A

Dissertation On (Major Project-II)

"Human Activity Recognition For Friend Recommendations"

Submitted in Partial Fulfillment of the Requirement
For the Award of Degree of

Master of Technology

In

Software Technology

By

Amandeep Singh
University Roll No. 2K15/SWT/505

Under the Esteemed Guidance of

Prof. Rajni Jindal
Professor & Head of Department, Department of Computer Science & Engineering

2015-2019(Jan)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
DELHI – 110042, INDIA

STUDENT UNDERTAKING

Delhi Technological University
(Government of Delhi NCR)
Bawana Road, Delhi- 110042

This is to certify that the thesis entitled **"Human Activity Recognition for Friend Recommendations"** done by me for the Major project-II for the achievement of **Master of Technology** Degree in **Software Technology** in the **Department of Computer Science & Engineering**, Delhi Technological University, Delhi is an authentic work carried out by me under the guidance of Prof. Rajni Jindal.

Signature:
Student Name
Amandeep Singh
2K15/SWT/505

Above Statement given by Student is Correct.

Project Guide:
Prof. Rajni Jindal
Professor & Head of Department, Department of Computer Science & Engineering, DTU

ACKNOWLEDGEMENT

I would like to express sincere thanks and respect towards my guide **Prof. Rajni Jindal, Professor & Head of Department, Department of Computer Science & Engineering, Delhi Technological University Delhi.**

I consider myself very fortunate to get the opportunity for work with her and for the guidance I have received from her, while working on this project. Without her support and timely guidance, the completion of the project would have seemed a far. Special thanks for not only providing me necessary project information but also teaching the proper style and techniques of documentation and presentation.

AMANDEEP SINGH
M.Tech (Software Technology)
2K15/SWT/505

ABSTRACT

In this thesis, I have used human activity recognition to find out the performed activity of a user and thereby predicting the lifestyle of the user based on his activities which is hence used to construct a friend recommendation model.

With the advent of technology and social network platforms, friend suggestions are mostly based on the social graphs. Greater the number of mutual friends you have with a person, higher the chances of getting the friend recommendation of that person. Such an approach does not reflect end user's preference for friend selection. Thus, we have presented an approach for friend suggestions which is based on the similarity of the lifestyle of different users.

With the variety of sensors which are embedded in mobile devices, it has become convenient to analyze human activities. To implement an intelligent friend recommendation system, we made use of Convolution Neural Network (popularly known as CNN) for activity recognition by extracting the local feature and superimposing the statistical features, though making sure that information related to global features is preserved. Subsequent to discovering activities of user, Latent Dirichlet Allocation (LDA) [1] algorithms used to identify the life style. We additionally propose a similar metric algorithm in order to compute the similarity of life styles among users by a friend-similarity graph.

Whenever a user requests for friends, then my algorithm will return people's list sorted by recommendation score, from which user can choose whom to send request. The recommendation friends can be user's immediate friends or they can be friends of friends based on their scores.

Open dataset has been taken from http://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions# which contains user data for 30 person having 10402 data entries of sensor data. 21 friends are taken as a test case and a pictorial graph for user similarity is constructed. In the graph, white color box corresponds to a high similarity score and the darker the color, lower is the similarity score. So, this pictorial representation gives a clear picture for the closeness in users.

The one thing which has come out as a very strong argument and a contributing parameter in our model is the statistical parameters. Pictorial representation of the data features gave a pretty good idea about the different statistical features that can be used in order to classify among different classes more efficiently.

TABLE OF CONTENTS

Chapter 1: Introduction

1.1 PROBLEM STATEMENT

Traditionally, people used to be friends with those people who work closely with them or who live nearby. Such friends are known as Geographical friends [6]. The recent advent of technology and advances made in social networks has completely changed the idea of how we make friends. Many social networking platforms have completely dominated our social lives for some time now [4].

Some of the leading names that come into mind are Facebook, Twitter, Snapchat and Instagram. One of the most important challenges that these social networking platforms have to deal with is providing good friend suggestions to a particular user. Most recommendations are based on users' relationship with other individuals and based on the number of mutual friends one has with other individual. *Recent studies [12] have completely revolutionized the ideas of recommendation models and suggest that people, who have similar lifestyles, should form the basis of friend suggestions*.

In our day to day life, our lifestyle will be determined on the activities that we do throughout the day. Similar idea is adopted by us in our research. In our approach, we have used the term 'activity' to denote the action taken by the user, like "walking", "standing" and, "sitting", meanwhile using term 'daily-life-style 'for actions such as "shopping" or "office work" which is a collection of activities keeping the time into consideration. For example, "daily exercise" life style should consist of mainly "walking" as the activity, but might also consist of "sitting" or "standing". Probabilistic topic models treats document as combination of topics and collection of words for a topic [1]. In the same way, our daily lives, which are analogous to the documents is a combination of life styles, which are analogous to topics, and every life style is a collection of activities, which are analogous to words, as shown in Fig 1.

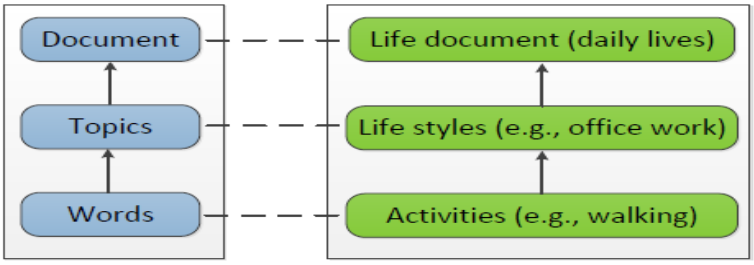


Fig 1: Relation between documents and Person's daily lives

1.2 WHAT ARE THE RECOMMENDATION SYSTEMS

Recommendation system is an information filtering system which removes unnecessary and redundant information and predicts a 'preference' to the end user.

Recommendation systems work in two way:-

Knowledge Based Filtering- is the process of information or pattern filtering using techniques which involve collaboration between multiple data sources, agents, viewpoints, , etc.

Content Based Filtering- makes use of keywords to describe the items and building a user profile which indicates what item an end user would like.

Both these approaches have their own strengths and weaknesses and hence both these approaches have been combined to make **Hybrid Recommender Systems**. Netflix and Amazon are prime examples of Hybrid recommendation systems as they add content based capabilities to collaborative study.

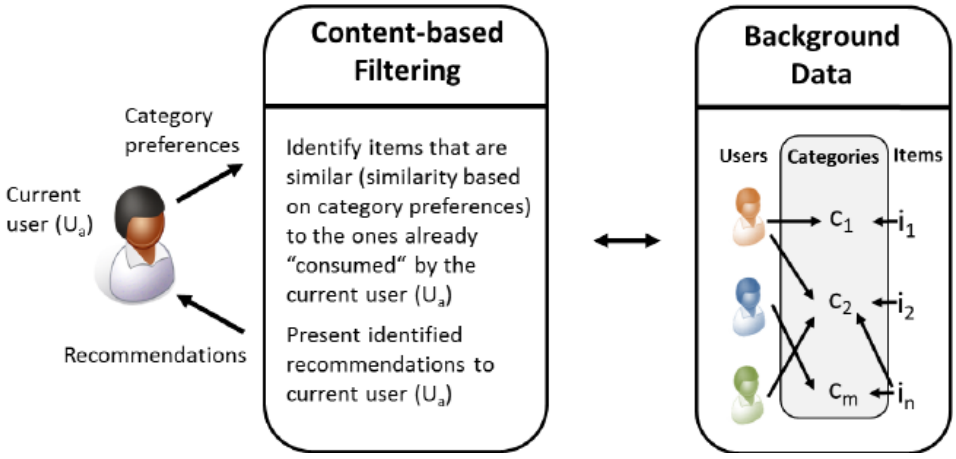


Fig 2: Content based filtering

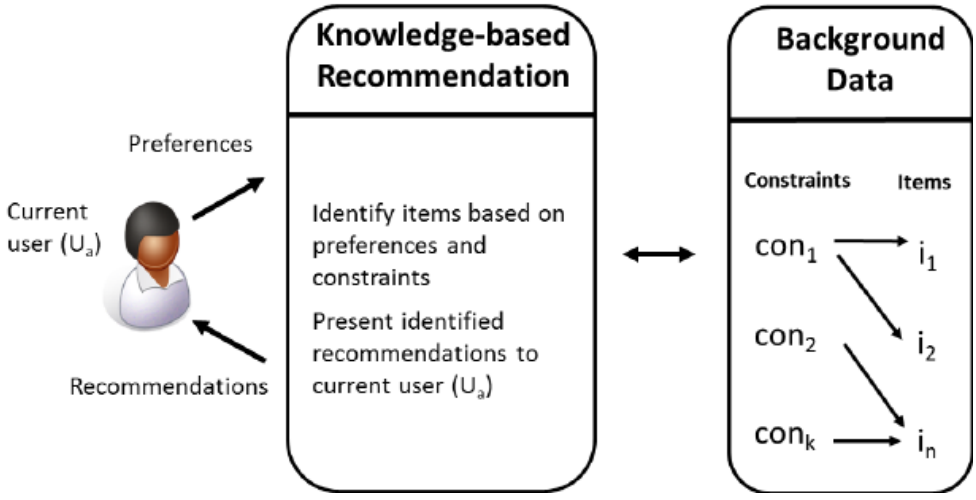


Fig 3: Knowledge based filtering

Measuring User Similarity–Once the life style vector of user is obtained, the next step is to measure how closely related he is with other users in order to give him the best recommendation. One of the vector similarities measuring technique is to calculate the cosine similarity between the two vectors. But in our research, we have introduced another measure along with cosine similarity to calculate appropriate similarity among users.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

1.3 THESIS MOTIVATION AND GOAL

Recommender systems mostly use historical data to filter possible choices for users related to their interests. Such systems have widespread usage in e-commerce, movie suggestions, and friend recommendations in social network etc.

