Yelp dataset

* Using only reviews data, build a simple recommendation system first
  + Matrix factorization – derive embedding vectors for users and restaurants
* Next stage ideas
  + Clustering food preferences.
    - E.g. people who like Indian food have taste preferences similar to people who like Sri Lankan food
    - …..and maybe interesting insights like “people who prefer Swedish food have tastes very different from all other preference-based groups of people”
* train classifier on reviews’ text to predict rating from review text. Then, predict ratings for each tip. This predicted rating will be supplementary data for the recommendation system.
* Incorporate bias?

############################

Before you do any commit through any instances, you need to do the following operations. This will help count your commits to your contributions in repo:

git config --local user.name "other-user-name"

git config --local user.email other-email-address

######################################################

**Notes on EMR setup**

Jan 15, 2018

1. **AWS blog on EMR setup and a few other useful things:**

<https://aws.amazon.com/blogs/big-data/building-a-recommendation-engine-with-spark-ml-on-amazon-emr-using-zeppelin/>

Steps after connection (on AWS):

sudo yum install mosh

then, exit AWS and reconnection using mosh:

mosh --ssh="/usr/bin/ssh -i spark\_kp.pem" hadoop@ec2-52-36-178-189.us-west-2.compute.amazonaws.com --server="/usr/bin/mosh-server"

1. **Adding new users to your AWS instance/EMR:**

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/managing-users.html>

Make new user sudo user (increase permissions)

<https://www.helicaltech.com/create-multiple-sudo-users-to-ec2-amazon-linux/>

* Edit visudo file

<https://www.garron.me/en/linux/visudo-command-sudoers-file-sudo-default-editor.html>

* Note: the new user can now do ‘sudo su’ and assume the role of root user. This allows

Steps for a user

: Give your user.pub file to kunal

: ssh -i user.pem sooraj@ec2-54-202-177-212.us-west-2.compute.amazonaws.com

: spark-submit --packages org.mongodb.spark:mongo-spark-connector\_2.11:2.2.0 ./codes/CF.py > /home/hadoop/output1.txt --driver-memory 45G --worker-memory 45G

Steps for mongo configuration for spark

:Follow Diane’s slide

Steps for EMR cluster

:Follow the article on the EMR optimization

:Open Port 22 , 27017

KK Notes

Run spark on EMR:

pyspark --packages org.mongodb.spark:mongo-spark-connector\_2.11:2.2.0 --master yarn --driver-memory 20G --driver-cores 4 --executor-cores 4 --executor-memory 20G

Naïve Bayes notes:

<http://blog.datumbox.com/machine-learning-tutorial-the-naive-bayes-text-classifier/>

Usually Multinomial Naive Bayes is used when the multiple occurrences of the words matter a lot in the classification problem. Such an example is when we try to perform Topic Classification. The Binarized (Boolean) Multinomial Naive Bayes is used when the frequencies of the words don’t play a key role in our classification. Such an example is Sentiment Analysis, where it does not really matter how many times someone mentions the word “bad” but rather only the fact that he does.

* Binarized (Boolean) Multinomial Naive Bayes : instead of measuring all the occurrences of the term t in the document, it measures it only once
* <https://gist.github.com/mitallast/87f0d0c5a8e5447c1626>

Spark notebook on ML examples:

<https://github.com/apache/spark/tree/master/examples/src/main/python/ml>

ngrams example:

<https://github.com/apache/spark/blob/master/examples/src/main/python/ml/n_gram_example.py>

**try StopWordsRemover**

great page for handy spark-ml functions:

<https://spark.apache.org/docs/2.1.0/ml-features.html>