COMP9727: Recommender Systems
Assignment: Content-Based Music Recommendation

Due Date: Week 4, Friday, June 27, 5:00 p.m.

Value: 30%

This assignment is inspired by a typical application of recommender systems. The task is to build a content-based "music recommender" such as might be used by a streaming service (such as Spotify) to give users a personalized playlist of songs that match their interests. The main learning objective for the assignment is to give a concrete example of the issues that must be faced when building and evaluating a recommender system in a realistic context. **It is not the purpose of this assignment to produce a very good music recommender system.** Note that, while music recommender systems commonly make use of the ratings or listening history of users with similar tastes, our scenario is not unrealistic as sometimes a music recommender system can make recommendations using features of the songs liked by the users.

For this assignment, you will be given a collection of 1500 songs that have been labelled as one of 5 main topics: *dark*, *emotion*, *lifestyle*, *personal* and *sadness*. The songs are in a single `.tsv` file with 6 fields: *artist_name*, *track_name*, *release_date*, *genre*, *lyrics* and *topic*.

The assignment is in three parts, corresponding to the components of a content-based recommender system. The focus throughout is on *explanation* of choices and *evaluation* of the various methods and models, which involves choosing and justifying appropriate metrics. The whole assignment will be prepared (and submitted) as a Jupyter notebook, similar to those being used in tutorials, that contains a mixture of running code and tutorial-style explanation.

Part 1 of the assignment is to examine various supervised machine learning methods using a variety of features and settings to determine what methods work best for topic classification in this domain/dataset. For this purpose, simply concatenate all the information for one song into a single "document". You will use Bernoulli Naive Bayes from the tutorial, Multinomial Naive Bayes from the lecture, and one other machine learning method of your choice from scikit-learn or another machine learning library, and NLTK for auxiliary functions if needed.

Part 2 of the assignment is to test a potential recommender system that uses the method for topic classification chosen in Part 1 by "simulating" a recommender system with a variety of hypothetical users. This involves evaluating a number of techniques for "matching" user profiles with songs using the similarity measures mentioned in the lecture. As we do not have real users, for this part of the assignment, we will simply "invent" some (hopefully typical) users and evaluate how well the recommender system would work for them, using appropriate metrics. Again you will need to justify the choice of these metrics and explain how you arrived at your conclusions.

Part 3 of the assignment is to run a very small "user study" which means here finding *one* person, preferably not someone in the class, to try out your recommendation method and give some informal comments on the performance of your system from the user point of view. This does not require any user interface to be built, the user can simply be shown the output (or use) the Jupyter notebook from Parts 1 and 2. However, you will have to decide how many songs to show the user at any one time, and how to get feedback from them on which songs they would click on and which songs match their interests. A simple "talk aloud" protocol is a good idea here (this is where you ask the user to use your system and say out loud what they are thinking/doing at the same time – however please do not record the user's voice – for that we need ethics approval).

Note that standard UNSW late penalties apply.

Assignment

Below are a series of questions to guide you through this assignment. Your answer to each question should be in a separate clearly labelled section of the Jupyter notebook you submit. Each answer should contain a mixture of explanation and code. Use comments in the code to explain any code that you think readers will find unclear. The "readers" here are students similar to yourselves who know something about machine learning and text classification but who may not be familiar with the details of the methods.

Part 1. Topic Classification

1. (2 marks) There are a few simplifications in the Jupyter notebook in the tutorial: (i) the regex might remove too many special characters, and (ii) the evaluation is based on only one training-test split rather than using cross-validation. Explain how you are going to fix these mistakes and then highlight any changes to the code in the answers to the next questions.

2. (2 marks) Develop a Multinomial Naive Bayes (MNB) model similar to the Bernoulli Naive Bayes (BNB) model. Now consider all the steps in text preprocessing used prior to classification with both BNB and MNB. The aim here is to find preprocessing steps that maximize overall accuracy (under the default settings of the classifiers and using `CountVectorizer` with the standard settings). Consider the special characters to be removed (and how and when they are removed), the definition of a "word", the stopword list (from either NLTK or scikit-learn), lowercasing and stemming/lemmatization. Summarize the preprocessing steps that you think work "best" overall and do not change this for the rest of the assignment.