Regarding friend recommendation many recommender system are based on user's friend graph, which do not model the real world scenario as the current recommendation systems employ the use of preexisting relationships of user, whereas many researches have recently countered this

approach and suggested that friend recommendations should be based on person's lifestyle(based on their daily activities) and behaviors

To address these shortcomings, we have been determined to provide a solution to recommend 'good' friends on the basis of their matching lifestyles and behaviors [12].
Throughout this thesis, we have tried to find out user characteristic and behavior, which will help in friend making and use this approach to drive an algorithm.

1.4 THESIS ORGANIZATION

This thesis is classified into six different chapters.

Chapter 1 defines the problem statement for the thesis, which is 'friend recommendation on the basis of social maps'. Such techniques have been shown to be inconsistent with user preferences and hence recommendation systems have been suggested based on user lifestyles in this thesis.

Chapter 2 describes the related work done in the areas of deep learning for Activity Recognition and Friend Recommendations on the basis of user's activities. This thesis deals with combining these two research areas and designing a combination which will be very efficient as a friend recommender on the basis of user's lifestyle.

Chapter 3 explains the research topics used in detail. In our research details, we explain the terms, which are being used in the thesis like Neural Networks, Deep Learning, Convolution Neural Networks, Topic Model, Cosine Similarity, Statistical Features like Mean/Median, Standard Deviation, Histogram values, Root Mean Squared values etc.

Chapter 4 is explaining the solution architecture of the CNN used for human activity recognition and explains about the algorithm used for friend recommendation based on the matching of user's lifestyle. It deals with the extraction of life style by topic model, where the Life style document consists of topics and topic consists of words/Activities. With Support development in the area of text mining, we have designed a model for user's daily lives as life document. Using

the “Document”, topics could be found by using this probabilistic topic model. Latent Dirichlet Allocation decomposition algorithm is used to find out the “life styles” probabilities from the “documents”. This chapter also deals with the activity recognition, in which Life style is considered as a combination of motion activities that have distinct occurrence probability. We have used two sensors, gyroscope and accelerometer, that are used to drive motion activities. We have made use of Convolutional Neural Network for this classification work. This chapter also deals with Friend-matching graph to depict the likelihood among user life style and hence a user querying system for friend recommendation is discussed.

Chapter 5 illustrates friend recommendation results. Using the representation of gray-scale image of 30 users, low recommendation score color is represented with green color followed with blue color and the white color, representing high recommendation score. The blocks that have green color which represents low recommendation score.

Chapter 6 is the conclusion of the thesis. It describes the benefits of that approach for recommending the friend on the basis of user’s behavior and their life style instead of following the social graph. The recommendation of friend by using this approach is more appropriate.

Chapter 2: Related Work

[1] Describes the Latent Dirichlet Allocation algorithm which is a flexible generative probabilistic model for text corpus used for topic modeling.

[2] Describes Convolutional Neural Network providing dimensionality reduction and uniform to minor changes in the image sample. It also provides partial invariance to scale, rotation, translation and deformation. The CNN extracts features in a set of layers. It detects events, and provides the estimation

[3] Presents how recognition accuracy is affected by time series length. It uses Convolutional neural network to solve the recognition task.

[4] Presents Matchmaker, it is their friend recommendation system which helps to recommend friends based on their personality matching. Their system aimed at recommending the users based on their social network connections and based on their physical world interaction.

The paper [5] combines social and physical context on social networking sites to aim to recommend the truly valuable friends. The concern with this approach is that it does not consider the lifestyle of the users.

[6] Aims at combining the user's social networking platform with their geographical position to recommend friends to them. They build a heterogeneous information network using geographical information and social networking information and then used this network to recommend friends to the users.

[7] Presents how local dependency can be captured along both temporal and spatial domains for multi-modal data, making use of 2D CNN, so that it achieves high performance with low number of parameters compared to 1D operation. This paper achieved very good results using 2D CNN approach. [9] Had also used Convolutional Neural Networks to extract the local dependency and scale invariance of a signal and achieved state of the art results at that time.

[8] Proposes SFViz (Social Friends Visualization), which allows users to find friends based of their interests. Their approach makes use of both topological structures in social networks and semantic structure of activity data .

[10] Also used Deep Convolutional Neural Networks on smart phone sensor data and derived relevant and more complex features. Their method achieved very good accuracy on moving activities.

[11] Focuses on detecting fall activities by using two sensors, accelerometer and gyroscope data.

Friend suggestion by using the daily activities of a user is explored in [12]. The paper generates a life style vector for every user and then calculates the similarity between different vectors using cosine similarity and another similarity measure. Friend matching graph is generated by them which enable friend recommendation very easily.

[13] Shows real time classification of different types of human movements using waist-mounted triaxial accelerometer. It detects events, and provides the estimation of metabolic energy expenditure.

[14] Single, waist-mounted triaxial accelerometer provides signal which are used to identify the movements of the user using the binary decision tree. The activities were classified as walking, transition between postural orientations, falls or other movement.

[15] Uses frequency-domain entropy, mean, energy and correlation of acceleration data and tested several classifiers using this data. Decision tree outperforms other.

Base-level and meta-level classifiers performance is compared in [16] for accelerometer data. Plurality Voting is found to get the best results.

[17] Compares multiple time domain feature sets and sampling rates, and compares between computational complexity and recognition accuracy.

Augmented features are used in [18] for classification of the activities. The features include signal magnitude areas (SMA), autoregressive (AR) modeling coefficients of activity signals and title angles (TA). They demonstrated that high accuracy is possible to achieve with low run time complexity of the model.

Chapter 3: Research Background

3.1 Convolution Neural Network [2] (CNN)

Neural Network is a technique in machine learning which has been inspired from the structure of the brain. The main learning unit/cell of the network is called a neuron. The neurons are the basic building block of the whole network which learns all sorts of things ranging from an edge in a picture to semantic meaning of a complex sentence.

In normal neural network architecture like MLP (Multi-layer Perceptron, the input is an array/vector. Whereas CNN [2] is inspired from the vision in humans so it takes input a multi-channeled image as its input.

CNN comprises of 4 main steps: convolution, pooling, activation and fully connected layers.

Step 1: Convolution

The starting layers of a CNN which takes input are the convolution layers. Convolution is basically recognizing any hidden features in the image by looking at the previously learnt weights by the convolution layers' weights. Input signals are analyzed to find out if similar objects have been identified before, and if reference signals are found to be similar then they will be mixed into, or convolved with, the input signal. In this way the method continues for multiple layers and resulting weights are carried forward.

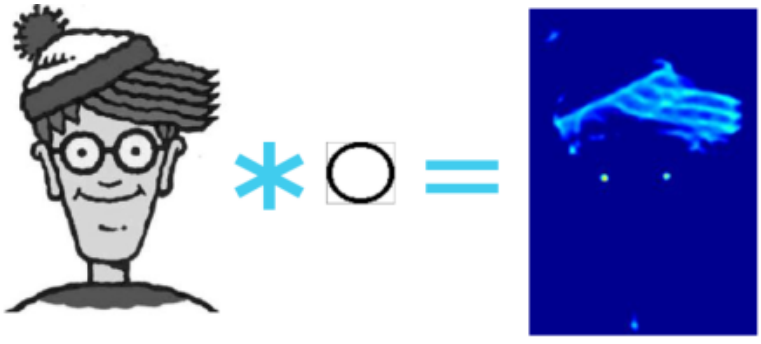


Fig 4: Convolution of face with round filter

Convolving a cartoon face image with a circle filter is expected to highlight the eyes of the cartoon. The circle filter is expected to respond strongly to the eyes area.

Convolution is **translational invariant**, which means, each convolution filter presents a different set if feature (e.g. circles, edges etc.) and the CNN learn which features consist of the resulting reference. The output signal strength is dependent only on whether the feature is present or not , it does not depend upon where the feature is located. Hence, a dog could be lying in different ways in the images, and the CNN algorithm would still be able to identify it.

