MWDB Project Phase 3 Report

Nithish Moudhgalya, Preethi Raman, Raghavendar Thiruvoipadi

Samyuktha Sridhar, Skanda Suresh, Trilok Kumar Tourani

Abstract

In this phase of the project, we aim to create a fully operative query retrieval and feedback system. The system would take user desired query and use a combination of new and old algorithms to compute the top m closest files to the query and return to the user. The user then has an option to give feedback to the system. The system uses the user feedback and optimizes itself or the query and retrieves a new set of results to show to the user. In this process, the system slowly adapts its results to user preference. We use 4 retrieval methods - K Nearest Neighbour(KNN), Personalized Page Rank (PPR) [1], Locality Sensitive Hashing (LSH) [2] and Probabilistic Retrieval methods [3], and 2 query optimization techniques namely, - Vector optimization and Probabilistic optimization [3]. Using a combination of these retrieval and optimization techniques, we create an intelligent system that learns user model and adapts its choices of retrieval and optimization accordingly. We also explored the possible advantages of different vector models and latent semantics that can be used to improve the results of classification using Decision Trees, KNN or PPR based classifiers in task2.

Keywords - gesture recognition, pattern analysis, personalized page rank, locality sensitive hashing, decision tree, k nearest neighbour, query retrieval, user feedback, query optimization

1 Introduction

Gesture recognition is an active research field which tries to integrate the gestural channel in Human Computer Interaction. It has applications in virtual environment control, but also in sign language translation, robot remote control or musical creation. Recognition of human gestures comes within the more general framework of pattern recognition. In this framework, systems consist of two processes: the representation and the decision processes. The representation process converts the raw numerical data into a form adapted to the decision process which then classifies the data. Gesture recognition systems inherit this structure and have two more processes: the acquisition process, which converts the physical gesture to numerical data, and the interpretation process, which gives the meaning of the symbol series coming from the decision process. In device-based systems, the acquisition of gestures is made by a physical device that directly measures some characteristics of the gesture, generally the different joint bending angles.

In this work, we use a dataset constructed from a mixture of data seen in the task ChaLearn Looking at People wherein the gestures are recorded by using 20 different sensors positioned at 20 different spots in the human body. Each file has a multivariate time series recorded across multiple sensors. The objective is to use this multivariate time series data to retrieve 10 closest resembling gestures to a query gesture file. The details of the proposed solution can be seen in the following section.

1.1 Terminology

1. **Term Frequency** Term Frequency (tf) measures the number of times a particular pattern or word occurs in a document as opposed to the total number of words in the document. In

our case it calculates the ratio of frequency of occurrence of a word to the total number of words in each sensor in each file in each component. Equation 1 shows the tf formula used in the phase.

$$tf(w, f, s, c) = \frac{F(w, file\_id = f\ sensor\_id = s,\ component = c)}{(file\_id = f\ sensor\_id = s, component = c)} \tag{1}$$

2. **Inverse Document Frequency** Inverse Document Frequency (idf) measures the number of documents a particular pattern or word occurs in as opposed to the total number of documents. In our case it calculates the log of ratio of total number of documents to number of documents the given word occurs in each sensor. Equation 2 shows the idf formula used in the phase.

$$idf(w, s, c) = log\left(\frac{num\_total\_files}{N_{\forall f \in files\ and\ \forall c \in components}(w)}\right) \tag{2}$$

3. **Quantization**: We have continuous time series data. We need to make this a series of words. To do this we need discretize the data, this is where quantization comes in. We Normalize and assign data represenatative values based on the gaussian band under which it occurs. This assignment is based on the formulation below. The representative quantized value assigned to the data point is based on the mid point of the length of its gaussian band.

$$length_i = \frac{\int_{(i-r-1)/r}^{(i-r)/r} Gaussian_{(\mu=0,\sigma=0.25)}(x)dx}{\int_{-1}^{1} Gaussian_{(\mu=0,\sigma=0.25)}(x)dx} \tag{3}$$

4. **Normalization**: Different sensors record data in different ranges. In order to ensure one sensor does not affect results more than others, we normalize the data using Min Max normalization.

$$MinMax(X) = \frac{X - X_{Min}}{X_{Max} - X_{Min}} \tag{4}$$

5. **Similarity Graph**: It is used to model the degree of similarity/association between different nodes in a graph. An edge between 2 nodes indicates an association and the weight of the edge is the magnitude of similarity. We use this graph to represent the Gesture - Gesture similarity matrix. There are different types of similarity graphs we can use as shown below:
    - **k-nearest neighbor graphs**: In this graph we only consider the *K* closest neighbours to a particular node as similar. However this could lead to the notion of an directed graph which is undesired.
6. **Adjacency Matrix**: We represent the similarity graph using a matrix *A*, known as its adjacency matrix. It is used to model all the node -node correspondences in a graph. Thus for a graph with *n* nodes, the adjacency matrix is of dimensions *n X n*.
7. **Dimensionality Reduction**: It is the transformation of data from high dimensional space to a low dimensional space. The low dimensional representation obtained retains meaningful properties of original data.
8. **Dot product**: It is the product of magnitude of the projections of the vectors. It helps measure both composition similarity and numeric similarity between different dimensions of the vectors.

1.2 Goal Description

The goal of this phase in the project is to explore and use vector and sequence based similarity measures to find most similar gestures from the database and cluster the gestures based on similarity scores. We can divide the goal into sub-goals for each task as follows,

- task1 - implementing personalized page rank based on [1]
- task2 - performing classification using decision tree, KNN, PPR algorithms
- task3 - implementing locality sensitive hashing and using it to retrieve top n similar gestures
- task4 - implementing probabilistic retrieval and feedback system
- task5 - implementing PPR based retrieval and feedback system
- task6 - implementing query based retrieval, feedback system with user options and interface

1.3 Assumptions

In this phase of the project, we make a few assumptions about the dataset, the algorithms we use and the cost functions.

2 Proposed solution

2.1 Task1

In this task, we implement a personalized page rank (PPR) algorithm based on [1]. In PPR, the dataset can be seen as a graph with each node representing a gesture. The edges in the graph are weighted based on the similarity value between the two gestures. This graph is called the similarity graph G_s.

Algorithm 1: Personalized Page Rank Algorithm

Result: Steady state probability for reaching every node, starting from node i
u_i = zero-vector of size N with 1 at i^{th} position
v_i = zero-vector of size N with 1/n at all n seed positions
c = restart probability, default $= 0.8$
while u_i *not converged* **do**
  $u_i = (1 - c) * A.u_i + c * v_i$

In our implementation, we first begin by using the similarity graph that we constructed in phase2 of this project - using different vector models and semantic models. Then, we create a k-nearest neighbour graph - G_k. This graph is constructed by keeping only the best k neighbours for every node, in other words, the number of edges leaving a node would always be k. After constructing the new k-nearest graph G_k, we then normalize the similarity matrix column-wise ensuring the values sum up to 1. The matrix hence constructed is the Adjacency matrix used in PPR equation, A. Seed nodes are a list of nodes that could act as restart points in the random walk expression. In this task, we get n seed nodes from the user. Now we implement the PPR algorithm shown in Algorithm 1.

The goal of this task is to find m dominant gestures given a set of seed files. Hence, we follow the procedure as seen above to first estimate the steady state probabilities of reaching every node in the graph starting from a set of seed nodes. This vector can also be seen as a measure of similarity between the nodes in the graph and the seed nodes. Then we choose the m highest probable nodes and return then as the dominant gestures to the user.

2.2 Task2

In this task, gesture classification is performed based on three algorithms mentioned in the following sections.

2.2.1 KNN

In this option, K-nearest neighbors algorithm is used to retrieve a set of top k dominant gestures or similar gestures to a given input gesture. The top k nearest neighbor to the input gesture is computed using either of the distance measures-Euclidean distance/Mahalanobis distance.In this implementation, the features extracted for the gesture files in task 1 of Phase2 is used as the dataset. The formula to compute Euclidean distance is given in equation 5

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{5}$$

p,q represents two two different files/gestures in our case. The formula to compute Mahalanobis distance is given in equation 6

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (p-q) S^{-1} (p-q)} \tag{6}$$

where S^{-1} represents the co-variance matrix of p.

2.2.2 PPR

We use two approaches to classify the unlabeled gestures using PPR algorithm. Both these methods are viable and fetch reasonable good results.

Method1
As seen from task1, PPR algorithm can be used to retrieve a set of dominant gestures or similar gestures. We use the PPR probability estimates to classify the files using a majority voting and weighted score based voting. In other words, given m dominant gestures for a given unlabelled gesture, we can associate a label to the file by choosing the majority label from the dominant gestures or associate the label that has maximum score across all dominant gestures.

Method2
In this method, we use smart PPR approach. As we know the labels of a few nodes in the graph, we first partition the similarity graph into n sub-graphs with each sub-graph having the set of nodes with same label. Now, for each sub-graph we perform a PPR algorithm keeping all the labelled nodes as seed nodes, and find the probability of reached all the unlabelled nodes from these seed nodes. Performing the above step for each sub-graph renders us with probabilities of reaching the unlabelled nodes from each sub-graph. Then, we associate a label to the unlabelled nodes based on which sub-graph has higher probability of reaching that node. This way, unlabelled nodes get labelled in a simple and faster way.

2.2.3 Decision Trees

In this option, decision tree based algorithm is used to classify the given gestures. In decision trees, the predictions are made by traversing the tree from root to leaf and at each node, we split the tree based on a threshold value. In our implementation, we have used the features extracted for the gesture files in task 1 of Phase2 as our dataset. We have used Classification and Regression Trees as our training algorithm. Gini impurity is used here to find how pure a node is. A node is said to be pure if all its samples belong to the same class. Gini impurity is defined by equation 7. In this equation, n is the training samples, k is the number of classes/labels and p_k is the fraction of samples belonging to class k.

$$G = 1 - \sum_{k=1}^{n} p_k^2 \tag{7}$$

The tree is split such that the gini impurity of each node is minimized. The CART algorithm is a recursive algorithm which stops when the maximum depth is reached or when no split can lead to children purer than their parent.

2.3 Task3

In this task, we implement a Locality Sensitive Hashing (LSH) algorithm for finding approximate nearest neighbors in high dimensions [2]. Contrary to normal hashing algorithms where the aim is to minimize collisions, the aim for LSH is to use hashes such that nearby objects always collide with the same hash value and farther objects have different hash values.

We calculate a hash value based on the data point's position to a random vector (0 for negative and 1 for positive). A simple dot product between the data point and random vector yields us a bucket number in which it falls (ie. 0 or 1). Usage of only one hash function can increase both false positives and misses. Therefore, we use k hash functions to give a k-bit string of 0s and 1s which is our bucket id. This makes the partition or hashing process more conservative. Being conservative, we can see a decrease in false positives but many misses of our nearest neighbors.

