Lab 10, Joke Recommendations

Due date: Wednesday, Friday, May 29, 11:59pm.

Please note: There may be a small individual lab assigned in parallel at some point early next week.

Overview

For this lab you will work on two problems.

First, you will implement and test a simple joke score predictor. Your goal is to test how well the Jester data that is avialable to you allows one to predict a person's attitude towards a specific joke.

Second, you will attempt to identify and study the behavior of some groups of users of Jester. Of interest to us are people who show similar sense of humor to each other, but whose sense of humor appears to be different from that of others. This part of the assignment is open-ended — you will have to come up with the methodology for doing this on your own based on what you have developed in prior labs, and based on what we have discussed and will be discussing in the class. Consider it a preview of your final project.

Assignment Preparation

This is an team assignment. Teams of three people are the default teams on this lab, I will allow for two teams of two, or, for one team of four to accommodate the class size.

For this assignment you will build a collection of Python functions, and a number of notebooks (at least one per question studied). Everything you build shall be from scratch, i.e., no notebooks are provided to you. See below for a more detailed list of deliverables.

Data

*This section is a copy of the **Data** section of Lab 7 document. It is kept here to give you easier access to reference material.*

You will be using *joke ratings data* from the **Jester** project, run by Professor Ken Goldberg at UC Berkeley. **Jester**[1] is an on-line joke recommender system available at

http://shadow.ieor.berkeley.edu/humor/

Disclaimer. Please read this note before proceeding! Jester has a database consisting of 100 jokes. The jokes are shown to the user, and the user's reaction to them is measured on a continuous scale. **Please be aware** that the jokes available through `Jester` may contain some examples that you personally will find objectionable. Please know, that *it is not my intent to offend anyone's sensibilities* (and neither is it the intent of the authors of the dataset and **Jester**).

The Dataset

The portion of the **Jester** dataset that you will be studying consists of two files uploaded to the `/home/dekhtyar/data` directory on the Jupyter server.

List of jokes. The first file is `Jokes.xml`. This file contains the list of jokes that **Jester** uses. The file has the following simple format:

```
<jokes>
  <joke>
    Joke 1 goes here...
  </joke>
   ...
  <joke>
     Joke 100 goes here...
  </joke>
</jokes>
```

There are 100 jokes in the file — these are all the jokes **Jester** uses. This file can be easily parsed with `BeautifulSoup` using the techniques you have aquired from the previous labs. The jokes are not numbered internally in the XML file, but their order corresponds to the joke ids in the ratings data file.

Ratings Data. The `jester-data-1.csv` file contains our joke ratings data. The full dataset contains three data files of roughly equal size. Of the three files, you will be using only the first one, `jester-data-1`.

The **Jester** dataset web page describes the format of the data as follows[2]:

"Format:

1. 3 Data files contain anonymous ratings data from 73,421 users.
2. Data files are in .zip format, when unzipped, they are in Excel (.xls) format
3. Ratings are real values ranging from -10.00 to +10.00 (the value "99" corresponds to "null" = "not rated").
4. One row per user
5. The first column gives the number of jokes rated by that user. The next 100 columns give the ratings for jokes 01 - 100.
6. The sub-matrix including only columns $\{5, 7, 8, 13, 15, 16, 17, 18, 19, 20\}$ is dense. Almost all users have rated those jokes (see discussion of "universal queries" in the above paper)."

There are 24,983 rows (ratings records) in the `jester-data-1.csv` file. Each row has 101 values as described above.

Lab Assignment

The Jester dataset has been developed for the purpose of studying the ability of data analytical methods to predict human reactions to jokes, and to be able to recommend the jokes someone might find funny.

In this lab you will finally get to use this dataset for its core purpose.

In all, you will study two questions:

Question 1: Accuracy of prediction for a number of simple prediction methods. You will implement a number of simple joke score prediction methods and will study their accuracy. Of our primary interest is to see whether methods that rely on predicting the scores based on user similarity are more accurate than methods that predict scores based on everyone's behavioral patterns. In addition to implementing all the prediction methods, you will also conduct the appropriate accuracy study and will report the accuracy results comparing the methods you have implemented.