Step 2: Pooling

Sensitivity of the filters to noise and variation in the convolved images can be reduced by smoothing the inputs. This smoothing process is called **pooling**, and is of many types, namely average pooling where the average over a sample of the input is taken and other is max pooling where the maximum is taken over a sample of the signal. Pooling methods examples would be: reducing the color contrast across red, green, blue (RGB) channels or image dimensionality.

Fig 5: Pooling creating a low resolution image

Step 3: Activation

The task of activation layer is to pass the result from one layer to another after applying some non-linearity to it. Output signals which have strong assosciation with prior references would activate more neurons, enabling these output signals to be propagated for identification in more efficient way.

CNN is generally used with a wide range of complex activation functions, where the most common is the *Rectified Linear Unit (ReLU)*, as it trains quite faster.

Step 4: Fully Connected Layers

The final layers in the CNN are fully connected i.e. all the neurons in the precious layer are connected to all the neurons in the next layer which results in a set up of high level function learning and is able to learn some complex functions.

(During Training) Step 5: Loss

The last layer of the CNN is the loss layer which calculates how far the prediction was of the CNN from the ground truth and hence directs the CNN to tune its weights accordingly so that the next time we encounter such input, our solution answer should be as close to ground truth as possible. This facilitates the training of the CNN.

3.2 Topic Modeling

Definitions:

- **C:** collection of documents containing N texts.
- **V:** vocabulary (the set of unique words in the collection)

Dimensionality Reduction

Topic modeling is a form of dimensionality reduction. Instead of representing the text in it feature's space like {words with their count in the text}, text is represented in its topic space where the words are clustered to form a topic and then these topics are the basis to represent the text document.

Unsupervised Learning

Topic Modeling can be easily compared to unsupervised learning because in it we basically form cluster of topics which are a cluster of words. As in clustering the number of clusters is a hyper parameter, in topic modeling, the number of topics is a hyper parameter. The whole text is represented as a mixture of topics where each one has a coefficient weight associated to it.

A Form of Tagging

Topic modeling is basically assigning many tags to a single text whereas document classification is assigning one tag to a text. Once all the topics have been formed, an expert can then label the topic looking at its word composition and hence give it a human readable tag.

3.3 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) [1] is a topic modeling algorithm. It is a generative statistical model that permits observation's sets to be presented by unobserved groups, presenting why few parts of the data are analogous. LDA makes sure that the document is represented as a collection of topics which is a collection of words in the document and thus is given importance in a topic by assigning some weight to it.

The LDA only requires the number of topics we want to be formed for a specific text and thus all it does is to rearrange the topics within the document and hence the words in the topics and obtain a decent combination of topic-word distribution.

A topic is just a collection of words which are given their importance in the topic by assigning some weight to it. Any human expert can thereby look at the word composition and come up with the topic name which is human readable.

Segregation topics can be obtained using following factors:

1. The quality of text processing.
2. The variety of topics the text talks about.
3. The choice of topic modeling algorithm.
4. The number of topics fed to the algorithm.
5. The algorithms tuning parameters.

3.4 Cosine Similarity

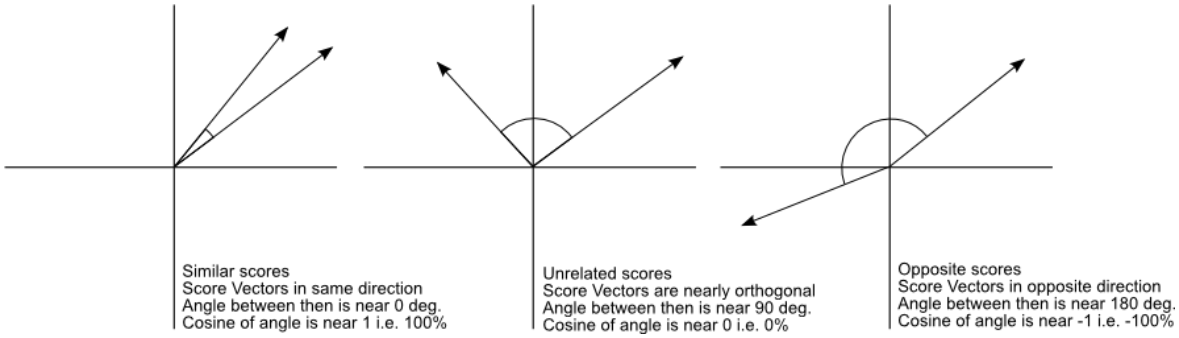
The cosine similarity [12] between two documents/texts is a measure of the cosine angle between their vectors in the Vector Space. It gives us the idea of orientation of the vectors and does not depend on the magnitude of the vector, it can be seen as a comparison between documents on a

Vector Space that their orientation is what is considered irrespective of their magnitude. In order to get the cosine similarity we need to solve this equation as under:

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos\theta$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

Cosine Similarity gives us the result that shows how two documents are related on the basis of angle instead of magnitude, in the Vector Space, like in image below



The Cosine Similarity values for different documents: 1 (same direction), 0 (90 deg.), -1 (opposite directions).

3.5 Histogram Values

All the accelerometer values for a single activity are used to generate statistical feature. The lowest and the highest values from the data are set as the end points and in that range; the intervals are divided to be 10, equally spaced. Once we get 10 intervals, the rest of the values are put in appropriate interval. Once all the data points are filled in to respective intervals, the count of each interval is then used as a statistical feature to be used in the CNN.

3.6 Sensors

1. **Accelerometer –** Accelerometer is widely used nowadays in wearable devices or even in our smart phones. It is an axis based motion sensing sensor which is responsible for counting our footsteps through our mobile even though we haven't purchased any separate wearable for it.

They are the reason how we know where our phone is pointing while we are navigating through Google maps etc. Hence it has been extensively used in the field of augmented reality nowadays.

As the name suggests, accelerometer measures the acceleration of the user in 3 dimensions and hence can be used in various kinds of apps for developing some cool features.

The sensor is composed of other sensors, inclusive of microscopic crystal structures. These become stressed when they experience acclerative force. The accelerometer then interprets the voltage coming from the crystals to find out speed and direction of the phone.

Accelerometer has wide range of use cases, ranging from switching between landscape and portrait on mobile or getting your current speed in a moving vehicle, it is one of the most important sensor contained in a smartphone.

2. **Gyroscope** – The gyroscope is the sensor which is responsible for assisting the accelerometer as to which way the phone is oriented.

   While playing car games, we tilt out phones in order to control the steer then those small tilts are detected by the gyroscope as accelerometer can only interpret changes or movements in the space.

   Gyroscopes find their place in aircrafts as well to determine the altitudes and positions of the plane in space.

   The gyroscopes are MEMS (Micro-Electro-Mechanical Systems) gyroscopes, a smaller variant of the concept embedded on an electronics board..

3.7 Activity Recognition

Activity Recognition [3] is to recognize what the user is doing based on his data collected over time in a particular environment. Suppose we closely examine an elderly man and try to recognize his daily routine through activity recognition. We can observe that the man wakes up at dawn, and then he ignites the stove to prepare coffee, turns on the geezer to bath, turns on the microwave and picks bread and jam from shelf. When he does his breakfast and morning exercise, he is reminded to switch off the geezer. This information can be used by any close

relative of the man to see that his father is eating properly or exercising daily or not. This is one of the use cases of Activity Recognition that can be very useful.

Activity Recognitions are:

- Sensor-based, single-user activity recognition
- Levels of sensor-based activity recognition
- Sensor-based, multi-user activity recognition
- Sensor-based group activity recognition

Chapter 4: Proposed Approach

4.1 System Architecture of Friend Recommendation

The Friend Recommendation system architecture follows client-server model. Where a phone used by a user is a client, while data centers are servers, as shown in the proposed system (Figure2).

On client side, every user's data (Accelerometer and Gyroscope sensor data) is recorded by his smart phone.

In this approach, activity recognition is done by using Artificial Neural Network namely Convolutional Neural Network [7].To develop good results, the training period of the model is set to be 1000 epochs which takes around 20 hours to train on a CPU with an i-7 core processor with 8 GB RAM. The running time of the test cases is not so large. In fact the model is designed such light weight so that the classification task for the test data can be done in real time.

Once the activities of the users are classified using our CNN model then the topic modeling algorithm comes into play. LDA [1] (Latent Dirichlet Allocation) algorithm is used to extract the life-style vectors of individual users.

On the server side, the task of friend recommendation is fulfilled by the design of the Modules. Collection of data form smart phones& creation of life documents of user is done by data collection module. By the help of probabilistic topic model, user's life styles are defined with the help of life style analysis module.

The friendship-matching module/script constructs friends-matching graph [12] and also friendship matching visual-records are constructed to represent the likeness relationship among different user life styles. Then, the ranking of the user's friend is done on the basis of a friend-matching graph. The user query module is used to intake user's request and present a ranked list of best friends as response.