To handle the misses, we create L layers with k unique hash functions at each layer which helps us solve this problem.

When a query is given to the LSH tool, the tool will hash the query using $L * K$ hash functions and maintain the bucket hashmap at each layer. We take $numResults$ as an input from the user to

give the functionality of getting sorted results starting with the closest neighbors. If there are not enough results in the corresponding buckets from each layer, then we find a bucket with similar id by changing 1,2,3...k-1 bits until we find $numResults$ results. For displaying these nearest neighbor files to the user, we calculate cosine similarity of the query point with each candidate point to show sorted results to the user. The working of LSH is shown in algorithm 2.

Algorithm 2: LSH Algorithm

Result: Ranked list of retrieved files and their associated similarity scores
$L =$ No. of layers
$k =$ No. of hash functions
$q =$ Query file
$\eta =$ family of hash functions
$maxResults =$ Maximum no. of results to be retrieved
Training
- Choose L functions g_j where $j = 1..L$, set $g_j = (h_{1j}, h_{2j}, ...h_{k,j})$ where $h_{i,j} \in \eta$
- Compute L hash tables where for $j = 1..L$, j^{th} entry contains points hashed by g_j.

Querying
- For each $j \in (1..L)$:
    1. Retrieve all points $p_j = g_j(q)$ in the j^{th} table.
    2. For each p_j compute distance from q and verify if distance is acceptable.
    3. Update total points retrieved L'.
    4. Stop if $L' > maxResults$.

2.4 Task4

In this task, we implement a probabilistic relevance feedback system to improve nearest neighbor matches following the works of [3]. The system enables the users to label the top n results returned by the search task as relevant or irrelevant. The search task then returns a new set of results by revising the query. We have used the vector models (TF/TF-IDF) constructed in task0b of Phase2 as the document vectors. We then implement the probabilistic retrieval model which returns a new query vector by considering the probabilities of relevant and nonrelevant documents. The query document similarity value is found using the equation 8.

$$sim(D, Q) = \sum_{i=1}^{t} d_i log \frac{p_i(1 - u_i)}{u_i(1 - p_i)} \tag{8}$$

Here D is the document vectors and Q is the query vector. For initial search, the values of p_i are taken as constant for all the terms. The value of u_i is taken as n_i/N. n_i is the total number of retrieved items with term i. When the document relevance information is available, the values of p_i and u_i can be computer using the equations 9 and 10 respectively. Here, R is the total number of relevant retrieved items, and r_i is the total number of relevant retrieved items that include term i.

$$p_i^{'} = Pr(x_i = 1|rel) = \frac{r_i + n_i/N}{R + 1} \tag{9}$$

$$u_i^{'} = Pr(x_i = 1|nonrel) = \frac{n_i - r_i + n_i/N}{N - R + 1} \tag{10}$$

The optimized query vector is then found using the equation 11. The value for $p_i^{'}$ and $u_i^{'}$ in this equation can be obtained using 9 and 10.

$$Q_{new} = log[p_i^{'}(1 - u_i^{'})/u_i^{'}(1 - p_i^{'})] \tag{11}$$

An adjustment is further made to the query terms to enhance the importance of the document terms that also occur in the queries. This is done by assuming that a term occurred in a query is equivalent

to a term occurred in 3 relevant documents. The probabilistic adjusted derivations is revised by updating the query terms in equation 9 and equation 10 by using $r_i^{'}$=r_i+3 and $R_i^{'}$ = R_i+3 instead.

The importance of each sensor, file and the components are found by finding the difference between each of them in all iterations. Heatmaps are generated and visualized for the same for better interpretation of the importance of these sensors, files and components over time. The heatmaps can be found in the results section.

2.5 Task5

2.5.1 Vector Processing Methods

Given document vectors D and query vector Q, the similarity between the corresponding vectors can be found using the equation 12. This similarity measure between the document and the query vectors helps in retrieving relevant items from the collection of documents. The query vector can then be optimized using the equation 13.

$$Sim(D,Q) = \sum_{i=1}^{t} d_i \bullet q_i \tag{12}$$

$$Q_{opt} = \frac{1}{n} \sum_{relevant\_items} \frac{D_i}{|D_i|} - \frac{1}{N-n} \sum_{nonrelevant\_items} \frac{D_i}{|D_i|} \tag{13}$$

D_i is the document vector and $|D_i|$ is the euclidean vector length. N is the document vector size and n is the relevant documents in the collection. This equation is further used in generating a feedback query after getting a relevant assessment from the user. This is found using equation 14.

$$Q_{l+1} = \alpha Q_l + \beta \sum_{rel} \frac{D_l}{|D_i|} - \gamma \sum_{nonrel} \frac{D_l}{|D_l|} \tag{14}$$

Here Q_1 represents the initial query. alpha, beta, gamma are the normalized term weights. In our implementation, we have set α=1, β = 1/n1 and γ = 1/n2. n1 is the number of relevant items and n2 is the number of nonrelevant items.

2.5.2 Implementation

In this task, we implement a PPR-based retrieval system that works along with user feedback to optimize the results. The PPR algorithm following Algorithm 1 and find the top m similar gestures to the given query gesture file. The steps below layout the exact process used in calculating the top 10 similar files using PPR and using the feedback to optimize the results further.

1. Calculate top 10 similar files to the given query file
2. Get user feedback for the retrieved results
3. Optimize the query vector using probabilistic or vectorized methods
4. Calculate similarity of new query vector with all files and update similarity matrix with these new values in the corresponding row of the initial query id
5. Add the positive results as seed nodes
6. Repeat 1-5 till user is satisfied

2.6 Task6

In this task we implement a query interface, which allows the user to provide a query and to tell the system on how many relevant query files had to be returned for the given query file. We are using graphical user interface to get the inputs from the user. We use the package tkinter to implement our graphical user interface. We use various algorithms to get the relevant query files for the given query file.

The algorithms we use for this task are KNN, LSH, PPR and probabilistic retrieval. KNN algorithm is used to retrieve a set of the necessary amount of similar gesture files to the query file. These similar gesture files are calculated by using either of the distance measures-Euclidean distance/Mahalanobis distance. PPR algorithm is also used to retrieve a set of the necessary amount of similar gesture files to the query file. To compute PPR we need the similarity matrix of the gesture files, we need k neighbours coming out of the gesture file and this k is inputted to the system by the user. We then compute u and v vectors. Now using the PPR equation we calculate the set of gesture files which is similar to the query file. The probability retrieval model returns a new query vector considering the probabilities of relevant and non relevant documents. This is again passed to all the algorithms for the upcoming iterations according to the user choice.

The system itself is made intelligent by remembering the user feedback from every iteration thus giving a better notion of relevant files for scoring the retrieved result sets from the aforementioned methods. We then use the results from the best method and show to the user. The confidence score is calculated as a combination of both variance of result set and the similarity of the result set with respect to user relevant files known so far. This gives a good trade-off between learning user profile and acting on the knowledge gained so far. The system uses the query vector that was used in showing results to user for further optimizations. Thus every iteration the search space based on query vectors gets pruned by one branch. We base this on the factor that the deeper the search space branches go the better the user profiling and eventually can cover the information contained in the pruned branches.

Algorithm 3: Customised Feedback Algorithm

Result: Remembers the user feedback to optimize and reassure the similar file retrieval
$results =$ Current output from the models
$prev\_result =$ Previous output from the model
$relevant =$ User feedback based on current results
$prob = 0.8$
if $prev\_result$ *present* **then**
    while $method$ *in* **do**
        $c =$ calculate relevant files count by finding intersection of files with relevant super-set
        $var =$ calculate variance by getting intersection between files and previous files
        $score =$ c / var
    Return maximum score method with probability $prob$
    Return random method with probability $1 - prob$
else
    Return random method
$prob = min(1, prob * 1.1)$

3 Interface specifications

In this section, we discuss the inputs and outputs for each task of this project phase.

1. task1
    - Inputs - (n - seed files, m - # of dominant gestures, k - nearest neighbours to keep)
    - Outputs - m dominant gestures
2. task2
    - Inputs - training data and algorithm to use
    - Outputs - classified test data
3. task3
    - Inputs - Query file name, maximum no of results to retrieve.
    - Outputs - Ranked list of retrieved file names and similarity scores.
4. task4
    - Inputs - (vm - Vector Model(TF or TF-IDF), query file, relevant or non relevant - Getting relevant feedback from the user)
    - Outputs - Giving files with relative importance to query file

5. task5
    - Inputs - (k - nearest neighbours to keep, query_file)
    - Outputs - top 10 files/gestures with feedback loop
6. task6
    - Inputs - (k - nearest neighbours to keep, query file, m - # of relevant files to be found, relevant or non relevant or don't know - Getting relevant feedback from the user)
    - Outputs - Giving files with relative importance to query file

4 System requirements

The proposed system requires a functional python 3 environment and a few popular libraries mentioned below. The programs take user inputs for tasks 0b through 4 to run, command line arguments for task0a.

1. Language: Python
2. Version:3 and above
3. Operating System: Windows or later, macOS, and Linux
4. Processors: Intel Atom® processor or Intel® CoreTM i5 processor
5. RAM: 6GB or more memory
6. Hard-disk drive: 50 GB

Libraries to be installed:

1. numpy
2. scipy
3. pandas
4. matplotlib
5. seaborn
6. scikit-learn
7. tkinter
8. path
9. os
10. pickle
11. json
12. threadpool
13. argparse
14. glob

Commands to execute the code:
To run the codes, we need to first place every python file in the parent directory of the data folder In terminal/ command prompt,

1. For task 1 -> python3 task1.py
2. For Task 2 -> python3 task2.py
3. For Task 3 -> python3 task3.py
4. For Task 4 -> python3 task4.py
5. For Task 5 -> python3 task5.py
6. For Task 6 -> python3 task6.py

5 Related work

Various exemplar-based methods are therefore proposed to circumvent the difficulties of model learning, by leveraging invariant visual representations and direct matching of example gestures. Among those visual representations, local spatio-temporal features [4, 5, 6] are the most widely exploited, though most of them are used in human action recognition. Other descriptors include motion trajectories [7], spatio-temporal gradients [8] and global histograms of optical flow [9]. However, most of these methods try to directly match the exemplars, without offering a scalable solution for efficient matching when the exemplar database is large. A few others have adopted the bag-of features framework with local spatio-temporal features for human action recognition. Shen et al proposed a new visual representation for hand motions based on the motion divergence fields, that can be normalized to gray-scale images such as heatmap [10]. The idea is to match a new gesture sample with database gestures through a term frequency-inverse document frequency (TF-IDF) mechanism. Florez et al presented a paper that uses the topology of a self-organizing neural network and its dynamics to determine hand postures and gestures [11]. In this method, the system has been trained with 12 gestures, some of which are very similar, and have obtained high success rates (over 97%).Meanwhile, the extraction of these features is generally slow, though some efforts toward real-time extraction and recognition are being made. Therefore, an efficient visual representation for real-time feature extraction and a scalable gesture matching over large exemplar databases is still highly desirable.