Question 2: Discovery of user communities. You will conduct an exploartory study designed to find one or more communities of Jester users who show very similar sense of humor. This question is open-ended. Each team needs to determine a strategy for such a discovery, methods of analysis and visualization, and implement it. I expect different results from different teams simply because of difference in methodology and difference in the data chosen as the starting point.

These questions are addressed below.

Deliverables

Each team shall work on one set of deliverables. The deliverables are a combination of Python files and Jupiter notebooks together with any other supplementary information. You do not need to submit the data file, but you *may*, as part of your data analytical process, extract portions of it, model them as desired and store them persistently in files.

Most of your functionality actually doing the work shall be "hidden" in Python functions (and classes) you implement. The two notebooks you will create - one for each question, shall document your actual studies. You can `import` your Python function collections and any other code into your notebooks to make it available, but I do not want the notebooks cluttered with hundreds of lines of utility code as we had to do on occasion in previous labs. Build libraries of Python functions, import them into notebooks, use them for demonstratable effects.

The notebooks are your reports/narratives. They should contain readable text explaining they work you have conducted, interspersed with Python code implementing the specific work - running the experiments, visualizing results, and so on. The notebooks shall be organized as academic reports, with the following information clearly present:

1. Introduction, overview of your study.
2. Description of each step of your study, with appropriate code embedded.
3. Results.
4. Analysis and discussion.

Question 1: Score Prediction

Please refer to the accompanying handout discussing recommendations and utility function predictions for the mathematics of the methods you will be implementing.

The key questions you are studying is:

> *Can we predict reliably, based on the* Jester *data available to use, the score a person would give to a specific joke?*
>
> *How accurate would such predictions be?*
>
> Which of the methods discussed in class would give the best predictions?

To answer this questions, you will conduct the following activities.

Step 1. Implement score prediction methods. You will implement the following score prediction methods discussed in class:

1. Collaborative predictions: given a person and a jokeId, predict the score based on the person's individual preferences and the similarities of the person's preferences to the preferences of all others.
2. Item-based predictions: given a person and a jokeId, predict the score based on the everyone's attitude to the specific joke and the similarity of this attitude to the ratings of other jokes.
3. Nearest Neighbor Collaborative predictions: just like collaborative predictions but restricted to N nearest neighbors of the current person in terms of similarity of sense of humor (you may need to select an appropriate N).
4. Nearest Neighbor Item-based predictions: just like item-based predictions but restricted to N jokes most similar, in terms of user attitude, to the current joke.

Each of the prediction categories have two-three methods you need to implement (average, weighted sum, adjusted weighted sum).

Implement all your prediction functions in a single Python file.

Step 2. Conduct accuracy of prediction studies. You need to conduct an accuracy study for each method you have implemented. There is a number of ways in which an accuracy study can be conducted. Some of them are discussed in class. Some of the options include:

- Reserved Set. Randomly select a set of `<user, joke>` pairs for which ratings are available, and remove them from the Ratings matrix (replace with the null value). Predict the rating for each reserved pair using each of your method. Study the overall accuracy of each method on the reserved set. Repeat if necessary for replication purposes.
- Cross-validation. Choose a number $K > 0$ (you may want a reasonably large K as your dataset contains a lot of data points). Split your dataset into an K sets of `<user, joke>` pairs of equal size. On each of the following K steps, select one set and reserve it. Predict the scores from this set, using the remaining $K - 1$ sets for prediction purposes. Compute the overall accuracy of predictions. This method allows you to make a prediction about every `<user,joke>` pair. (Note: only select `<user, joke>` pairs for users who did give a rating to the joke).
- All-but-one validation. For each non-empty rating in-turn, reserve it. Use the remainder of the dataset to predict its rating. Record the accuracy. This allows you to make a prediction about every `<user, joke>` pair using the maximal amount of information available.

Design and conduct an appropriate accuracy study. The error of each individual prediction can be measured as follows:

$$error(user, joke) = |predicted(user, joke) - rating(user, joke)|$$

Beyond that, you can compute overall accuracy in a number of ways.