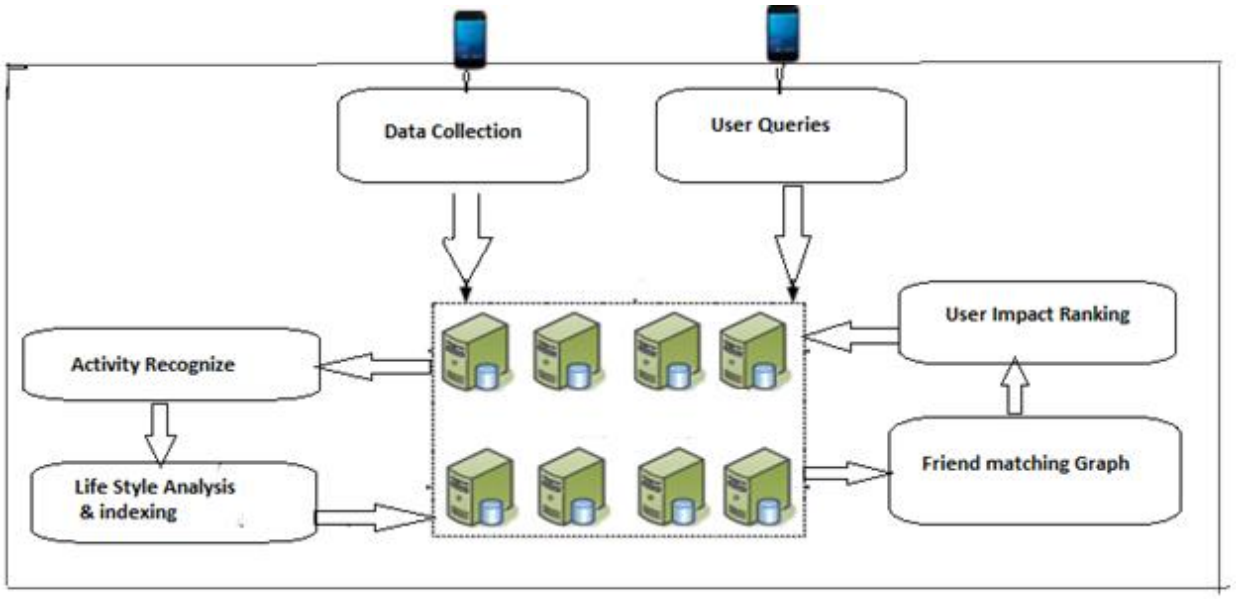


Fig 6: System Architecture of Friend Recommendation

4.2 Topic Model for Lifestyle Extraction

4.2.1 Lifestyle Model

In NLP, the probabilistic topic models can give us the probability of any particular topic in a document. In our approach we exploit this feature of probabilistic topic models to extract the life-style (analogous to topics) probabilities for a particular user (analogous to a document).

In probabilistic models [1], the frequency of different words is particularly important as this gives us an idea about the importance of the word or denotes their information entropy variance. Following this observation, "bag-of-activity" (Fig 3) model is constructed using the raw data with their probability distributions and replaces the original sequences of activities recognized. Thereafter, each user has a life document which is a 'bag-of-activity' representation, consisting of a mixture of activity words.

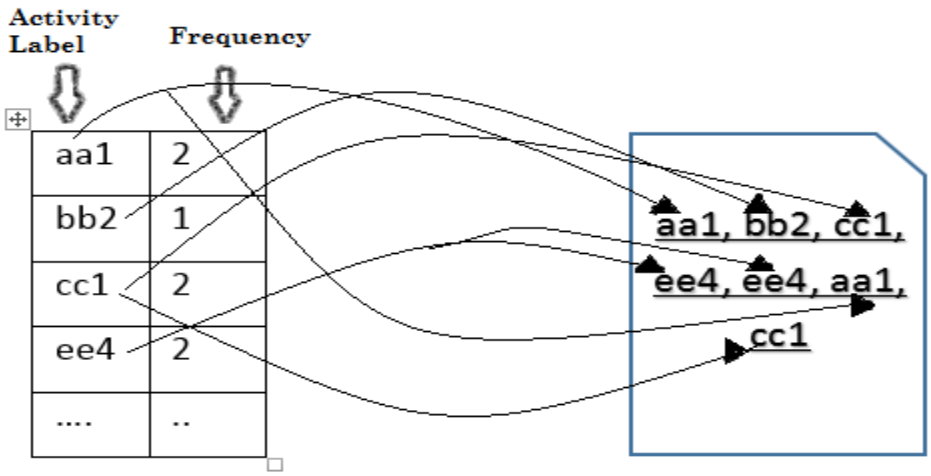


Fig 7: Pouch-of Activity Modeling for Life Document

Let w = [w_1,w_2 ...w_n] denote the set of activities, in which w_i represent i^{th} activity and W represent the total activities.

Let z = [z_1, z_2 ... z_n] represent a life styles set, in which z_i represent i^{th} life style and Z represent total life styles.

Let d = [d_1, d_2 ...d_n] represent a life documents set, in which d_i represent the i^{th} life document and total users is n.

Let $p(w_i|d_k)$ represent the probability measurement of the activity w_i in a particular living-life document d_k, $p(w_i|z_j)$ represent the probability of the degree the activity w_i adds to the life style z_j, and $p(z_j|d_k)$ represent the probability measurement of the living-life style z_j enrolled in the living-life document d_k.

$$p(w_i|d_k) = \sum_{j=1}^{Z} p(w_i|z_j)p(z_j|d_k)|$$

"bucket-of-activity" presentation for the life document d_k, $p(w_i|d_k)$ can be easily calculated as given by equation:

$$p(w_i|d_k) = \frac{f_k(w_i)}{\sum_{i=1}^{W} f_k(w_i)}$$

Where $f_k(w_i)$ denotes the frequency of w_i in d_k.

The life style vector of a user is

$$L_k = [p(z_1|d_k), p(z_2|d_k), \ldots p(z_Z|d_k)]$$

To find out the life style vector of every user by using the user's life documents is the objective. $p(w_i|d_k)$ can be calculated easily, but $p(w_i|z_j)$ and $p(z_j|d_k)$ are hard to compute due to the hidden characteristic of life styles.

$p(w_i|d_k)$, calculated by using Activity recognition, and but $p(w_i|z_j)$ and $p(z_j|d_k)$ are computed by algorithm LDA [1] decomposition algorithm.

4.2.2 Activity Recognition

The dataset that we have was a collection of real numbered values collected using accelerometer and gyroscope. The reading from them is taken and after pre-processing of the raw data, it is decided that any activity can be certain to happen in a period of 2.56 seconds. Therefore one training example of our dataset consisted of 128 real numbered values of one axes of a sensor. As we have two sensors and 3 axes per sensor so our one training example consists of 128*6 = 768 real numbers.

Now the first objective that our machine learning model must accomplish is to extract some special features as well as temporal features quite intelligently. The standard neural networks are quite good at differentiating classes by making complex nonlinear functions but are not very useful when it comes to identification of special or temporal features.

Convolutional Neural [9] are very good at this task. Commonly used in the area of computer vision, CNN has transformed the area of image classification and NLP because of its ability to extract features from the data automatically. Several models with different hyper parameters

were tried out in order to achieve the best accuracy. The models used are as follows which improved gradually.

Implementation Details:

Framework Used – Pytorch
Language Used – Python

Dataset Used : http://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions#

Matab scripts are used to create the dataset that can be used in the models. Features dataset is also calculated in this process.

Model 1: Used 1D convolutional layers followed by 1D max polling layers and one fully connected layer. The data is not preprocessed at all. (No Data Centering and No Data Normalization). Zero padding is applied.

Architecture:

Input-> Conv1D layer -> MaxPool1D -> FC1 -> Softmax layer

- Input – (Number of rows in dataset, 128*6) #rows = 7409
- $Conv1D^1$ = (in_channels = 6 ,out_channels = 196 , kernel_size = 16, stride = 1)
- $MaxPool1D^2$ – Size of 4
- fc1 = (196*28 , 6)
- Softmax layer – 6 classes

Hyper Parameters:

number_of_epochs = 500

batch_size = 100

learning_rate = 0.001

segment_size = 128

Activation Function – ReLU

CrossEntropyLoss - loss function

Adam Optimizer with L2 norm – 0.0005 and default beta1 and beta2

Accuracy Achieved:

Training Accuracy 96.42395664822`56

Testing Accuracy 89.35289878990834

Model 2: Used 1D convolutional layer followed by 1D max polling layer and two fully connected layers. The data is not preprocessed at all. (No Data Centering and No Data Normalization). Zero padding is applied.