6 Results and Inference - 3 Class Test Dataset

In this section, we would be reporting the results obtained on the 3 class dataset for various options of vector and semantic models used in previous phase. For reference, we use the following nomenclature henceforth to refer to the different vector and semantic models.

- vector models
  - tf
  - tfidf
- semantic models
  - pca
  - svd
  - nmf
  - lda

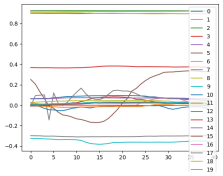
6.1 Task 1

In this task, we implement the PPR algorithm and find the top m dominant gestures given a set of seed nodes. The algorithm is defined in Section 2.1. The dominant gestures can also have an intersection with the set of seed nodes given by the user. Given this, we tried to implement this method for seed files with different types of input and observed the returned results from the system. Table 1 shows the top m dominant gestures retrieved for different sets of seed files/gestures.

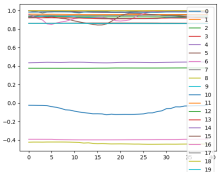
Table 1: Task1 PPR Dominant Gestures

user_files	k_value	dominant_gestures
[1, 3, 5]	3	[0000001, 0000005, 0000003, 00012-7, 00012-0, 00012-1, 00012-8, 00012-9, 00012-2, 00012-3]
[1, 40, 72]	3	[0000269, 0000001, 00001-8, 00012-1, 00012-8, 00012-9, 00012-2, 00012-3, 00004-9, 00004-3]
[1, 3, 5]	5	[0000003, 0000005, 0000001, 00012-7, 00012-0, 00012-1, 00012-8, 00012-9, 00012-2, 00012-3]
[1, 40, 72]	5	[0000269, 0000001, 00001-8, 00265-2, 00265-7, 00265-6, 00265-3, 00012-1, 00012-8, 00012-9]

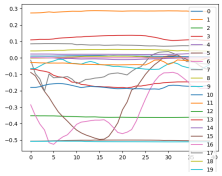
As we can see, when the seed files are more correlated with one another, the dominant gestures returned by PPR are usually within the same class. This shows that PPR is a strong algorithm that can be used for query retrieval once we have a good idea of similar files or gestures. This factor would be greatly exploited when we look into the results of tasks 5 and 6. Furthermore, Figures 1 to 10 shows the time series plots of dominant gestures obtained in this task.



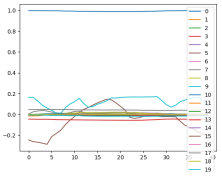
(a) file1_W



(b) file1_X

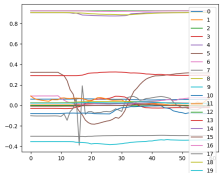


(c) file1_Y

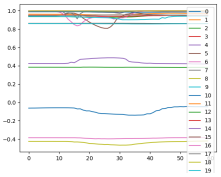


(d) file1_Z

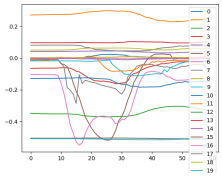
Figure 1: File 1



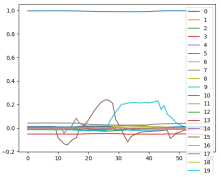
(a) file3_W



(b) file3_X

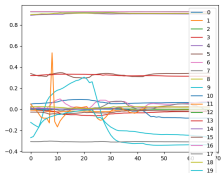


(c) file3_Y

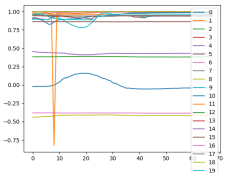


(d) file3_Z

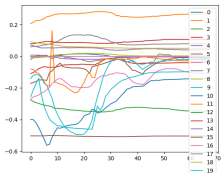
Figure 2: File 2



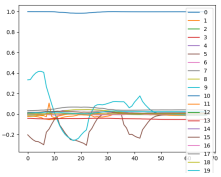
(a) file5_W



(b) file5_X

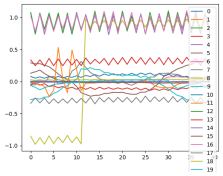


(c) file5_Y

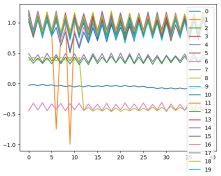


(d) file5_Z

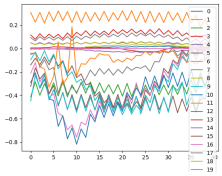
Figure 3: File 3



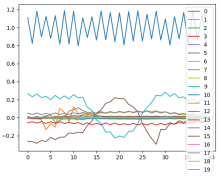
(a) file12-0_W



(b) file12-0_X

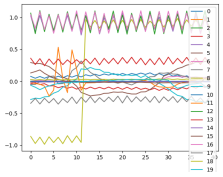


(c) file12-0_Y

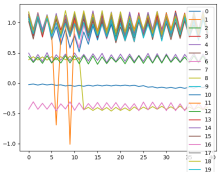


(d) file12-0_Z

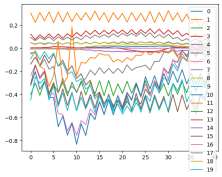
Figure 4: File 4



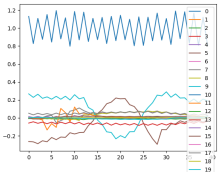
(a) file12-1_W



(b) file12-1_X

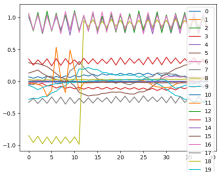


(c) file12-1_Y

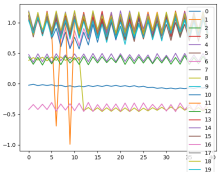


(d) file12-1_Z

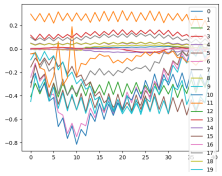
Figure 5: File 5



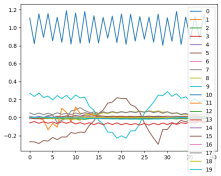
(a) file12-2_W



(b) file12-2_X

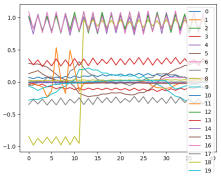


(c) file12-2_Y

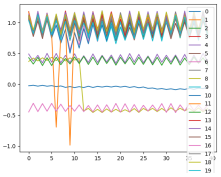


(d) file12-2_Z

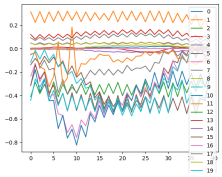
Figure 6: File 6



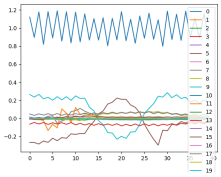
(a) file12-3_W



(b) file12-3_X

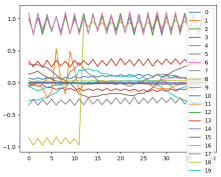


(c) file12-3_Y

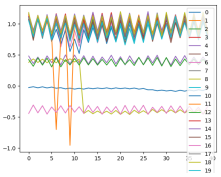


(d) file12-3_Z

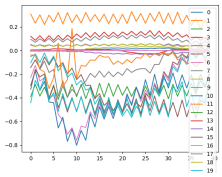
Figure 7: File 7



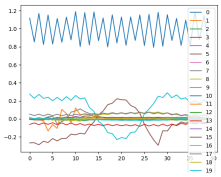
(a) file12-7_W



(b) file12-7_X

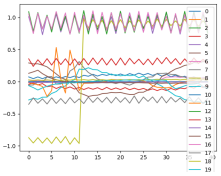


(c) file12-7_Y

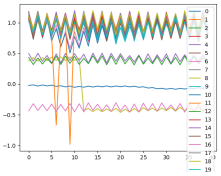


(d) file12-7_Z

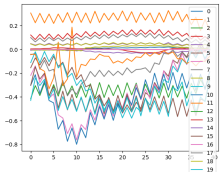
Figure 8: File 8



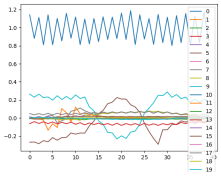
(a) file12-8_W



(b) file12-8_X

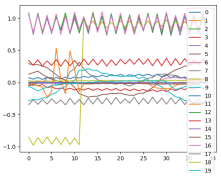


(c) file12-8_Y

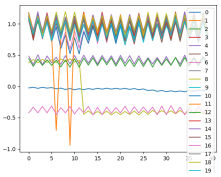


(d) file12-8_Z

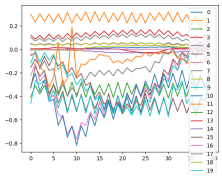
Figure 9: File 9



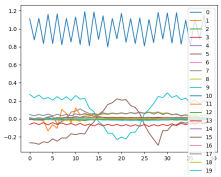
(a) file12-9_W



(b) file12-9_X



(c) file12-9_Y



(d) file12-9_Z

Figure 10: File 10

6.2 Task 2

In this task, we explore the classification efficiency of 3 types of algorithms - Decision Tree, Kth Nearest Neighbour and Personalized Page Rank. To understand the variance and significance of vectors used in the computations, we experiment over different semantic and vector models.

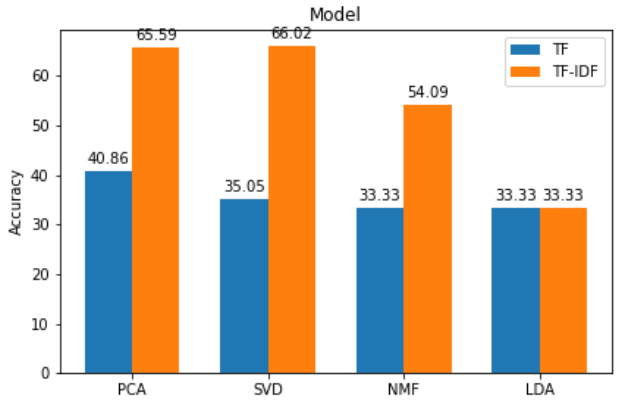


Figure 11: Decision Tree Accuracy Bar Plot

Figure 11 shows the bar plot that depicts the effectiveness of Decision Tree algorithm in classifying the remaining gestures of the dataset given a set of training gestures. We can see from the figure that using tf based vectors across different semantic models usually outperforms tfidf based vectors except in the case of LDA. This can be attributed to the factor that being a generative model, LDA is able to better capture the distribution of the dataset when more discrimination among files in introduced into the vectors, viz-a-viz, tfidf vectors.

