Web Science 2019: Quiz Master

This project description is likely to be adapted throughout the course.

This project counts for 40% of the final grade. The project will be graded as a whole. The aim is to create a quiz master that asks trivia questions, registers the user answers, and adapts its computations accordingly. Details are given below.

1 Week 1 (5 February)

The purpose of the first week is to familiarize yourself with the programming basics needed for the project.

You will be given a dataset of 8,913 question-answer pairs, each labelled according to 5 different topics (science-technology, for-kids, video-games, sports, and music). Please read the README file, so you understand the format. The dataset will be made available on Absalon.

- Load the dataset and compute the topic-wise statistics of the number of questions and mean/standard deviation of question and answer length
- Train a classifier to predict the topic of a question. (In Weeks 2-3 you will re-use this part to classify the topic of unseen questions).

  We suggest using a bag-of-words representation of each document and a Random Forest as the classification model. Perform a stratified split of the provided data into 80% for training, 10% for validation, and 10% for testing.
- Train a neural classification model (e.g., using the keras library) to also classify the topic of each question. If you do not have the necessary computer resources yourself, you can familiarize yourself with the Google Colab platform[1]. Google Colab provides free GPU resources for small-scale experiments (up to 12 hours at a time).

  We suggest using trained word embeddings[2] as word representations, a recurrent neural network with LSTM cells as the model, and cross-entropy as the loss function. Use the Adam optimizer for training the model. Split the provided data in a stratified fashion into 80% for training, 10% for validation, and 10% for testing.

[1]`https://colab.research.google.com`

[2]e.g. word2vec skip-gram, `https://code.google.com/archive/p/word2vec/`

- The above is a description of what we expect you to implement, but you are allowed to expand upon this to explore more complex models.

We recommend the use of Python Version 3 with the SciPy stack[3], scikit-learn, and keras. These are only recommendations, and you are allowed to use any programming language or libraries you want.

2 Week 2-3 (12 & 19 February)

The purpose of the second and third week is to (a) create a new dataset using crowdsourcing, and (b) classify the topic of the unseen crowdsourced questions. This is explained next.

2.1 Create a new dataset using crowdsourcing

Each student must come up with 1 question and answer for each of the same five topics introduced above (science-technology, for-kids, video-games, sports, and music) and for each difficulty level (easy, medium, hard). These 15 questions-answers must be factual, not subjective. Example of a factual question from the video-games topic:

> Q: What is the name of the video-game series featuring Mario, Luigi, and other characters racing each other in Go-karts and other vehicles?
>
> A: Mario Kart

Example of a subjective question from the same topic:

> Q: What is my favorite game?
>
> A: (Many possible answers)

You must submit your 15 questions-answers on the following Google form[4]. After you have submitted your 15 questions-answers, you will be given 45 unseen questions. To do this, we will generate a spreadsheet with all the questions collected during the crowdsourcing process, and assign 45 questions for each student. For each of these questions, you must do the following:

1. Answer the question
2. Label the question difficulty as easy, medium, or hard
3. Label how much you like the question on a scale from 1-3 indicating whether you do not like the question (1), have no preference (2), or like the question (3).
4. Give a binary score to indicate whether it is a factual question (1) or not (0).

[3] `https://www.scipy.org/stackspec.html`
[4] `https://goo.gl/forms/AWZ6Iic1X4aO9vd13`

2.2 Clean the new crowdsourced dataset

All the data provided by all students as described above will be collected. This will be our crowdsourced dataset. You will be given this crowdsourced dataset (without the topic labels) and must aggregate the scores given by many students to the same question and also clean the dataset.

- To aggregate the scores of many users: Use majority voting to estimate the final question difficulty score for each question. In case of ties, compute the average.
- To clean the dataset: Use majority voting to estimate the final factuality label for each question and discard those questions that are judged as non-factual (i.e. subjective).

You will use this version of your dataset in weeks 4-8.

2.3 Mini Competition (Start: 22 February. End: 1 March)

You must classify the topic of each question on the crowdsourced dataset. Specifically, we will provide a CSV file with all the crowdsourced questions and require you to generate a corresponding CSV file with the topic classification for each question in the same order as the provided question file. You must use your topic model (e.g. the one that you developed in week 1) for this task. We will host a mini competition with a public leaderboard to encourage the development of high-quality topic models. You must upload your CSV file to `http://main.cl-lab.dk/www/webscience2019-competition` where the public leaderboard will also be hosted.

3 Week 4-5 (26 February & 5 March)

The purpose of the fourth and fifth week is to get some experience in developing a recommendation system. You will use the user data in the crowdsourced dataset to recommend friends with similar preferences. Thus, you need to build profiles for users and recommend users with similar profiles. You must implement the following two implementation requirements: the first one treats topics as features to build the profile, and the second considers words in question text (documents) as features to build the profile. You are required to use the Pearson correlation to compute the similarity between users for the first one, and cosine similarity for the second one.

Recommend friends according to topic feature

- For each user, calculate the average preference score for each topic with the preference score of each question concerning that topic, then you get 5 scores for the 5 topics, as a vector of features for the user.
- Use Pearson correlation to calculate the similarity between every two users with the vectors of 5 features, and plot an N*N matrix, where there are N users.
- Choose one user ID, and rank that row of the similar users in a descending way and observe the top-K friends.