Architecture:

Input-> Conv1D layer -> MaxPool1D -> FC1 -> FC2 -> Softmax layer

- Input – (Number of rows in dataset, 128*6) #rows = 7409
- $Conv1D^1$ = (in_channels = 6 ,out_channels = 196 , kernel_size = 16, stride = 1)
- $MaxPool1D^1$ – Size of 4
- fc1 = (196*28 , 1024)
- fc2 = (1024 , 6)
- Softmax layer – 6 classes

Hyper Parameters:

number_of_epochs = 500

batch_size = 200

learning_rate = 0.001

segment_size = 128

Activation Function – ReLU

CrossEntropyLoss - loss function

Adam Optimizer with L2 norm – 0.0005 and default beta1 and beta2

Accuracy Achieved:

Training Accuracy 99.12446664866325

Testing Accuracy 93.46713429736783

Model 3: Used 1D convolutional layer followed by 1D max polling layer and two fully connected layers. Data is normalized before feeding into the CNN. Zero padding is applied.

Architecture:

Input-> Conv1D layer -> MaxPool1D -> FC1 -> FC2 -> Softmax layer

- Input – (Number of rows in dataset, 128*6) #rows = 7409
- $Conv1D^1$ = (in_channels = 6 ,out_channels = 196 , kernel_size = 16, stride = 1)
- $MaxPool1D^1$ – Size of 4
- fc1 = (196*28 , 1024)
- fc2 = (1024 , 6)
- Softmax layer – 6 classes

Hyper Parameters:

number_of_epochs = 500

batch_size = 200

learning_rate = 0.001

segment_size = 128

Activation Function – ReLU

CrossEntropyLoss - loss function

Adam Optimizer with L2 norm – 0.0005 and default beta1 and beta2

Accuracy Achieved:

Training Accuracy 98.56982414866634

Testing Accuracy 90.32587566654322

Model 4: Used 1D convolutional layer followed by 1D max polling layer and two fully connected layers. Data centering is done before feeding into the CNN. Zero padding is applied.

Architecture:

Input-> Conv1D layer -> MaxPool1D -> FC1 -> FC2 -> Softmax layer

- Input – (Number of rows in dataset, 128*6) #rows = 7409
- $Conv1D^1$ = (in_channels = 6 ,out_channels = 196 , kernel_size = 16, stride = 1)
- $MaxPool1D^1$ – Size of 4
- fc1 = (196*28 , 1024)
- fc2 = (1024 , 6)
- Softmax layer – 6 classes

Hyper Parameters:

number_of_epochs = 500

batch_size = 200

learning_rate = 0.001

segment_size = 128

Activation Function – ReLU

CrossEntropyLoss - loss function

Adam Optimizer with L2 norm – 0.0005 and default beta1 and beta2

Accuracy Achieved:

Training Accuracy 99.67825518826368

Testing Accuracy 93.96886725143525

Model 5: Used two 1D convolutional layers followed by 1D max polling layers and two fully connected layers. The data is not preprocessed at all. (No Data Centering and No Data Normalization). Zero padding is applied. Dropout is applied on fc1.

Architecture:

Input-> Conv1D layer -> MaxPool1D -> Conv1D layer -> MaxPool1D -> FC1 -> FC2 -> Softmax layer

- Input – (Number of rows in dataset, 128*6) #rows = 7409
- $Conv1D^1$ = (in_channels = 6 ,out_channels = 92 , kernel_size = 20, stride = 1)
- $MaxPool1D^1$ – Size of 2
- $Conv1D^2$ = (in_channels = 92 ,out_channels = 256 , kernel_size = 12, stride = 1)
- $MaxPool1D^2$ – Size of 2
- fc1 = (256*21 , 1560)
- fc2 = (1560 , 6)
- Softmax layer – 6 classes

Hyper Parameters:

number_of_epochs = 500

batch_size = 200

learning_rate = 0.001

segment_size = 128

Dropout – 0.05

Activation Function – ReLU

CrossEntropyLoss - loss function

Adam Optimizer with L2 norm – 0.0005 and default beta1 and beta2

Accuracy Achieved:

Training Accuracy 99.82446664866325

Testing Accuracy 94.92148346140996

Model 6: Used two 1D convolutional layers followed by 1D max polling layers and two fully connected layers. Data Centering is done. Zero padding is applied. Dropout is applied on fc1.

Architecture:

Input-> Conv1D layer -> MaxPool1D -> Conv1D layer -> MaxPool1D -> FC1 -> FC2 -> Softmax layer

- Input – (Number of rows in dataset, 128*6) #rows = 7409
- $Conv1D^1$ = (in_channels = 6 ,out_channels = 92 , kernel_size = 20, stride = 1)
- $MaxPool1D^1$ – Size of 2
- $Conv1D^2$ = (in_channels = 92 ,out_channels = 256 , kernel_size = 12, stride = 1)
- $MaxPool1D^2$ – Size of 2
- fc1 = (256*21 , 1560)
- fc2 = (1560 , 6)
- Softmax layer – 6 classes

Hyper Parameters:

number_of_epochs = 500

batch_size = 200

learning_rate = 0.001

segment_size = 128

Dropout – 0.05

Activation Function – ReLU

CrossEntropyLoss - loss function

Adam Optimizer with L2 norm – 0.0005 and default beta1 and beta2

Accuracy Achieved:

Training Accuracy 99.59736672886436

Testing Accuracy 94.14563261403423

Model 7: Changed the architecture inspired by the research paper. https://www.sciencedirect.com/science/article/pii/S1568494617305665.

Data Centering applied to the dataset. Weights are initialized with almost standard normal distribution with mean 0 and standard deviation 0.01. Zero padding

Architecture:

Input -> Conv1D layer -> MaxPool1D -> FC1 ->FC2 -> Softmax layer

- Input – (Number of rows in dataset, 128*6) #rows = 7409
- $Conv1D^1$ = (in_channels = 6 ,out_channels = 196 , kernel_size = 16, stride = 1)
- $MaxPool1D^1$ – Size of 4
- fc1 = (196*28 , 1024)
- fc2 = (1024 , 6)
- Softmax layer – 6 classes

Weights in the model initialized with standard deviation of 0.01

Hyper Parameters:

number_of_epochs = 1000

batch_size = 200

learning_rate = 0.0005

segment_size = 128

Dropout – 0.05

Activation Function – ReLU

CrossEntropyLoss loss function

Adam Optimizer with L2 norm – 0.0005 and default beta1 and beta2

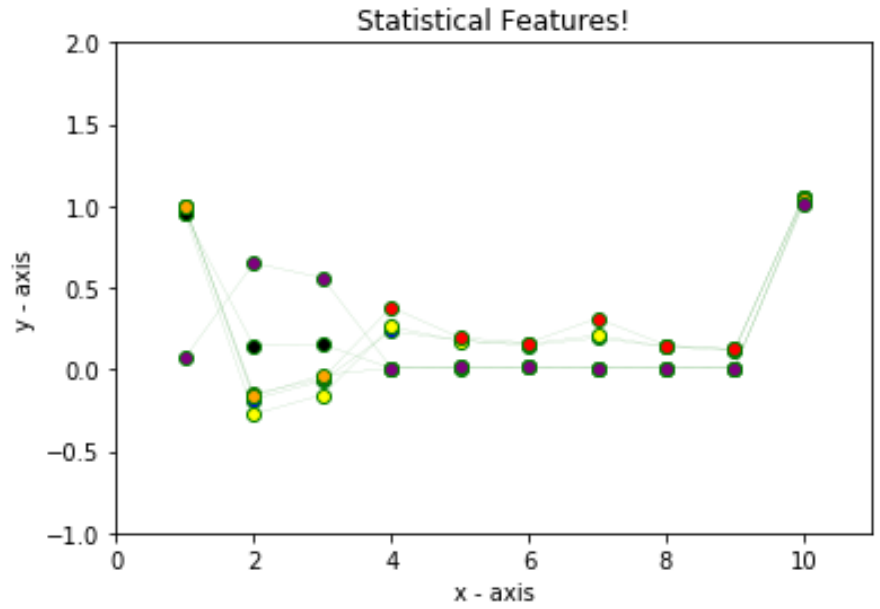
Accuracy Achieved:

Training Accuracy 99.94598973805023

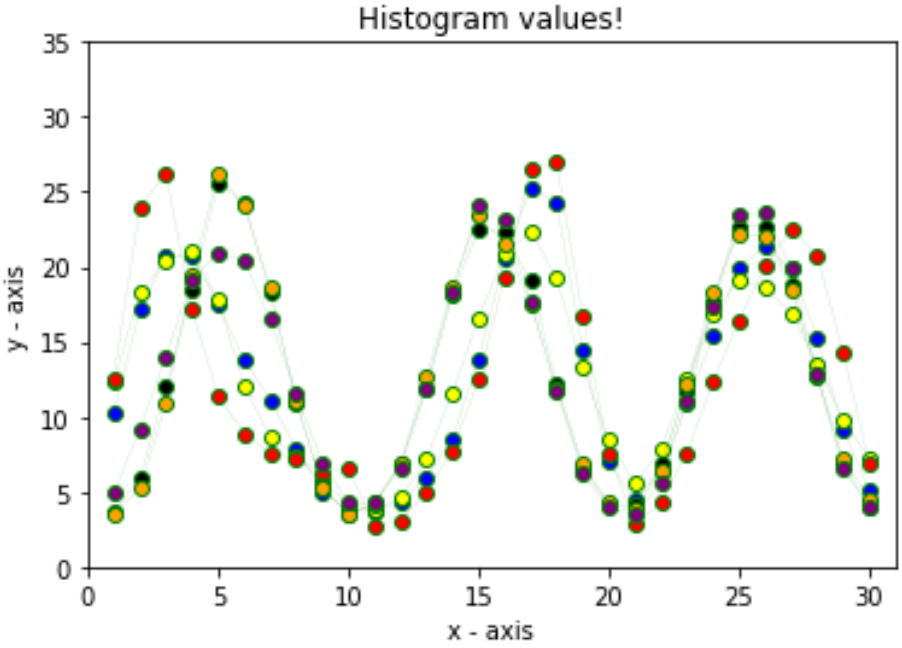
Testing Accuracy 95.35583027063147

After some data analysis using graph plots, 30 histogram values and 10 statistical values were added:

1. **mean(x)**
2. **mean(y)**
3. **mean(z)**
4. **std(x)**
5. **std(y)**
6. **std(z)**
7. **mean(abs(x - mean(x)))**
8. **mean(abs(y - mean(y)))**
9. **mean(abs(z - mean(z)))**
10. **mean(sqrt(x.^2 + y.^2 + z.^2))**
11. **hist(x, 10)/n**
12. **hist(y, 10)/n**
13. **hist(z, 10)/n**



1-10 Features of 6 different classes



11-40 features of 6 different classes

Model 8: So far the results were not as what were expected. So 40 statistical features were added in the FC1 layer of the previous model and padding was introduced in the convolutional layers because the data was getting lost a lot after convolutional layers and pooling layers and thus results were observed.

The 2D convolutional and 2D max pooling layers were used in this model. Some Padding, dropout and weight initialization is applied. Data Centering applied.

Architecture:

Input -> Conv2D layer -> MaxPool2D -> FC1 (+ stat features) -> FC2 -> Softmax layer

- Input – (Number of rows in dataset, 128*6) #rows = 7409
- $Conv2D^1$ = (in_channels = 1 ,out_channels = 196 , kernel_size = (16,6), stride = 1,padding = (8,3))
- $MaxPool2D^1$ – Size of (1,4) with stride (1,4)
- fc1 = (196*129 + 40 , 1024)
- fc2 = (1024 , 6)
- Softmax layer – 6 classes

Reshaped the data to (7409, 1, 128, 6) making 1 input channel and (128*6) 2D input

Hyper Parameters:

number_of_epochs = 1000

batch_size = 200

learning_rate = 0.0005

segment_size = 128

Dropout – 0.05

Activation Function – ReLU

CrossEntropyLoss loss function

Adam Optimizer with L2 norm – 0.0005 and default beta1 and beta2

Accuracy Achieved:

Training Accuracy 99.97299486902511

Testing Accuracy 94.85466087537587

Model 9:

Max Pool 2D is changed to 1, which is effectively no pooling applied.

Architecture:

Input -> Conv2D layer -> MaxPool2D -> FC1(+ stat features) -> FC2 -> Softmax layer

- Input – (Number of rows in dataset, 128*6) #rows = 7409
- Reshaped the data to (7409,1,128,6) making 1 input channel and (128*6) 2D input
- Conv2D = (in_channels = 1 ,out_channels = 196 , kernel_size = (16,6) , stride = 1, padding = (8,3))
- MaxPool2D – (1,1)
- fc1 = (196*129*7 + 40, 1024)
- fc2 = (1024 , 6)
- Softmax layer – 6 classes

Hyper Parameters:

number_of_epochs = 1000

batch_size = 200

learning_rate = 0.0005

segment_size = 128

Dropout – 0.05

Activation Function – ReLU

CrossEntropyLoss loss function

Adam Optimizer with L2 norm – 0.001 and default beta1 and beta2

Accuracy Achieved:

Training Accuracy 99.31136916014043

Testing Accuracy 96.19111259605747

Model 10: The padding is tuned a bit to achieve better results.

Architecture:

Input -> Conv2D layer -> MaxPool2D -> FC1 (+ stat features) -> FC2 -> Softmax layer

- Input – (Number of rows in dataset, 128*6) #rows = 7409
- Reshaped the data to (7409,1,128,6) making 1 input channel and (128*6) 2D input
- Conv2D = (in_channels = 1 ,out_channels = 196 , kernel_size = (16,6) , stride = 1, padding = (7,2))
- MaxPool2D – (1, 1)
- fc1 = (196*129*4 + 40, 1024)
- fc2 = (1024,6)
- Softmax layer – 6 classes.

Hyper Parameters:

number_of_epochs = 1000

batch_size = 200

learning_rate = 0.0005

segment_size = 128

Dropout – 0.05

Activation Function – ReLU

CrossEntropyLoss loss function

Adam Optimizer with L2 norm – 0.001 and default beta1 and beta2

Accuracy Achieved:

Training Accuracy 99.93248717256279

Testing Accuracy 96.49181423321083

	Architecture	*Training Acc(%)*	*Testing Acc(%)*
Model #1	Input-> Conv1D layer -> MaxPool1D -> FC1 -> Softmax layer No data preprocessing, Zero padding	**96.4**	**89.3**
Model #2	Input-> Conv1D layer -> MaxPool1D -> FC1 -> FC2 -> Softmax layer No data preprocessing, Zero padding	**99.1**	**93.4**
Model #3	Input-> Conv1D layer -> MaxPool1D -> FC1 -> FC2 -> Softmax layer Data Normalization, Zero padding	**98.5**	**90.3**
Model #4	Input-> Conv1D layer -> MaxPool1D -> FC1 -> FC2 -> Softmax layer Data Centering, Zero padding	**99.6**	**94.0**
Model #5	Input-> Conv1D layer -> MaxPool1D -> Conv1D layer -> MaxPool1D -> FC1 -> FC2 -> Softmax layer No data preprocessing, Zero padding, Dropout applied.	**99.8**	**94.9**
Model #6	Input-> Conv1D layer -> MaxPool1D -> Conv1D layer -> MaxPool1D -> FC1 -> FC2 -> Softmax layer Data Centering, Zero padding, Dropout applied.	**99.5**	**94.1**
Model #7	Input -> Conv1D layer -> MaxPool1D -> FC1 ->FC2 -> Softmax layer Data centering, Zero padding, Dropout applied.	**99.9**	**95.3**
Model #8	Input -> Conv2D layer -> MaxPool2D -> FC1 (+ stat features) -> FC2 -> Softmax layer Data Centering, Padding applied, Dropout applied	**99.9**	**94.8**
Model #9	Input -> Conv2D layer -> MaxPool2D -> FC1(+ stat features) -> FC2 -> Softmax layer Data centering, Padding applied, Dropout applied	**99.3**	**96.2**
Model #10	Input -> Conv2D layer -> MaxPool2D -> FC1 (+ stat features) -> FC2 -> Softmax layer Data Centering, Padding applied, Dropout applied.	**99.9**	**96.5**

4.2.3 LDA for Lifestyle Extraction

Latent Dirichlet Allocation is basically a matrix decomposition problem.

If we have the life documents of all users then the above discussed equation for LDA can be formulated as a matrix decomposition problem

$$p(w|d) = p(w|z)p(z|d)$$

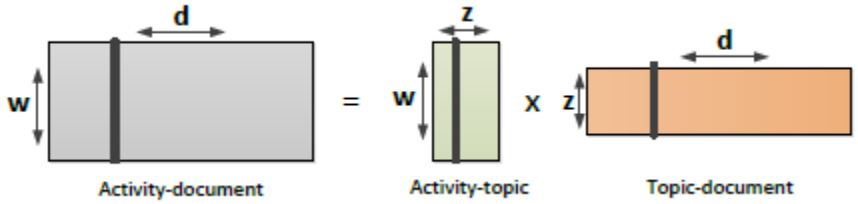


Fig 8: Matrix Decomposition for Life Styles Analysis

Here $p(w|d) = [p(w|d_1), p(w|d_2), \ldots p(w|d_n)]$ which is the probability of a particular activity in different documents, is calculated easily once we have accurate activity prediction model, model 5 can be used to give very promising results. $P(w|d_k) = [p(w_1|d_k), p(w_2|d_k), \ldots p(w_W|d_k)]^T$ gives the k^{th} position of column in the matrix of human's action document presenting the probabilities measurement of activities across the life document d_k of user k. $P(wj|z) = [p(wj|z_1, p(wj|z_2), \ldots p(wj|z_Z)]$ is the activity-topic matrix which represents the probability of each activity over each life style (topic),and $p(w_j|z_k) = [p(w_{1j}|z_k), p(w_{2j}z_k), \ldots p(w_wj|z_k)]T$ is the kth column in the activity-topic matrix representing the probabilities of activities over the life style z_k. $P(zj|d) = [p(z_j|d_1); p(z_j|d_2), \ldots p(z_j|d_n)]$ is the topic-document matrix containing the probability of each topic over each life document, and $p(z_j|d_k) = [p(z_{1j}|d_k), p(z_{2j}|d_k), \ldots p(z_{Zj}|d_k)]T$ is the kth column in the topic-document matrix representing the probabilities of life styles over the life document d_k of user k.