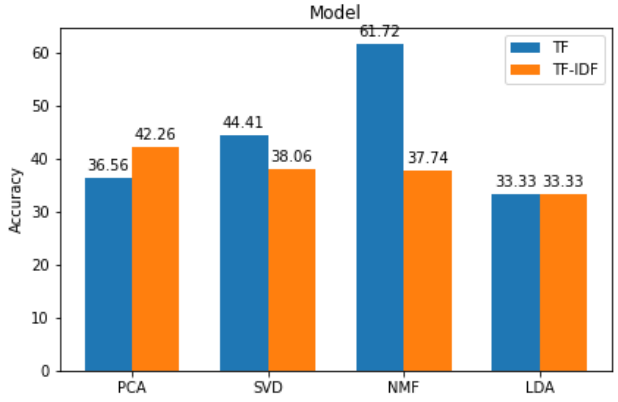
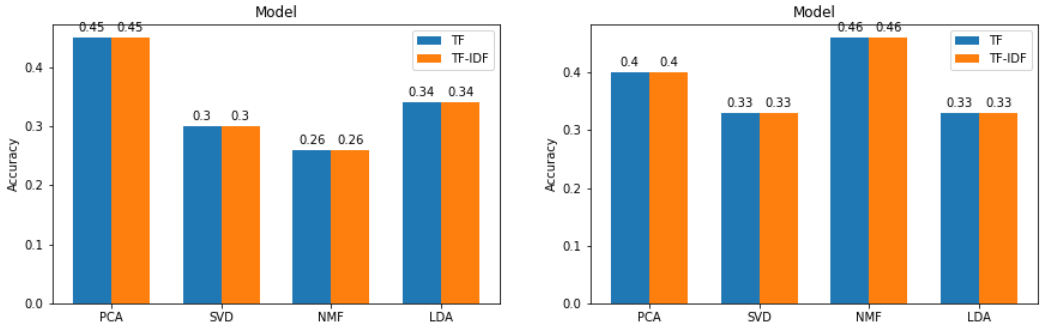


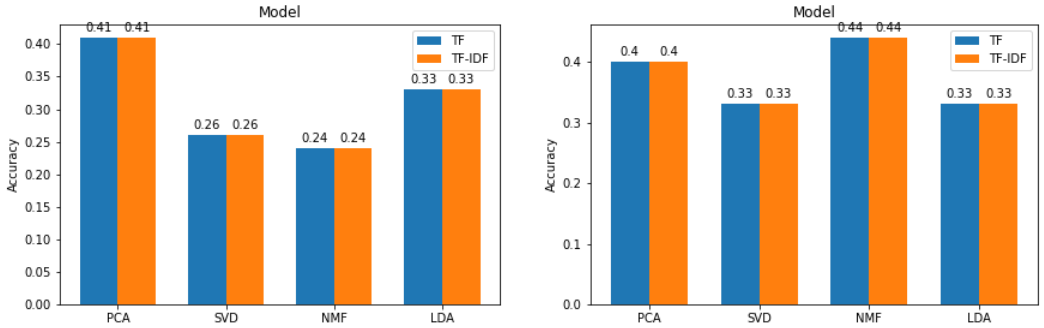
Figure 12: K Nearest Neighbour Accuracy Bar Plot

Figure 12 shows the bar plot that depicts the effectiveness of K^{th} Nearest Neighbour (KNN) algorithm in classifying the remaining gestures of the dataset given a set of training gestures. We can see from the figure that using tf based vectors across different semantic models usually outperforms tfidf based vectors except in the case of PCA, which is again only by a small amount. This can be attributed to the fact that KNN being a simple and lazy algorithm, prefers to use more descriptive vectors of the files that can help associate them with one another using distance measures than discriminative measures that would be induced by tfidf vectors.



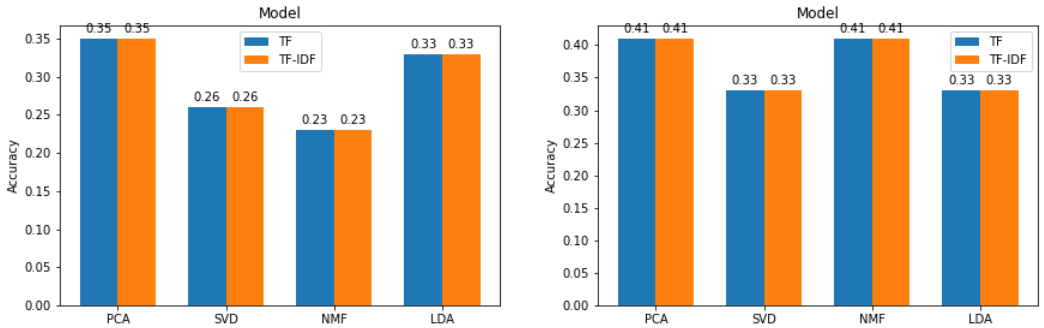
(a) old (b) new

Figure 13: Personalized Page Rank Accuracy Bar Plot k=20



(a) old (b) new

Figure 14: Personalized Page Rank Accuracy Bar Plot k=25



(a) old (b) new

Figure 15: Personalized Page Rank Accuracy Bar Plot k=30

Figures 13 to 15 shows the bar plot that depicts the accuracy of Personalized Page Rank (PPR) algorithm in classifying the remaining gestures of the dataset given a set of training gestures. We can see from the figure that using either tf based or tfidf based vectors across different semantic models makes no big difference. This can be attributed to the fact that PPr, using the similarity matrix, depends on the cosine similarity between the vectors representing the gestures and hence as observed from the previous phase, the matrices are not best describers of the dataset and their classes. Thus even though PPR is a strong algorithm, it fails to perform as well as it did in the previous task where is achieved around 90% accuracy in similar files retrieval with multiple similar seed files.

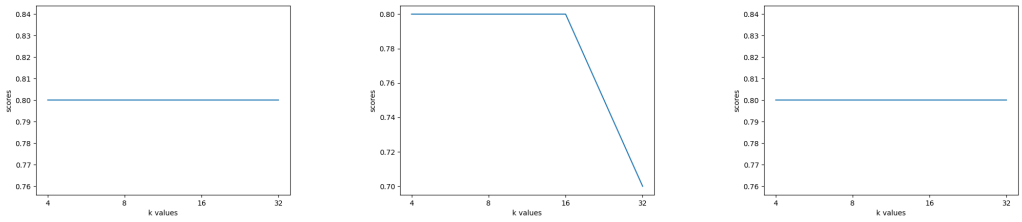
Another curious observation is that using the second method of PPR algorithm, we can see small improvements in accuracy scores across all models. Furthermore, it can be observed that the higher the neighbourhood value for PPR graph, the better the results. This can be attributed to the fact that more links/edges corresponds to more viable paths for random walk and better probability estimates.

6.3 Task 3

We analyze the retrieval performance of LSH on data from all 3 classes. To measure performance we retrieve the 10 most similar files for a user inputted query file. We choose a query file from each of the 3 classes to ensure consistency. For each retrieval we compute a **relevance_score**. This score is computed by analyzing the class label of the query file and each of the retrieved files. The score is computed as follows:

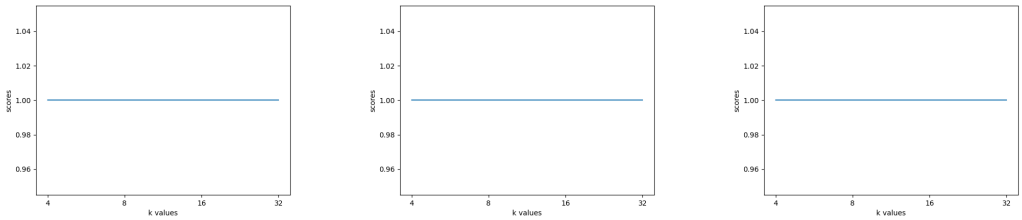
$$relevance\_score = \frac{\sum_1^m (C_i = C_q)}{m} \tag{15}$$

where m is the number of files retrieved, C_q is the query file's class label. We plot graphs for a fixed value of L and $k \in [4, 8, 16, 32]$. Through empirical observation, we find $L = 4$ and $k = 8$ give us the best results, and we use this in **Task 6**. All graphs below are shown for vector models decomposed by PCA algorithm.



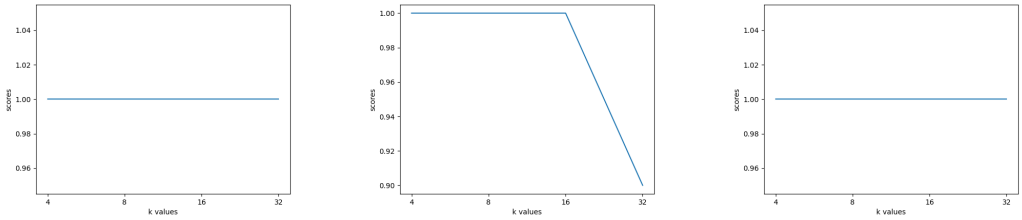
(a) L=4 (b) L=8 (c) L=12

Figure 16: LSH relevance scores for File: 4_6 with TF vectors and k ∈ (4,8,16,32)



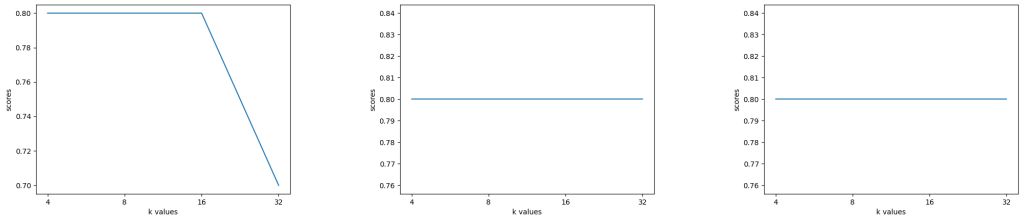
(a) L=4 (b) L=8 (c) L=12

Figure 17: LSH relevance scores for File: 4_6 with TF-IDF vectors and k ∈ (4,8,16,32)



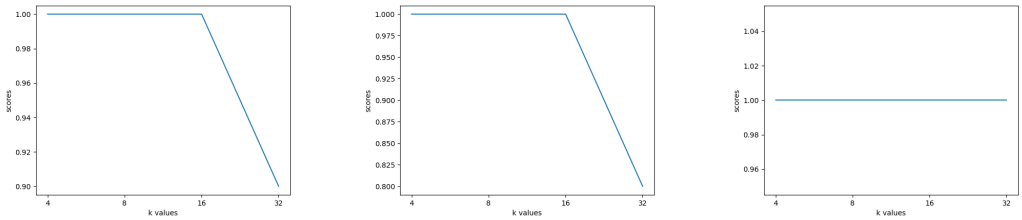
(a) L=4 (b) L=8 (c) L=12

Figure 18: LSH relevance scores for File: 275 with TF vectors and k ∈ (4,8,16,32)



