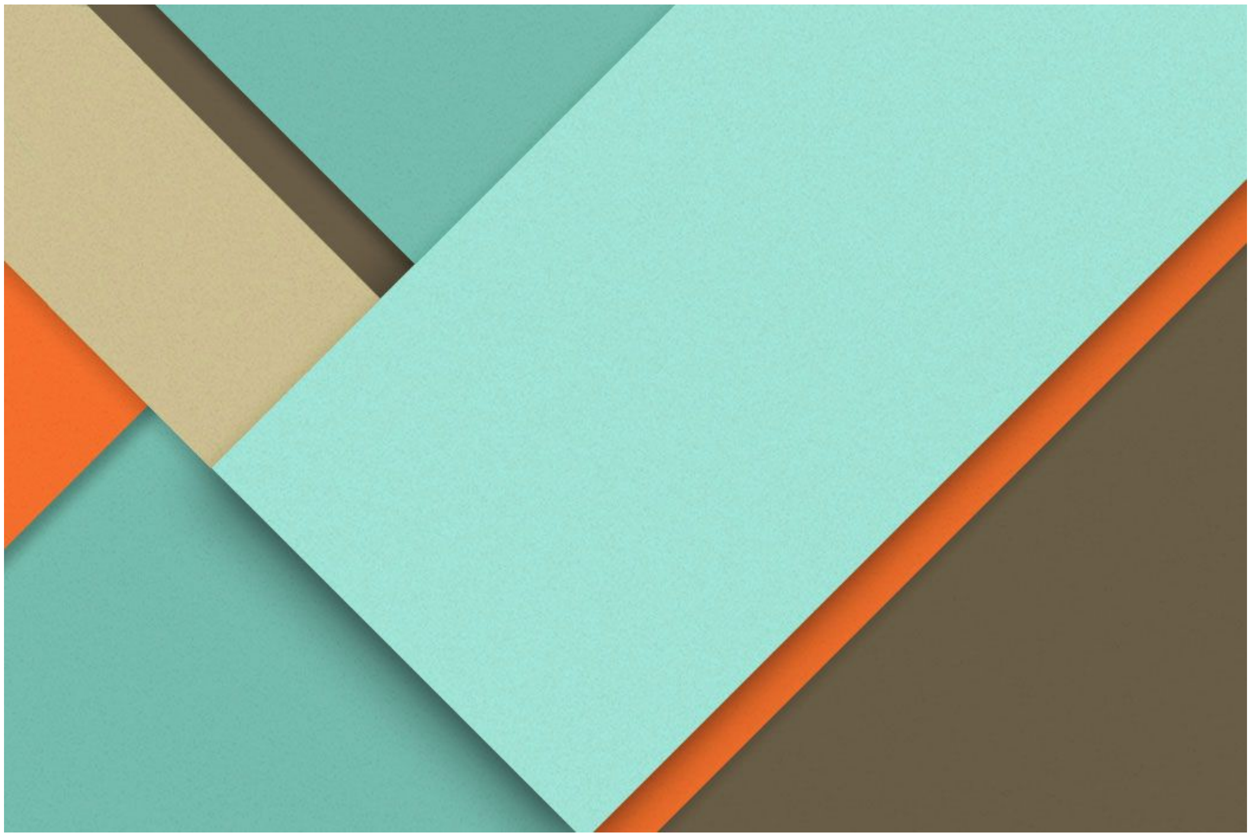

Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα Εργασία 2

Ονοματεπώνυμο: Μπαρμπάρ Ηλίας-Ελιάς
ΑΜ: 1115201200118

Summary

This program implements two cryptocurrency Recommendation methods, Cosine LSH and Clustering. Given a .csv file of tweets, the program calculates a sentiment score for each tweet and then proceeds to creating the User-Cryptocurrency vectors. We remove users that provide no information (users with 0^d vectors, or users that haven't mentioned any cryptocurrency in their tweets) and we proceed with our recommendation. The program also implements a 10-fold cross validation for both methods with the -validate command line argument).

Files

mainRecc.cpp:

Summary gives a good idea of how our main works. Additional info: We use the static int virtualOpt, to define whether we are running with normal users or virtual ones.

recommendationFunctions.cpp/.hpp:

In here we have the functions used in our main. The function names speak for themselves.

Functions:

- sentimentScoresCalculation.
- userCryptoVectorCreation.
- removeZeroVectors.
- calculateRatingMean.
- createVirtualUsers.
- prerecClusteringFunc.
- prerecCosineFunc.
- Recommend.

simplefunctions.cpp/.hpp:

In here we have all around simple functions.

Functions:

- setVectors
- functionVec
- searchInLexiconA
- searchInLexiconK
- checkIfZeroVector
- compareByDist
- findInTweets
- AsetUserCryptLoc
- shuffleArray, kfold: These two functions are influenced by http://fernandojsg.com/project/kfold-cross-validation/

readFileInput.cpp/.hpp:

Reads the input file and creates our tweets and our normal users.

readCmdRec.cpp/.hpp:

Reads the command line argument.

readLexicon.cpp/.hpp:

Reads the the lexicons we are using for this project.

readFileCsv.cpp/.hpp:

Reads the training set from the csv file we were given.

myTweet/myLexicon/myVector.hpp:

Structs for our data.

Instructions

We use make to compile our program.

A call for our program should look like this:

./recommendation -d <input file> -o <output file> or ./recommendation -d <input file> -o <output file> -validate

If we want to use the 10-fold cross validation option.

Compilation

We compile using a makefile so all the user has to do is type ‘make’ which has the following form:

```
recommendation:
	g++ mainRecc.cpp readCmdRec.cpp LSH.cpp HyperCube.cpp kmeans.cpp gfunction.cpp cosineH.cpp euclideanH.cpp simplefunctions.cpp recommendationFunctions.cpp -o recommendation


.PHONY: recommendation
```

Results

Calculating rating mean

Normal Users:

Without filling with rating mean before Cosine lsh: 0.31
With … Cosine lsh: 0.33

Without filling with rating mean before Clustering: 0.7416
With … Clustering: 1.2445

Virtual Users:

Without filling with rating mean before Cosine lsh: 1.72
With … Cosine lsh: 1.75

Without filling with rating mean before Clustering: 1.77
With … Clustering: 2.81