Recommend friends according to features in question text

- For each question text, compute the TF-IDF score for each word and use top n words as vectors so that each question is represented as a vector of features (words). 'n' is the parameter that you can adapt.
- For each user, collect those questions that the user has rated as "like the question," and compute the average of their vectors as the profile vector.
- Use cosine similarity to calculate the similarity between every two users and plot an N*N matrix, where there are N users
- Choose one user ID, and rank that row of the similar users in a descending way and observe the top-K friends.

4 Week 6-7 (12 & 19 March)

You will now develop your quiz master. Your quiz master must include all of the following six components: Classification, Convergence, Users Preference, Friend Recommender, Difficulty, and Simulator. The quiz master interface will not be graded, meaning that using the terminal as the interface is enough.

4.1 Classification Component

The quiz master should ask the user to choose one of the five topics (science-technology, for-kids, video-games, sports, and music) and one of the three levels of difficulty (easy, medium, hard). The quiz master should then output a question of the correct topic and level of difficulty. To do this, the quiz master must:

- Classify each question by its topic (using the classification model trained in weeks 2-3).
- Automatically distinguish the questions accordingly to their difficulty level (by taking the majority vote among the difficulty assessments, as done in weeks 2 and 3).
- Allow the user to input the desired topic and difficulty level using the terminal input.
- Provide questions about the selected topic and difficulty level to the user.

4.2 Convergence Component

The aim of this component is to predict the user's knowledge based on his/her answers. The quiz master should identify the topic that the user knows the most about and then converge its questions to this topic. To do this, the quiz master must:

- Start by asking questions about different topics.
- Collect the user's answer these questions.

- From these questions and answers, identify topics that the user knows the most about. A naive approach would be to check the user answer correctness variation over each topic and discard the topics that the user answered incorrectly.
- Repeat this process until the topic of the questions displayed to the user matches with the topic that the users know the most.

4.3 Friend Recommender Component

The quiz master should recommend friends, using the methods developed in weeks 4 and 5. To do this, the quiz master must:

- Allow the user to select which friend recommendation strategy the user would like to use (topic features or question text features).
- Ask the user to input his User ID and cutoff (k).
- After collecting the user ID and cutoff, the quiz master should recommend a list of k friends that are more similar to that used by using the strategy previously selected.

4.4 Users Preference Component

The quiz master should provide questions that match the user's preference. Remember that, when crowdsourced answers, all students were asked to provide their preference indicating whether they like or not a question. Now, your quiz master should use that information to identify the user's preference and provide questions about the topic that the user likes the most. To do this, the quiz master must:

- Ask the user to input his/her user id.
- Identify the user's preference by calculating the user average preference score of the questions with regards to a specific topic for each one of the five topics. For instance, consider a user that gave preference scores (3, 3, 2, 2, 3) for 5 questions of the topic Movie. The user preference for the topic Movie would be 2.6, which is the average of the 5 questions.
- After identifying the topic that the user likes the most, the quiz master should provide questions about the topic that matches the best with the user's preference.

4.5 Difficulty Component

The quiz master should start by providing easy questions to the user, and increase the question difficulty as the user answers a question right. The difficulty of a question should be estimated by using majority vote among the multiple difficulty assessments collected in weeks 2 and 3. To do this, the quiz master must:

- Ask the user a question
- Check whether the user's answer is correct or not.

- If correct, move on to the next level of difficulty, otherwise, ask another question of the same level of difficulty. This can be done by taking the majority vote over the multiple difficult assessments provided in week 2 and 3.
- Repeat the process until users complete the three levels of difficulty (easy, medium, hard).

4.6 Simulator Component

The aim of this component is to evaluate the other components automatically. The main idea consists of generating multiple simulations of how users interact with the quiz master in order to systematically evaluate it. We here provide a list of user simulations that you must create for your simulator component:

- Polymath user: simulating a user whose expertise spans a significant number of different topic areas. This simulation must contain a correctness parameter setting (Default: 90%).
- Topic Expert user: simulates a user that knows everything about one specific topic. This profile should also include a correctness parameter (Default: 90%).

The above list is not an exhaustive list, meaning that you can create other users simulations to test your quiz master. Note that this component is essential to test the Convergence component as it will simulate the convergence of the quiz Master for different users.

5 Week 8 (26 March)

You will orally present your quiz master (including a live demo) and will have to answer questions about the implementation and methods from the audience. The audience will be the class and the teachers. The quiz master must contain all the components described above. In addition, you must submit a Jupyter notebook showcasing your quiz master and its components via the Simulator Component. The presentation and Jupyter notebook counts for 40% of your final grade. The project (including the presentation and Jupyter notebook) will be graded as a whole.

6 Academic Code of Conduct

You are welcome to discuss the project with other students, but sharing of code is not permitted. Copying code directly from other students will be treated as plagiarism. Please refer to the University's plagiarism regulations if in doubt. For questions regarding the project, please ask on the Absalon discussion forum.