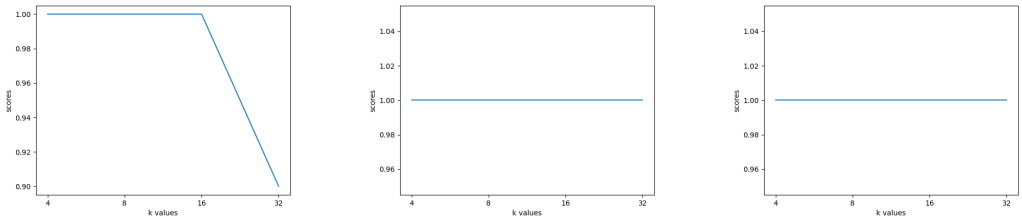
(a) L=4 (b) L=8 (c) L=12

Figure 19: LSH relevance scores for File: 275 with TF-IDF vectors and k ∈ (4,8,16,32)



(a) L=4 (b) L=8 (c) L=12

Figure 20: LSH relevance scores for File: 570_3 with TF vectors and k ∈ (4,8,16,32)



(a) L=4 (b) L=8 (c) L=12

Figure 21: LSH relevance scores for File: 570_3 with TF-IDF vectors and k ∈ (4,8,16,32)

6.4 Task 4

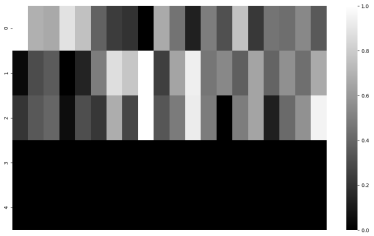
The heat maps are used for visualization of importance of sensors, components and words over iterations. The x-axis represents different components/sensors/words and the y-axis represents the iteration number. White shade represents increase in importance and black shade represents decrease in importance of either words, sensors, components. The heat map that represents the importance of components over iterations, is generated in such a way that the first four columns represent X,Y,W,Z respectively. The heat maps depicting the importance of sensors is in such a way that the 20 columns represent sensors 1-20 respectively.

The figure 22 represents change of components, sensors, words for both TF and TF IDF vector models for the file 6 from Class-1.

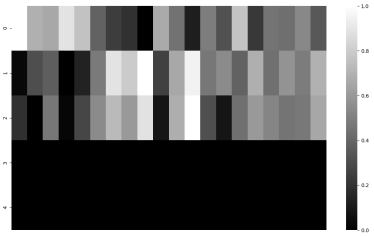
As we can see in the figure 22a, the importance of the X component has increased from 0th iteration to 3rd iteration for TF vector model. The importance of Y,W,Z components have decreased from 0th iteration to 3rd iteration. After 3 iterations the relative importance of all the components is decreased. The importance of these components have changed purely based on the user feedback on relevant and irrelevant documents and then revising the query according to this feedback.

(a) Component change when vector model = TF

(b) Component change when vector model = TF-IDF



(c) Sensor change when vector model = TF



(d) Sensor change when vector model = TF-IDF

(e) Words change when vector model = TF

(f) Component change when vector model = TF-IDF

Figure 22: Relative importance of sensors, components, different words in File 6 over 4 iterations

In the figure 22b, for TF-IDF vector model, the importance of X component has increased from the initial iteration to 4th iteration while the importance of all other components have decreased over time.

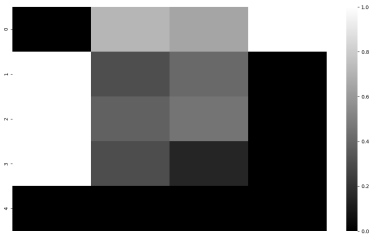
The figure 22c, represents the importance of sensor 0 that has decreased over iterations while the importance of sensor 8 and 11 that have increased over time for TF vector model.

The figure22d represents the importance of sensor 0 and 9 that have decreased over iterations while the importance of sensor 8 and 11 that have increased over time for TF IDF vector model.

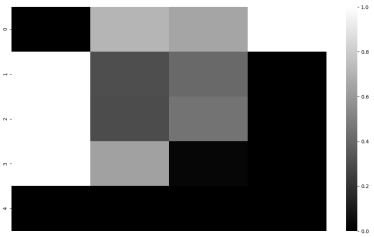
The figures 22e and 22b represent that the relative importance of the words have increased drastically in the second iteration for a few words. In the 3rd iteration, the importance of these words decreased relatively and from the next iteration onwards, their importance decreased uniformly.

The figure 23 represents change of components, sensors, words for both TF and TF IDF vector models for the file 257 from Class-2.

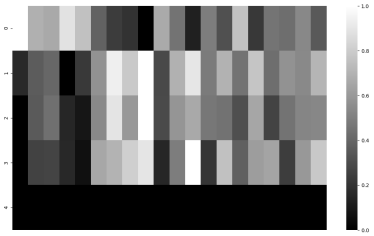
As we can see in the figure 23a, the importance of the X component has increased from 0th iteration to 3rd iteration for TF vector model. The importance of Y,W components have decreased from 0th iteration to 3rd iteration. The importance of Z component is decreased drastically in the second iteration itself. After 4 iterations the relative importance of all the components is decreased. The importance of these components have changed purely based on the user feedback on relevant and irrelevant documents and then revising the query according to this feedback.



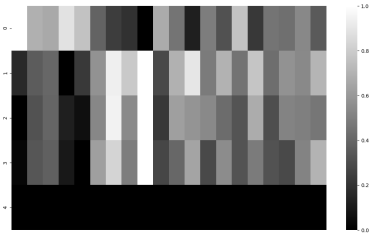
(a) Component change when vector model = TF



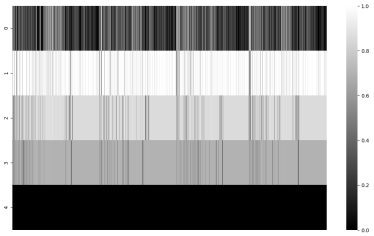
(b) Component change when vector model = TF-IDF



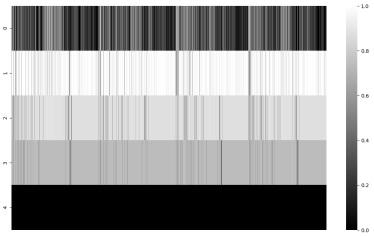
(c) Sensor change when vector model = TF



(d) Sensor change when vector model = TF-IDF



(e) Words change when vector model = TF



(f) Words change when vector model = TF-IDF

Figure 23: Relative importance of sensors, components, different words in File 257 over 4 iterations

In the figure 23b, for TF-IDF vector model, the importance of X component has increased from the initial iteration to 4th iteration while the importance of all other components have decreased over time.

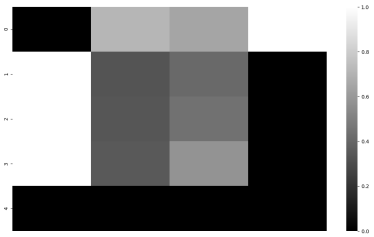
The figure 23c, represents the importance of sensor 0 that has decreased over iterations while the importance of sensor 6 and 8 that have increased over time for TF vector model. The importance of all the sensors have decreased after 4 iterations.

The figure23d represents the importance of sensor 0 that has decreased over iterations while the importance of sensor 6 and 8 that have increased over time for TF IDF vector model.

The figures 23e and 23f represent that the relative importance of the words have increased drastically in the second iteration for a few words. In the 3rd iteration, the importance of these words decreased relatively and from the next iteration onwards, their importance decreased uniformly.

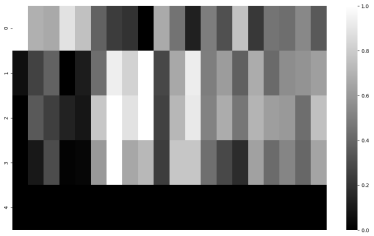
The figure 24 represents change of components, sensors, words for both TF and TF IDF vector models for the file 565 from Class-3.

As we can see in the figure 24a, the importance of the X component has increased from 0th iteration to 4th iteration for TF vector model. The importance of Y,W components have decreased from 0th iteration to 3rd iteration. The importance of Z component is decreased drastically in the second iteration itself. After a few iterations the relative importance of all the components is decreased. The importance of these components have changed purely based on the user feedback on relevant and irrelevant documents and then revising the query according to this feedback.

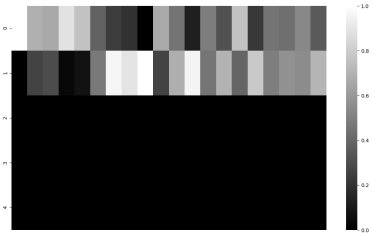


(a) Component change when vector model = TF

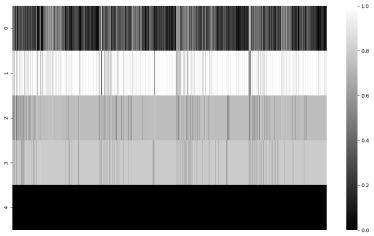
(b) Component change when vector model = TF-IDF



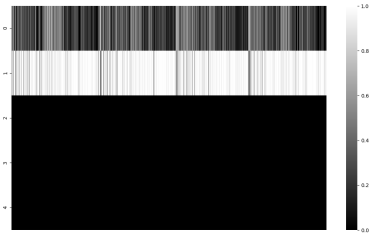
(c) Sensor change when vector model = TF



(d) Sensor change when vector model = TF-IDF



(e) Words change when vector model = TF



(f) Words change when vector model = TF-IDF

Figure 24: Relative importance of sensors, components, different words in File 565 over 4 iterations

In the figure 24b, for TF-IDF vector model, the importance of X component has increased from the initial iteration to the next iteration while the importance of Y ans W components have decreased. The importance of Z component has decreased drastically from first iteration to the next.The importance of all 4 components have decreased from the 3rd iteration onwards for the TF IDF vector model.

The figure 24c, represents the importance of sensor 0 that has decreased over iterations while the importance of sensor 7 and 9 that have increased over time for TF vector model. The importance of all the sensors have decreased after 3 iterations.

The figure24d represents the importance of sensor 0 that has decreased over iterations while the importance of sensor 7, 9 and 12 that have increased over time for TF IDF vector model. The importance of these sensors have decreased altogether from 3rd iterations.

The figure 24e represents the importance of words for the TF vector model. The word's importance have increased in the first few iterations while it kept on decreasing in the later iterations.

The figure 24f represent that the relative importance has increased drastically in the second iteration for a few words while the importance decreased after the 3rd iteration onwards for TF IDF vector model.

