COMP9727: Recommender Systems

Assignment 1: Content-Based Movie Recommendation

Due Date: Week 5, 10:00 p.m.

Value: 30%

This assignment is a practical application of recommender systems. The task is to build a content-based "movie recommender" such as might be used by a streaming service (such as Netflix) or review site (such as IMDb) to give users a personalized list of movies 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. Note that while movie recommender systems commonly make use of user ratings, our scenario is not unrealistic as often all that a movie recommender system has are basic summaries of the movies and the watch histories of the users.

For this assignment, you will be given a collection of 2000 movies that have been labelled as one of 8 main genres (*topics*): *animation*, *comedy*, *drama*, *family*, *horror*, *romance*, *sci-fi* and *thriller*. The movies of each genre are in a separate `.tsv` file named for the genre (such as `animation.tsv`) with 7 fields: *title*, *year*, *genre*, *director*, *cast*, *summary* and *country*.

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 (genre) classification in this domain/dataset. For this purpose, simply concatenate all the information for one movie 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 movies 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 movies to show the user at any one time, and how to get feedback from them on which movies they would click on and which movies 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 [illegible]u through this assignment. Your answer to each question should be in a sep[illegible]ection of the Jupyter notebook you submit. Each answer should contain a n[illegible]and code. Use comments in the code to explain any code that you think re[illegible] The "readers" here are students similar to yourselves who know someth[illegible]ning and text classification but who may not be familiar with the details of

Part 1. Topic (Genre) 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", and 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 movie 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 [illegible]mation retrieval algorithms for "matching" user profiles to "documents" d[illegible] as a recommendation method. The overall idea is that the classifier from [illegible]ew movie to one of the 8 genres, and this movie will be recommended to t[illegible]ctor for the movie is similar to the tf-idf vector for the profile of the user [illegible] The user profile for each genre will consist of the words, or top M words, r[illegible]ts of the user in that genre, computed as a tf-idf vector across all movies p[illegible] of interest to the user.

To get started, a[illegible]ng data" for the user profiles and "test data" for the recommender defined as follows. There are 250 movies in each file. Suppose that the order in the file is the time ordering of the movies, and suppose these movies came from a series of weeks, with 50 movies from each week. Assume Weeks 1–3 (movies 1–150) form the training data and Week 4 (movies 151–200) are the test data. Use `TfidfVectorizer` on all documents in the training data to create a tf-idf matrix that defines a vector for each document (movie) in the training set.

Use these tf-idf values to define a *user profile*, which consists of a vector for each of the 8 genres. To do this, for each genre, combine the movies from the training set predicted to be in that genre that the user "likes" into one (larger) document, so there will be 8 documents, one for each genre, and use the vectorizer defined above to define a tf-idf vector for each such document (genre).

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 genre. These user profiles can be found in the files `user1.tsv` and `user2.tsv` where each line in the file is a genre and (followed by a tab) a list of keywords. All the words are case insensitive. **Important: Although we know the pairing of the genres and keywords, all the recommender system "knows" is what movies the user liked in each genre.**

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 movies from Weeks 1–3 that matched their interests and were predicted to be in the right category, i.e. assume the true genre is not known, but instead the topic classifier is used to predict the movie genre, and the movie is shown to the user listed under that genre. Print the top 20 words in their profiles for each of the genres. Comment if these words seem reasonable.

Define another hypothetical "user" (User 3) by choosing different keywords across a range of genres (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 movies 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 movies will be presented. Pay attention to the large variety of movies and the need to obtain useful feedback from the user (i.e. they must like *some* movies shown to them).

Evaluate the performance of the recommendation method by testing how well the top N movies 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 movies in the top N recommendations that matched their profile for the predicted (not true) genre (where N is your chosen value). State clearly whether you are showing N movies in total or N movies per genre. As part of the analysis, consider various values for M, the number of words in the user profile for each genre, 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 movies and explain your decision.

Part 3. User Evaluation

1. (5 marks) Conc[illegible] hypothetical recommender system based on the method chosen in Part 2. [illegible] 2 will have included a choice of the number N of movies to show the user [illegible] simplicity, suppose the user uses your system once per week. Simulate ru[illegible]r system for 3 weeks and training the model at the end of Week 3 using i[illegible]d from the user, and testing the recommendations that would be provided[illegible]4.

Choose one friend[illegible]hem to view (successively over a period of 4 simulated weeks) N movies chosen [illegible] each "week", for Weeks 1, 2 and 3, and then (after training the model) the recommended movies 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 movies, one batch from Week 1 (N movies from 1–50), one batch from Week 2 (N movies from 51–100), and one batch from Week 3 (N movies from 101–150), 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* movies from Week 4 (N movies from 151–200) in rank order, and metrics are calculated based on which of *these* movies 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 movies they are interested in. Calculate and show the various metrics for the Week 4 recommended movies 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.**
- Submit your notebook files 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 (linked from the course web page), noting, in particular, that *collusion* (working together on assignment, or sharing parts of assignment solutions) is a form of plagiarism.

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