- *Sum Squared Error.* Add up the squares of all the errors.
- *Average Sum Squared Error.* Find the average squared error.
- *Threshold Accuracy.* Pick an $\epsilon > 0$. We consider the prediction to be accurate at the threshold ϵ if $error(user, joke) \leq \epsilon$. The accuracy of a method at threshold ϵ is the percentage of the method's predictions that were accurate at this threshold.
- *Sentiment Accuracy.* A prediction $predicted(user, joke)$ is considered to *have the right sentiment* if

  $$sign(predicted(user, joke) = sign(rating(user, joke),$$

  that is, if both the prediction and the rating share the same sign. The sentiment accuracy of a method is the percentage of the method's predictions that have the right sentiment.

So... Select the study protocol from the options above. Select one or more accuracy measures (I suggest implementing as many as you can). Measure the accuracy of each method using each of the accuracy measures.

For Threshold Accuracy you can conduct the study determining the relationship between the ϵ and the threshold accuracy at ϵ for each method (so that you could actually plot the accuracy in a nice graph).

Step 3. Visualize your results. Report the results you have observed in the Jupiter notebook using appropriate visualization techniques.

I am leaving it up to individual teams to determine what is the best way to visualize the results. I am happy to discuss the visualization with each team as the lab proceeds, and give suggestions.

Step 4. Analysis. Present your conclusions. Which method or methods are the most accurate? Why? What methods are faster? What methods are slower? Is there a relationship between the efficiency/speed of a method and its accuracy? (note: you can use `%timeit` directives to evaluate the time it took your methods to run). Are nearest-neighbor-based methods more accurate?

Question 2: Community Discovery

The core question I am asking you to study here is this:

> *Are there any groups of users that show very similar sense of humor to each other, but whose sense of humor appears to be different from that of all other users? Can you find one or more such groups?*

This is an **open-ended question.** There is are some advanced techniques traditionally used to study questions like this in a more strict and formal way. What I am trying to do is to see if you can come up with some *ad-hoc* ways of *separating some users from the herd.*

Some hints to how to approach this problem are below.

1. Explore the data. You will not be able to find anything unless you actually explore your data. Use the code from your past labs (content of the jokes, t-test, linear regression), as well as `matplotlib` to compare things to each other and visualize portions of the dataset. Eyeball the visualizations, see if you can find something interesting.

2. Talk. The key reason why this is a team assignment is to get you to discuss data analysis with your teammates. Data Science is a collaborative sport. Please remember that.

3. Understand what to look for. We are looking for people with peculiar tastes different from the tastes of others. The groups you may discover do not have to be large - in my prep for this lab I identified a group that wound up having only 11 people, but that exhibited very specific unexpected behavior.

4. Select your features. You have 100 jokes, but maybe not all jokes are good "features" to represent humans. Figure out which ones may be ignored, and you solve most of your problems. Similarly, look at the content of the jokes (this is why you did the inverted indexing exercise) and see if the content of the jokes may help.

5. Look for outlier behaviors. You have over 24 thousand data points. If you build a scatter-plot of anything against anything with your data, you will see the space filled with data. What you want to look for is users who exhibit behaviors that are not exhibited by the vast majority of other users. Find a group of people with such a behavior, and see if you can extrapolate it into a recognizable patterns that separates the "outlier" users from others.

6. Document. You can build as many notebooks for exploring the data. Make notes in them, come back to them later to cross-reference, and eventually, combine the best results you are seeing in your final notebook. You can submit your draft notebooks as well so that we could see your exploration.

7. Explain. You should be able to explain in English the specific behaviors you observed that made a group of users you discovered different from other users. Please be aware of that. As with Lab 5, you will be explaining these results to others in the class at some point.

Deliverables and submission instructions

Only one lab submission per team is needed. Please make sure that all your files: Python code, Jupiter Notebooks, and any other file you created contain the names and email addresses of all team members.

Develop your lab in a `Lab10` directory (you can fetch an empty directory using `nbgrader fetch --course=dekhtyar Lab10` command). Once finished, submit your lab using `nbgrader`:

```
$ nbgrader submit --course=dekhtyar Lab10
```

References

[1] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval*, 4(2), 133-151. July 2001.

[2] Ken Goldberg, Anonymous Ratings Data from the Jester Online Joke Recommender System, http://www.ieor.berkeley.edu/~goldberg/jester-data/.