Table 2: Sensor and its location in the human body

Sensor ID	Body location
1	HipCenter
2	Spine
3	ShoulderCenter
4	Head
5	ShoulderLeft
6	ElbowLeft
7	WristLeft
8	HandLeft
9	ShoulderRight
10	ElbowRight
11	WristRight
12	HandRight
13	HipLeft
14	KneeLeft
15	AnkleLeft
16	FootLeft
17	HipRight
18	KneeRight
19	AnkleRight
20	FootRight

(a) Vattene

(b) Combinato

(c) Daccordo

Figure 25: Gestures

6.4.1 Relation of Heatmap with gestures- Vattene, Combinato, Daccordo

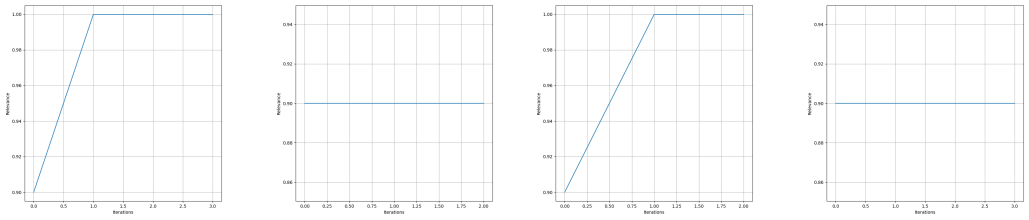
As we can see in the Table 2, the sensors 5-12 correspond to the entire hand movement of a person. The sensor 5-8 are placed in the left hand of the human body while the sensors 9-12 are placed in the right hand of the human body. The other sensors 1-4 and 13-20 are placed in other parts of the body. The Figures 25 represents the gestures- Vattene, Combinato and Daccordo which involves movement of the hand of a person and not any other body parts. The heatmaps in figures 22 c & 22 d, figure 23 c & 23 d and figure 24 c & 24 d clearly depicts that the relative importance of sensors 5-12 have changed when compared to other sensors 1-4 and 13-20.

For file 6 that belongs to Class-1 and the Vattene gesture, the figures 22 c & 22 d, depict that the importance have increased for sensors 9 and 12 in both TF and TF IDF vector models. These sensors corresponds to the right hand of the person and the Vattene gesture is also performed using right hand only. Hence, it is justified that over iterations with query optimisation we are able to retrieve accurate results.

For file 257 that belongs to Class-2 and the Combinato gesture, the figures 23 c & 23 d, depict that the importance have increased for sensors 7, 9 and 12 in both TF and TF IDF vector models. These sensors corresponds to both left and right hand of the person and the Combinato gesture is also performed using both left and right hands. Hence, it is justified that over iterations with query optimisation we are able to retrieve accurate results.

For file 565 that belongs to Class-3 and the Daccordo gesture, the figures 23 c & 23 d, depict that the importance have increased for sensor 9 in both TF and TF IDF vector models. This sensor corresponds to right hand of the person and the Daccordo gesture is also performed using right hand only. Hence, it is justified that over iterations with query optimisation we are able to retrieve accurate results.

6.5 Task 5



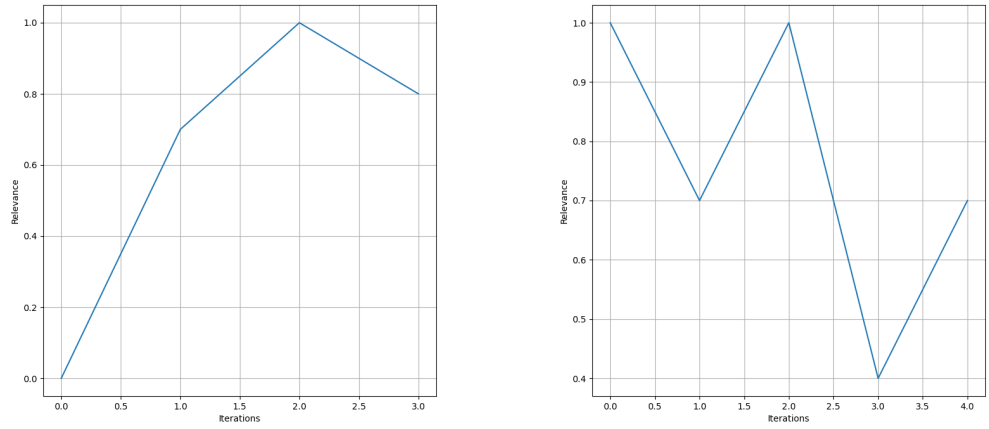
(a) With Query Opt (b) Without Query Opt (c) With Query Opt (d) Without Query Opt

Figure 26: Relevance scores v/s iterations

From Figure 26, we can observe that using query optimization on top of adding new seed nodes makes PPR convergence to more relevant solutions quicker. The idea behind this can be explained by a simple vector-space derivation. During query optimization, we try to modify the original query file/vector by adding more information from relevant files and remove information of irrelevant files. In doing so we construct a new vector/gesture that now acts as an embodied query gesture which is a better node than the existing query node in the graph. So we replace the new node over the old node in the graph by substituting the values in similarity matrix and then recompute PPR from the new node. This makes the algorithm more effective and efficient in reaching similar nodes from whose vectors, the new query node has gained more information with iterations. The vitality of performing this step can be easily observed from Figure 26 itself where the relevance score jumps rapidly when using this method.

6.6 Task 6

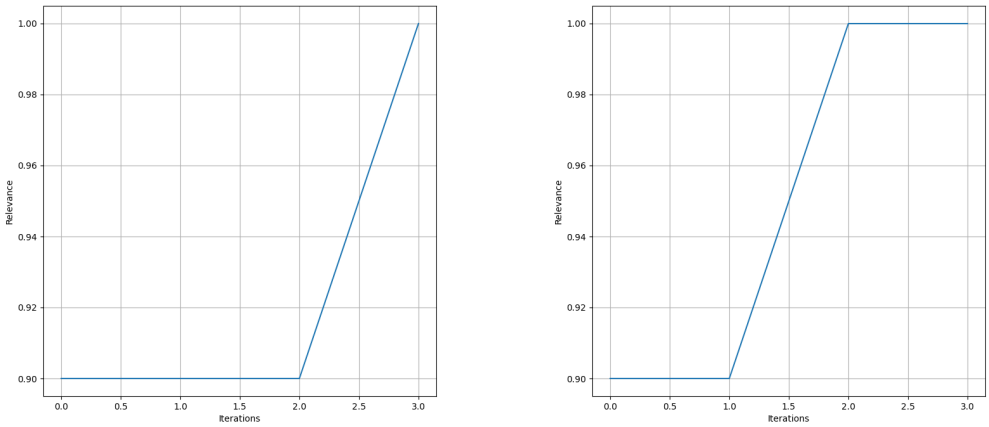
In this task we implement a customised intelligent algorithm to get the most optimal and relevant files for the given query file. From Figure 27 we can come to a conclusion that PCA on TF vector performs better than PCA on TF-IDF vectors. From visualizing the plot we can see PCA on TF vector converges to 1 on very less iterations compared to PCA on TF-IDF vectors.



(a) File 4-6 (TF and PCA) (b) File 4-6 (TF-IDF and PCA)

Figure 27: Relevance Plot for PCA

From Figure 28 we can come to a conclusion that SVD on TF-IDF vector performs better than SVD on TF vectors. From visualizing the plot we can see SVD on TF-IDF vector converges to 1 on very less iterations compared to SVD on TF vectors.



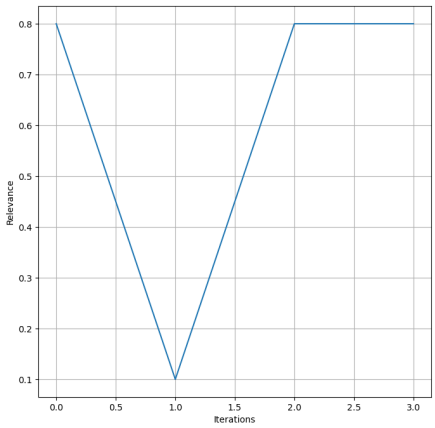
(a) File 4-6 (TF and SVD) (b) File 4-6 (TF-IDF and SVD)

Figure 28: Relevance Plot for SVD

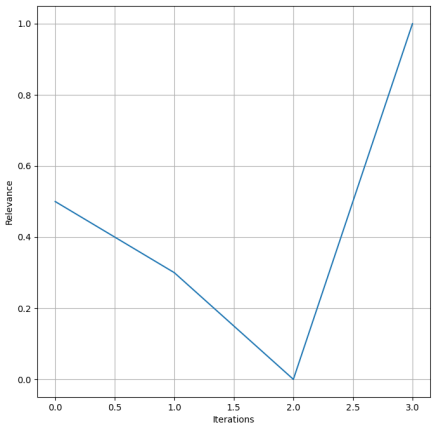
From Figure 29 we can come to a conclusion that NMF on TF vector performs better than NMF on TF-IDF vectors. From visualizing the plot we can see NMF on TF vector converges to 1 on very less iterations compared to NMF on TF-IDF vectors.

From Figure 30 we cannot come to a conclusion whether LDA on TF vector or LDA on TF-IDF vectors performs better than each other. From visualizing the plot we can see LDA on TF vector and LDA on TF-IDF vectors converges to 1 on same number of iterations.

From Figure 31 we can come to a conclusion TF-IDF vector performs better than TF vectors. From visualizing the plot we can see TF-IDF vector converges to 1 on very less iterations compared to TF vectors.

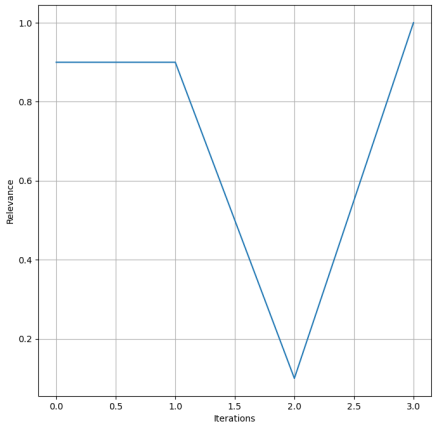


(a) File 4-6 (TF and NMF)

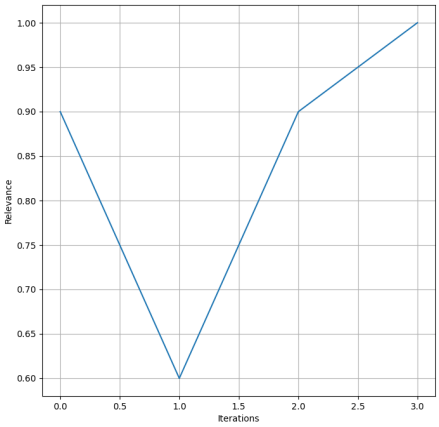


(b) File 4-6 (TF-IDF and NMF)

Figure 29: Relevance Plot for NMF

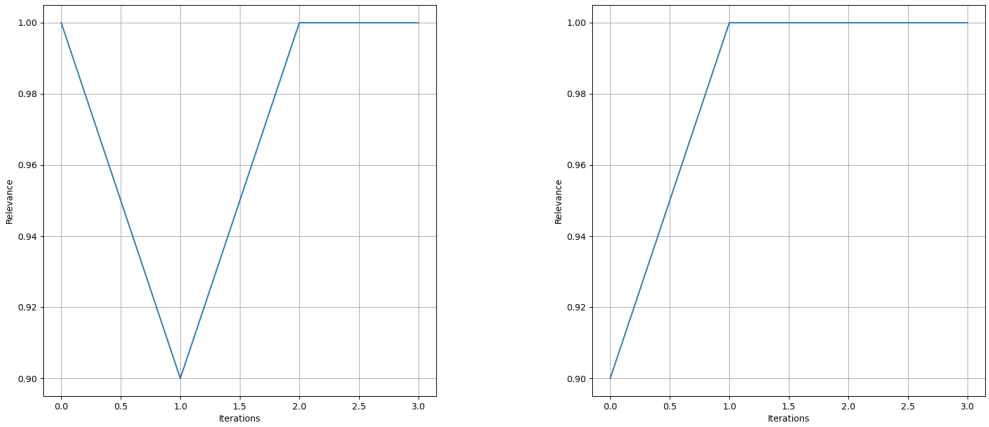


(a) File 4-6 (TF and LDA)



(b) File 4-6 (TF-IDF and LDA)

Figure 30: Relevance Plot for LDA



(a) File 4-6 (TF)

(b) File 4-6 (TF-IDF)

Figure 31: Relevance Plot for TF and TF-IDF

Figure 32 is the sample of our User Interface. Using this interface sample we show how the user has to enter the query file and the number of files that has to be retrieved similar to the query file inputted.

