D. **get_next_segment(start_time, sentence_timestamp) / get_previous_segment(start_time, sentence_timestamp)**. Gets the next consecutive segment in the dictionary, according to time elapsed.
   E. **get_smallest_segment(sentence_timestamp)**. Gets the smallest segment in the dictionary in terms of duration.
   F. **cosine_similarity(start_time1, start_time2, sentence_timestamp)**. Gets the similarity between two segments based on the cosine similarity model.
   G. **find_lecture_segments(transcript, num_topics)**. Puts everything together. Pseudocode from research paper:

http://icsvideos.cs.uh.edu/intro/Topic%20Based%20Segmentation%20of%20Classroom%20Videos.pdf

Data: A list of transition points ;
Required number of index points (N);
Grouping duration in seconds;
Result: N index points that are a subset of given transition points;
repeat
  Select transition segment with smallest duration;
  if *the similarity is more towards right group* **then**
    merge right;
  else
    merge left
  end
until *Number of transition points == Required number of index points*;

Algorithm 1: Text-based indexing algorithm

III. **LDA.py.** Latent Dirichlet Allocation for topic extraction. Used to develop a bag of words model for each outputted lecture segment.
   A. **lemmatize_stemming(text) / preprocess(text)**. Removing stopwords, performing tokenization, lemmatization and stemming words.
   B. **run_LDA_on_transcript(transcript).** Runs LDA on given lecture segment.

3) Documentation of the usage of the software including either documentation of usages of APIs or detailed instructions on how to install and run a software, whichever is applicable.

1. Clone the project into an empty repository.
   a. git clone https://github.com/urviawasthi/CourseProject.git new-repo
2. Change directories into the CS410-UI directory.
3. Run ./test.py. Open the server on the link that shows up. If asked to install any libraries, please do so.
4. Enjoy the enhanced SmartMoocs website.

4) Brief description of contribution of each team member in case of a multi-person team.

- Urvi Awasthi (urvia2): researched methods for topic segmentation, programmed LDA, cosine similarity, initialized web server and all webpages.
- Anupam Ojha (anupamo2): researched methods for topic segmentation, updated UI to show segmented time stamps, helped debug features, tested platform.