3. (2 marks) Compare BNB and MNB models by evaluating them using the full dataset with cross-validation. Choose appropriate metrics from those in the lecture that focus on the overall accuracy of classification (i.e. not top-N metrics). Briefly discuss the tradeoffs between the various metrics and then justify your choice of the main metrics for evaluation, taking into account whether this dataset is balanced or imbalanced. On this basis, conclude whether either of BNB or MNB is superior. Justify this conclusion with plots/tables.

4. (2 marks) Consider varying the number of features (words) used by BNB and MNB in the classification, using the `sklearn` setting which limits the number to the top N most frequent words in the Vectorizer. Compare classification results for various values for N and justify, based on experimental results, one value for N that works well overall and use this value for the rest of the assignment. Show plots or tables that support your decision. The emphasis is on clear presentation of the results so do not print out large tables or too many tables that are difficult to understand.

5. (5 marks) Choose one other machine learning method, perhaps one mentioned in the lecture. Summarize this method in a single tutorial-style paragraph and explain why you think it is suitable for topic classification for this dataset (for example, maybe other people have used this method for a similar problem). Use the implementation of this method from a standard machine learning library such as `sklearn` (**not** other people's code from the Internet) to implement this method on the music dataset using the same text preprocessing as for BNB and MNB. If the method has any hyperparameters for tuning, explain how you will select those settings (or use the default settings), and present a concrete hypothesis for how this method will compare to BNB and MNB.

Conduct experiments (and show the code for these experiments) using cross-validation and comment on whether you confirmed (or not) your hypothesis. Finally, compare this method to BNB and MNB on the metrics you used in Step 3 and choose one overall "best" method and settings for topic classification.

Part 2. Recommendation Methods

1. (6 marks) The aim is to use the information retrieval algorithms for "matching" user profiles to "documents" described in the lecture as a recommendation method. The overall idea is that the classifier from Part 1 will assign a new song to one of the 5 topics, and this song will be recommended to the user if the tf-idf vector for the song is similar to the tf-idf vector for the profile of the user in the predicted topic. The user profile for each topic will consist of the words, or top M words, representing the interests of the user in that topic, computed as a tf-idf vector across all songs predicted in that topic of interest to the user.

To get started, assume there is "training data" for the user profiles and "test data" for the recommender defined as follows. There are 1500 songs in each file. Suppose that the order in the file is the time ordering of the songs, and suppose these songs came from a series of weeks, with 250 songs from each week. Assume Weeks 1–3 (songs 1–750) form the training data and Week 4 (songs 751–1000) are the test data. After splitting the training set into topics, use `TfidfVectorizer` on the documents in a topic to create a tf-idf matrix that defines a vector for each document (song) in that topic in the training set (so constuct 5 such matrices).

Use these tf-idf values to define a *user profile*, which consists of a vector for each of the 5 topics. To do this, for each topic, combine the songs from the training set predicted to be in that topic that the user "likes" into one (larger) document, so there will be 5 documents, one for each topic, and use the vectorizer defined above to define a tf-idf vector for each such document (topic).

Unfortunately we do not have any real users for our recommender system (because it has not yet been built!), but we want some idea of how well it would perform. We invent two hypothetical users, and simulate their use of the system. We specify the interests of each user with a set of keywords for each topic. These user profiles can be found in the files `user1.tsv` and `user2.tsv` where each line in the file is a topic and (followed by a tab) a list of keywords. All the words are case insensitive. **Important: Although we know the pairing of the topic and keywords, all the recommender system "knows" is what songs the user liked in each topic.**

Develop user profiles for User 1 and User 2 from the simulated training data (**not** the keywords used to define their interests) by supposing they liked all the songs from Weeks 1–3 that matched their interests and were predicted to be in the right topic, i.e. assume the true topic is not known, but instead the topic classifier is used to predict the song topic, and the song is "shown" to the user under that topic. Print the top 20 words in their profiles for each of the topics. Comment if these words seem reasonable.

Define another hypothetical "user" (User 3) by choosing different keywords across a range of topics (perhaps those that match your interests or those of someone you know), and print the top 20 keywords in their profile for each of their topics of interest. Comment if these words seem reasonable.