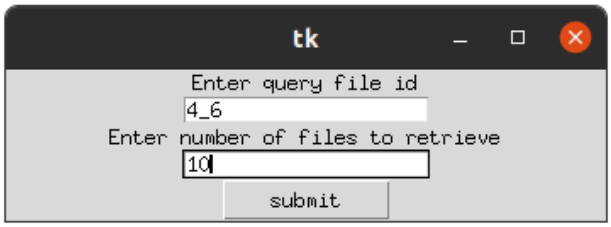
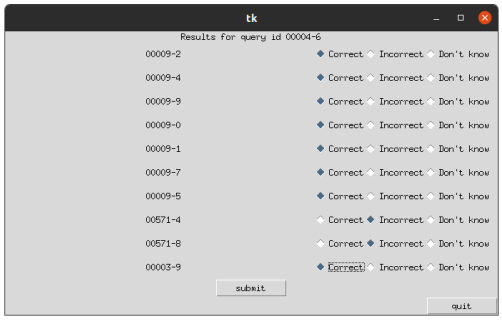
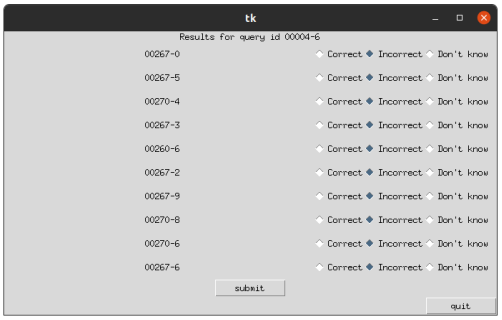


Figure 32: User Interface for getting query file ID and total number of files to retrieve

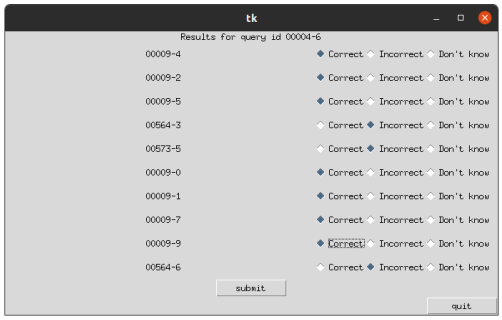
Figure 33, 34, 35, 36 and 37 is the sample of our User Interface for getting feedback from the user. We have three options based on which we further optimize our queries for effective retrieval of similar gesture files in relation with the query file.



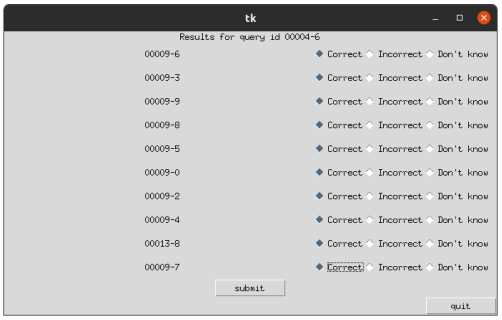
(a) File 4-6 (TF and PCA)



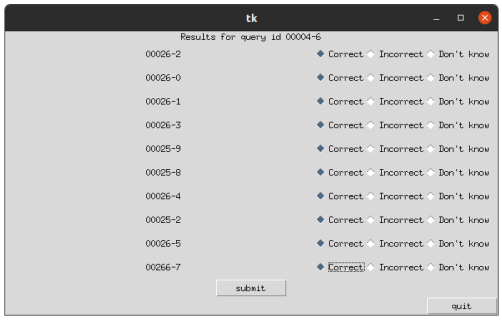
(b) File 4-6 (TF and PCA)



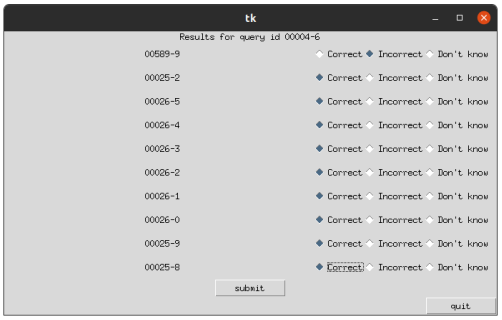
(c) File 4-6 (TF and PCA)



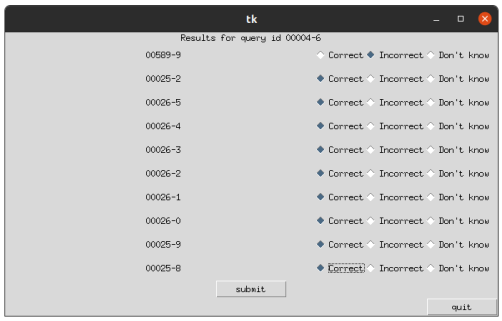
(d) File 4-6 (TF and PCA)



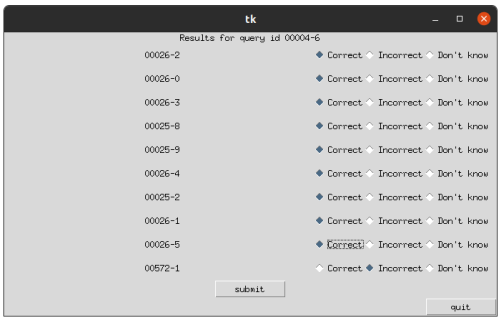
(e) File 4-6 (TF and SVD)



(f) File 4-6 (TF and SVD)

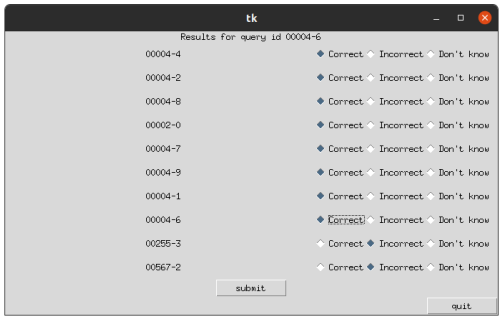


(g) File 4-6 (TF and SVD)

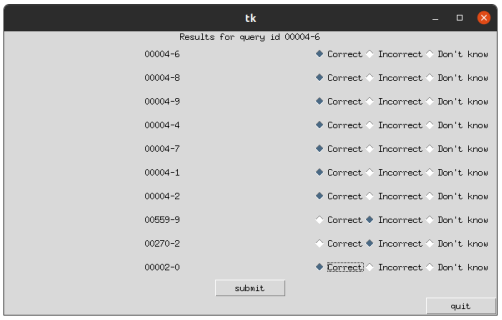


(h) File 4-6 (TF and SVD)

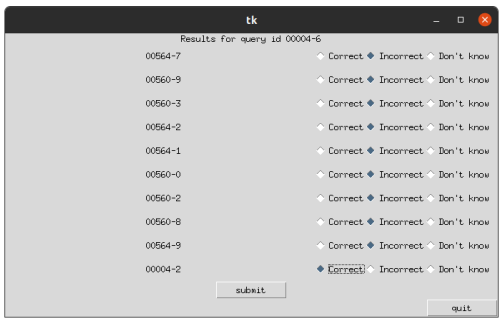
Figure 33: User Interface for getting feedback for PCA and SVD



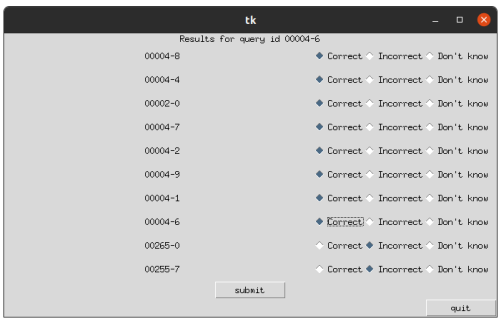
(a) File 4-6 (TF and NMF)



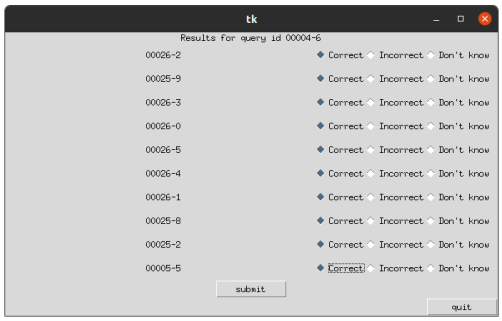
(b) File 4-6 (TF and NMF)



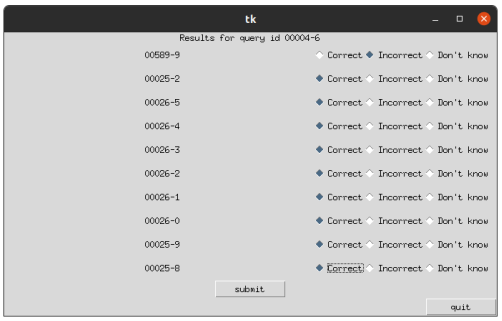
(c) File 4-6 (TF and NMF)



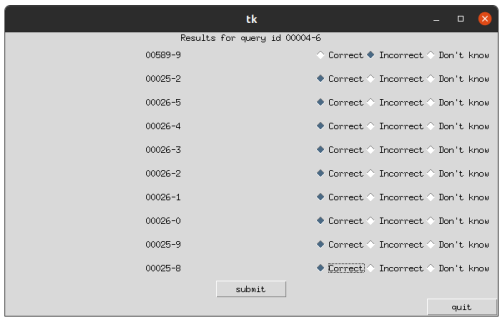
(d) File 4-6 (TF and NMF)