The main objective of us is to get the vector p(z|d). LDA does that work for us and hence we are able to compare user's lifestyles.

```
import numpy as np
import lda
import matplotlib.pyplot as plt
import collections
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.datasets import make_multilabel_classification
from sklearn.metrics.pairwise import cosine_similarity
from numpy.linalg import norm

user_bow = np.load('life_document.npy')
user_bow = user_bow.tolist()

res = []

for i in user_bow:
    r = collections.Counter(i)
    temp = []
    for j in range(0,6):
        temp.append(r[j])
    res.append(temp)

#print(res)

ldamodel = LatentDirichletAllocation(n_components=25,random_state=0,max_iter = 500)
ldamodel.fit(res)
result = np.array(ldamodel.transform(res))
print(result.shape)

```

Code Snippet for LDA

How LDA works internally?

Latent Dirichlet Allocation imagines every document as a distribution of topics in the particular dataset, like {Topic 1: 0.6, Topic 2: 0.0, and Topic 3: 0.1, Topic 4: 0.3}. Now each topic is a cluster of words like {Minister: 0.6, speech: 0.3, cities: 0.1}. In order to make a document from the distribution of topics, we first have to choose a topic according to the probabilities. So, intuitively we would be choosing Topic 1 instead of Topic 4 and least probably Topic 2. Once the topic is chosen, we have to choose a word from that topic, so again we would be choosing Minister instead of speech. Now this gives us our first word in our document. This process is repeated again and again for how long we need our document to be.

4.3 Friend Matching Graph

Friend Matching graph [3] is very helpful at the time of queries of the user. We can easily find closely related friends for a user once we have the friend-matching graph constructed. A node of the graph denotes users and the link/edge between them denotes that there is a connection between them. The more the weight of the edge, the more is the similarity of both the users. Also

when it comes to find out the top 5 friends or so, the graph data structure can be used as an efficient data structure to get fast results.

	V1	V2	V3	V4	V5	V6	V7
V1		0.87	0.99	0.43	0.67	0.43	0.87
V2	0.87		0.53	0.99	0.67	0.53	
V3	0.99	0.53		0.43			
V4	0.43	0.99	0.43		0.76		
V5	0.67	0.67		0.76		0.99	
V6	0.43	0.53			0.99		0.67
V7	0.87					0.67	

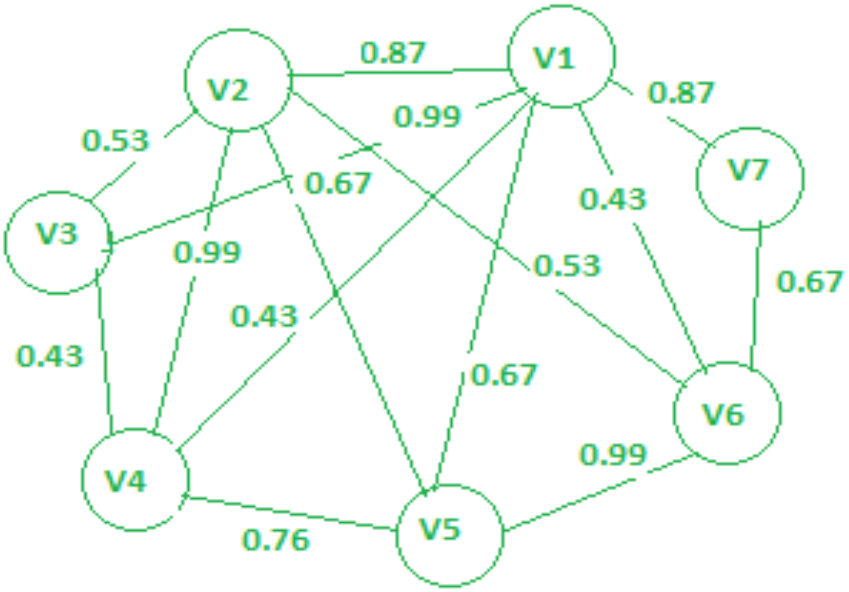


Fig 9: A Representation of Friend-Matching Graph

4.3.1 Similarity of Metric

Similarity metric is provided by the similarity between two life style vectors [12]. Let L_i = $[p(z_1|d_i); p(z_2|d_i); :::; p(z_Z|d_i)]$ and L_j = $[p(z_1|d_j); p(z_2|d_j); :::; p(z_Z|d_j)]$ represent the life style vectors of user we user i and user j.

Similarity between users is not only dependent upon their simply life style vector but it also depends on the dominating activities in the life style vector because the dominating values will affect the lifestyle a lot. .Hence, the likeness of living-life actions among person I and person j, represented by Sim(i; j), is defined as following:

$$Sim(i; j) = Sim_c(i;j) * Sim_d(i;j)$$

Where $Sim_c(i;j)$ is computing the likeness of the combined life style vectors of users, $Sim_d(i;j)$ is used to emphasize the likeness of users on their dominant life styles. Follow the cosine similarity metric for $Sim_c(i;j)$, which is,

$$Sim_c(i; j) = \cos(L_i;L_j)$$

To calculate $Sim_d(i; j)$, initially define the set of user's possessive life styles[12].

We define possessive life styles as a set where the probability distribution must be greater than or equal to an already defined value [lambda]

Using these conditions, there is an assurance that living activities to be included in the set of living-life styles are with larger probabilities. We sorted the life style vector L_i in the decreasing order in respect to the probabilities of life styles to find D_i.

"The similarity metric $Sim_d(i; j)$ for computing the likeness of the dominant living life-style sets among two users is denoted as given in equation below

$$Sim_d(i, j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|}$$

The range of $Sim_d(i; j)$ is [0; 1].

4.4 Query and Friend Recommendation

The server will do infusion the human living life style vector on receiving a human's request and on its basis will suggest likeness friends for better friendship to the user. The response results are filtered with best matching living life style.

Our recommendation mechanism is to recommend likely friends to a user query. Server takes use of the friend matching graph for extracting the top n best possible friends of a user. The best result can be an immediate neighbor of user or neighbor of the friend of user.

Chapter 5: Resuts and Analysis

5.1 Data Collection and Preprocessing Details

The raw data of the accelerometer and gyroscope are collected using the data collected from 30 different users over a period of 2 months. The data is time dependent and the respective user activity is noted which is used later for treating the problem as a supervised learning problem.
This raw data is first used to divide the data chunks into interval of 2.56 seconds time. The claim we have made is that 2.56 seconds is sufficient to determine a particular activity. The data collected is recorded at 50 hertz. So in 2.56 seconds, 128 entries are generated of each axis of each sensor for a particular activity. So for one training example, we have 128*6 = 768 real values as input.
Firstly we designed a very accurate Activity Recognition model using Convolutional Neural Network. The activities are as follows:

1. Activity of walking
2. Activity of walking downstairs
3. Activity of walking upstairs
4. Activity of sitting
5. Activity of laying
6. Activity of standing

Several models were tried on to get the most accurate train and test results. It was seen that the data values when trained with simple multi-layer perceptron did not perform that much well but once we extract some low level features from the data, and then feed it to the fully connected multi-layer perceptron, it works quite well. As the dimension of our dataset is quite low so hyper parameters like padding and stride were extremely helpful once they are tuned appropriately.
One surprising thing to be noted is that Data Centralization works quite well than data normalization. Data normalization works the worst.

Data was then processed to understand the nature of the dataset. Several statistical features were extracted and plotted to see the actual difference between different classes which when incorporated in the model, increased the accuracy.

Total of 40 statistical features were extracted from the accelerometer data which are as follows:

1. **Mean(x) –** mean of x values
2. **Mean(y) –** mean of y values
3. **Mean(z) –** mean of z values
4. **std(x) –** standard deviation of x values
5. **std(y) –** standard deviation of y values
6. **std(z) –** standard deviation of z values
7. **Mean (abs(x – mean(x))) –** mean of absolute values of difference of x axis values from the mean of x axis values.
8. **Mean (abs(y – mean(y))) -** mean of absolute values of difference of y axis values from the mean of y axis values.
9. **Mean (abs (z - mean (z))) -** mean of absolute values of difference of z axis values from the mean of z axis values.
10. **Mean (sqrt (x. ^2 + y. ^2 + z. ^2))–**Root Mean Square Values of x, y and z axis values.
11. **Hist(x, 10)/n –** Histogram values after dividing the x axis data into 10 equal interval chunks.
12. **Hist(y, 10)/n -** Histogram values after dividing the y axis data into 10 equal interval chunks.
13. **Hist (z, 10)/n -** Histogram values after dividing the z axis data into 10 equal interval chunks.