2. (6 marks) Suppose a user sees N recommended songs and "likes" some of them. Choose and justify appropriate metrics to evaluate the performance of the recommendation method. Also choose an appropriate value for N based on how you think the songs will be presented. Pay attention to the large variety of songs and the need to obtain useful feedback from the user (i.e. they must like *some* songs shown to them).

Evaluate the performance of the recommendation method by testing how well the top N songs that the recommender suggests for Week 4, based on the user profiles, match the interests of each user. That is, assume that each user likes all and only those songs in the top N recommendations that matched their profile for the predicted (not true) topic (where N is your chosen value). State clearly whether you are showing N songs in total or N songs per topic. As part of the analysis, consider various values for M, the number of words in the user profile for each topic, compared to using all words.

Show the metrics for some of the matching algorithms to see which performs better for Users 1, 2 and 3. Explain any differences between the users. On the basis of these results, choose one algorithm for matching user profiles and songs and explain your decision.

Part 3. User Evaluation

1. (5 marks) Conduct a "user study" of a hypothetical recommender system based on the method chosen in Part 2. Your evaluation in Part 2 will have included a choice of the number N of songs to show the user at any one time. For simplicity, suppose the user uses your system once per week. Simulate running the recommender system for 3 weeks and training the model at the end of Week 3 using interaction data obtained from the user, and testing the recommendations that would be provided to that user in Week 4.

Choose one friendly "subject" and ask them to view (successively over a period of 4 simulated weeks) N songs chosen at random for each "week", for Weeks 1, 2 and 3, and then (after training the model) the recommended songs from Week 4. The subject could be someone else from the course, but preferably is someone without knowledge of recommendation algorithms who will give useful and unbiased feedback.

To be more precise, the user is shown 3 randomly chosen batches of N songs, one batch from Week 1 (N songs from 1–250), one batch from Week 2 (N songs from 251–500), and one batch from Week 3 (N songs from 501–750), and says which of these they "like". This gives training data from which you can then train a recommendation model using the method in Part 2. The user is then shown a batch of *recommended* songs from Week 4 (N songs from 751–1000) in rank order, and metrics are calculated based on which of *these* songs the user likes. Show all these metrics in a suitable form (plots or tables).

Ask the subject to talk aloud but make sure you find out which songs they are interested in. Calculate and show the various metrics for the Week 4 recommended songs that you would show using the model developed in Part 2. Explain any differences between metrics calculated in Part 2 and the metrics obtained from the real user. Finally, mention any general user feedback concerning the quality of the recommendations.

Submission and Assessment

- **Please include your name and zid at the start of the notebook.**
- Make sure your notebook runs cleanly and correctly from beginning to end.
- Do **not** clear cells in the notebook before submission.
- Make sure your plots are in your notebook and **not** loaded from your file system.
- Submit your notebook file using the following command:

  ```
  give cs9727 asst <zid>.ipynb
  ```

  You can check that your submission has been received using the command:

  ```
  9727 classrun -check asst
  ```

- Assessment criteria include the correctness and thoroughness of code and experimental analysis, clarity and succinctness of explanations, and presentation quality.

Plagiarism

Remember that ALL work submitted for this assignment must be your own work and no sharing or copying of code or answers is allowed. You may discuss the assignment with other students but must not collaborate on developing answers to the questions. You may use code from the Internet only with suitable attribution of the source. You may not use ChatGPT or any similar software to generate any part of your explanations, evaluations or code. Do not use public code repositories

on sites such as github or file sharing sites such as Google Drive to save any part of your work – make sure your code repository or cloud storage is private and do not share any links. This also applies after you have finished the course, as we do not want next year's students accessing your solution, and plagiarism penalties can still apply after the course has finished.

All submitted assignments will be run through plagiarism detection software to detect similarities to other submissions, including from past years. You should **carefully** read the UNSW policy on academic integrity and plagiarism (linked from the course web page), noting, in particular, that *collusion* (working together on an assignment, or sharing parts of assignment solutions) is a form of plagiarism.

Finally, do not use any contract cheating "academies" or online "tutoring" services. This counts as serious misconduct with heavy penalties up to automatic failure of the course with 0 marks, and expulsion from the university for repeat offenders.