DATA SUMMARY FOR OUR MODEL:

Total User	Total Activities	Extra Features	Total Data Records For Model	Training Data Size (71%)	Testing Data Size (29%)
30	6	40	10402	7409	2993

Table 1: Data Summary for model

5.2 Friend Recommendation Results

The figure below illustrates the image representation of the likeness matrix for 21 users. 30 users are divided as 21 and 9 accounting for 70% and 30% of training and testing data respectively. A block color represents recommendation score between two users. The brighter the color of the square block is, the more is the recommendation score is. So, white represents the best score and black represents the worst score. As there is no sense of recommending a user to itself so the score of that is set to be zero, therefore all the diagonal square blocks are black in color. In the figure, User 1 has higher matching living life style with user 6 and user 3 has higher matching living life style with user 15, therefore they are represented with white color, and user 11 and user 4 have no matching in lifestyle with each other, which is represented by black color.

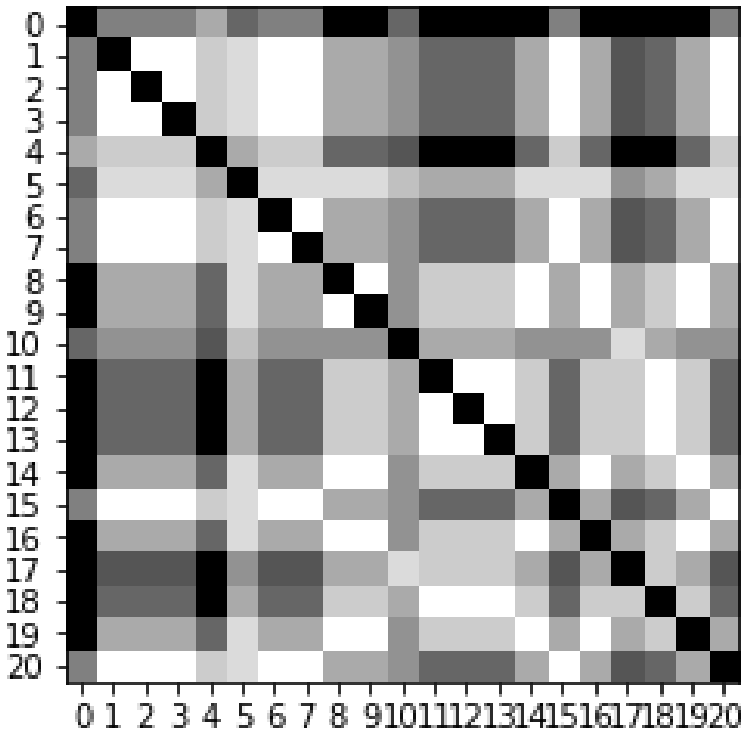


Fig 9: Friend recommendation results

5.3 Implication of Recommended Results

As we have seen so far that with the advent of technology and social network platforms, friend suggestions are mostly based on the social graphs. Greater the number of mutual friends you have with a person, higher the chances of getting the friend recommendation of that person. Such an approach does not reflect end user's preference for friend selection. Thus, our approach for friend suggestions which is based on the similarity of the lifestyle of different users fills the gap and proves to be very efficient at friend recommendations.

Whenever a user requests for friends, then our algorithm will return people's list sorted by recommendation score, from which user can choose whom to send request. The recommendation friends can be user's immediate friends or they can be friends of friends based on their scores.

In this manner, our algorithm proves to be very useful as a search tool which is based on machine learning model and therefore adapts itself with the different and dynamic types of data. It also gives very promising results to the user.

Chapter 6: Conclusion

6.1 Conclusive Results

In our project, we collected 30 user's sensor data mapping with 6 activities. Our models have improved gradually. The improvements have come based on careful and smart hyper parameter tuning, architecture selection and understanding of the data we are dealing with. The accuracy has grown from 80% to 96.5% if we include all the statistical parameters and using convolutional neural networks with appropriate hyper parameters.

After achieving such a good accuracy for activity recognition, we were able to make a friend recommendation system using the user's life style matching, which is obtained using LDA algorithm. LDA is the best known algorithm till now for topic modeling and has worked well for our use project. Once we get the user's lifestyle, user friend matching score is achieved by our unique similarity metric technique which takes into account both, the normal lifestyle of user as well as the dominant lifestyle of the user.

At the end, a friend matching graph is used to extract the best suitable friends for a user on his query.

Total User	Total Activities	Statistical Features	Accuracy Measure for model of Activity recognition
30	6	40	96.5%

Table 2: Data summary for activities recognition for model

6.2 Conclusive Summary

The proposal that we have presented contain design and implementation of Friend Recommendation System. This recommendation system is based on matching life-style which can be used to recommend friend in social networking applications. This approach is different from the friend recommendation mechanisms, which are currently using by different social networking applications rely on social graphs in current social networking technology. Friend Recommendation System extracts life styles from user-centric data obtained from sensors on the smart phone and recommend potential friends to users if they share similar life styles.

Implemented Friend Recommendation System can run on the Android-based smart phones, and can learn things on runtime. We have measured its performance on real small series of data. The results show that the recommendations accurately present the user prefernces in choosing friends.

6.3 Future Work

The future task would be to determine our proposed system on bigger-series field experiments. Also, we would add more devices for discovering more users' activities and also include statistical features along the gyroscope.

REFERENCES

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. Journal of Machine Learning Research, 2003, pp. 993-1022.

[2] Lawrence, Steve, et al. "Face recognition: A convolutional neural-network approach." *IEEE transactions on neural networks* 8.1, 1997, pp. 98-113.

[3] Ignatov, Andrey. "Real-time human activity recognition from accelerometer data using Convolutional Neural Networks." *Applied Soft Computing* 62, 2018, pp. 915-922.

[4] L. Bian and H. Holtzman. Online friend recommendation through personality matching and collaborative filtering. Proc. of UBI-COMM, 2011, pp. 230-235.

[5] J. Kwon and S. Kim. Friend recommendation method using physical and social context. International Journal of Computer Science and Network Security, 2010, pp. 116-120.

[6] X. Yu, A. Pan, L.-A. Tang, Z. Li, and J. Han. Geo-friends recommendation in gps-based cyber-physical social network. Proc. Of ASONAM, 2011, pp. 361-368.

[7] Ha, Sojeong, and Seungjin Choi. "Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors." *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 2016.

[8] L. Gou, F. You, J. Guo, L.Wu, and X. L. Zhang. Sfviz: Interestbased. friends exploration and recommendation in social networks. Proc. of VINCI, 2011, pp 15.

[9] Zeng, Ming, et al. "Convolutional neural networks for human activity recognition using mobile sensors." *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*. IEEE, 2014.

[10] Ronao, Charissa Ann, and Sung-Bae Cho. "Human activity recognition with smartphone sensors using deep learning neural networks." *Expert Systems with Applications* 59, 2016, pp. 235-244.

[11] Li, J. A. Stankovic, M. A. Hanson, A. T. Barth, J. Lach, and G. Zhou. Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information. Proc. of BSN, 2009, pp. 138-143.

[12] Wang, Zhibo, et al. "Friendbook: a semantic-based friend recommendation system for social networks." *IEEE transactions on mobile computing* 14.3, 2015, pp. 538-551.

[13] J. Biagioni, T. Gerlich, T. Merrifield, and J. Eriksson. EasyTracker: Automatic Transit Tracking, Mapping, and Arrival Time Prediction Using Smartphones. Proc. of SenSys, 2011 , pp. 68-81.

[14] D. M. Karantonis, M. R. Narayanan, M. Mathie, N. H. Lovell, and B. G. Celler, "Implementation of a real-time human movement classified using a tri-axial accelerometer for ambulatory monitoring," IEEE Trans. Inf. Technol. Biomed., vol. 10, no. 1, 2006, pp. 156–67.

[15] M. Mathie, B. Celler, N. Lovell, and A. Coster, "Classification of basic daily movements using a triaxial accelerometer," Med. Biol. Eng. Comput., vol. 42, 2004, pp. 679–687.

[16] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in Pervasive Computing, Berlin/Heidelberg, Germany: Springer-Verlag, 2004, pp. 158–175.

[17] N. Ravi,N.Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," in Proc. 20th Nat. Conf. Artif. Intell., 2005, pp. 1541–1546.

[18] U. Maurer, A. Smailagic, D. Siewiorek, and M. Deisher, "Activity recognition and monitoring using multiple sensors on different body positions," and monitoring using multiple sensors on different body positions," 2010, pp. 113–116.

[19] A. M. Khan,Y.K. Lee, and T.-S. Kim, "Accelerometer signal-based human activity recognition using augmented autoregressive model coefficients and artificial neural nets," in Proc. 30th Annu. IEEE Int. Conf. Eng. Med. Biol. Soc, 2008, pp. 5172–5175.