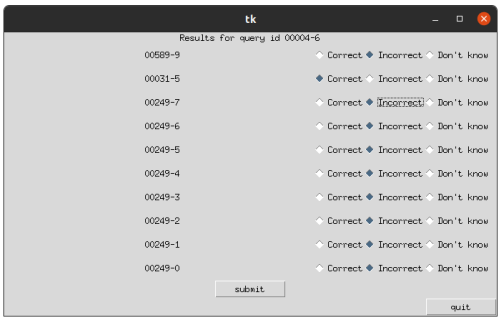
(e) File 4-6 (TF and LDA)



(f) File 4-6 (TF and LDA)

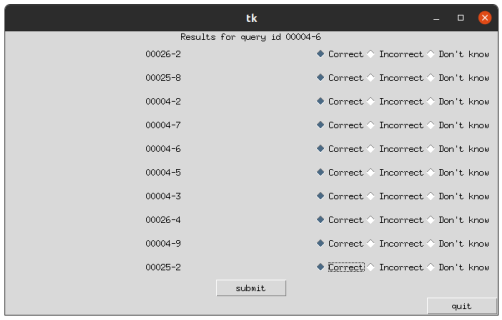


(g) File 4-6 (TF and LDA)

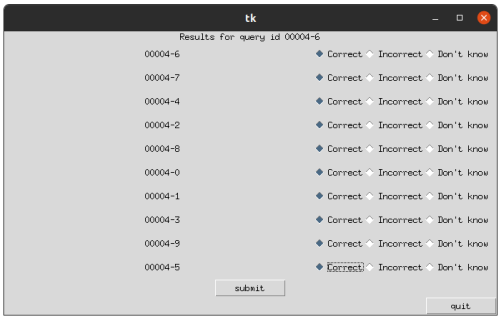


(h) File 4-6 (TF and LDA)

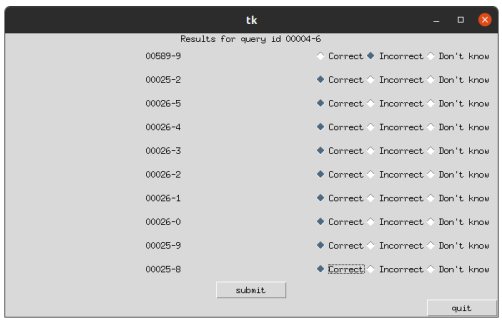
Figure 34: User Interface for getting feedback for NMF and LDA



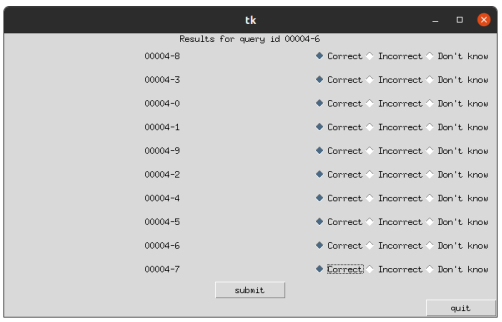
(a) File 4-6 (TF)



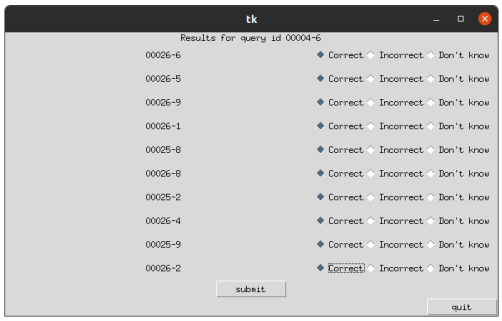
(b) File 4-6 (TF)



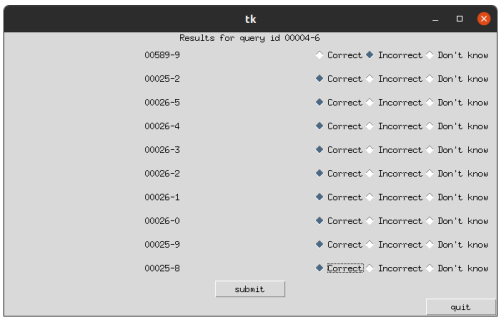
(c) File 4-6 (TF)



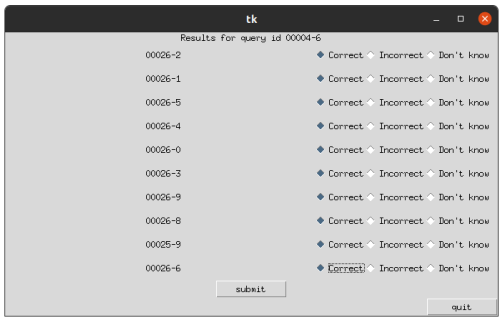
(d) File 4-6 (TF)



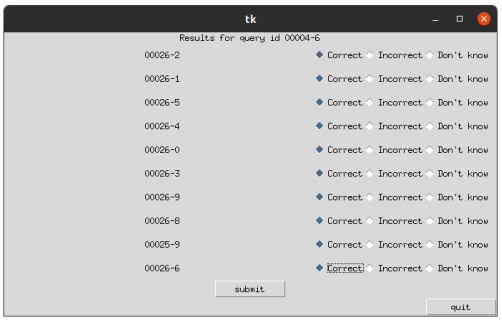
(e) File 4-6 (TF-IDF)



(f) File 4-6 (TF-IDF)

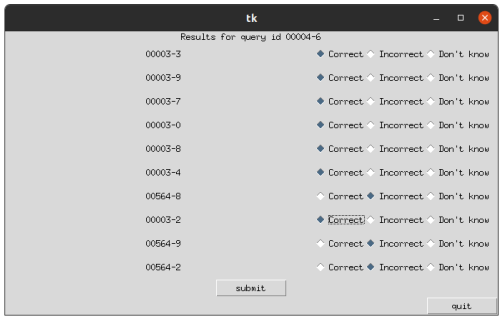


(g) File 4-6 (TF-IDF)

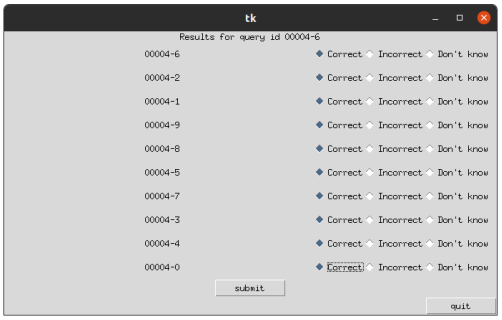


(h) File 4-6 (TF-IDF)

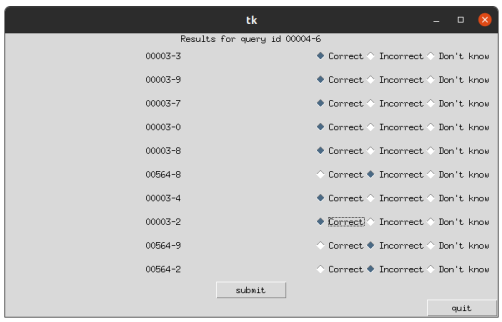
Figure 35: User Interface for getting feedback for TF and TF-IDF



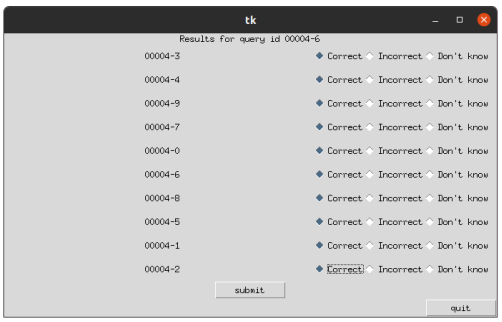
(a) File 4-6 (TF-IDF and PCA)



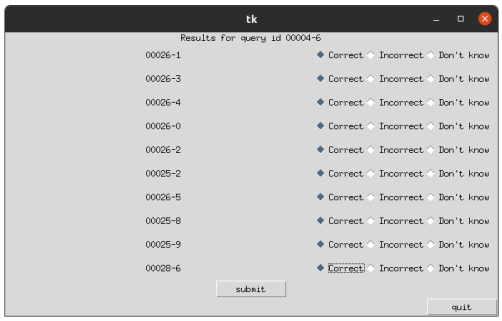
(b) File 4-6 (TF-IDF and PCA)



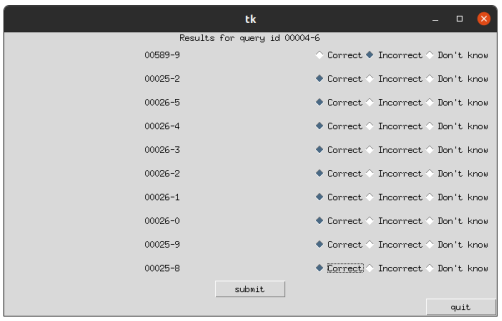
(c) File 4-6 (TF-IDF and PCA)



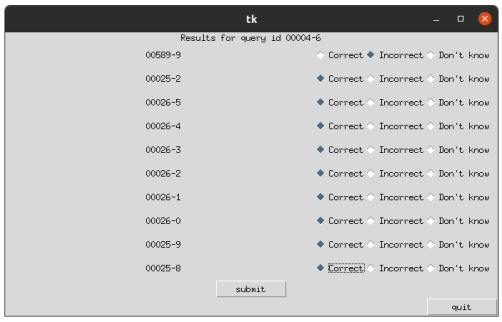
(d) File 4-6 (TF-IDF and PCA)



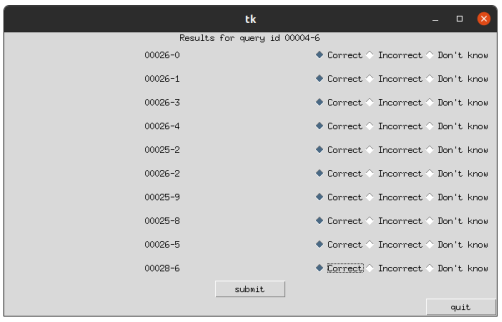
(e) File 4-6 (TF-IDF and SVD)



(f) File 4-6 (TF-IDF and SVD)

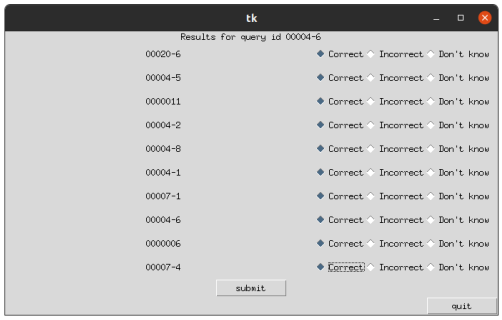


(g) File 4-6 (TF-IDF and SVD)

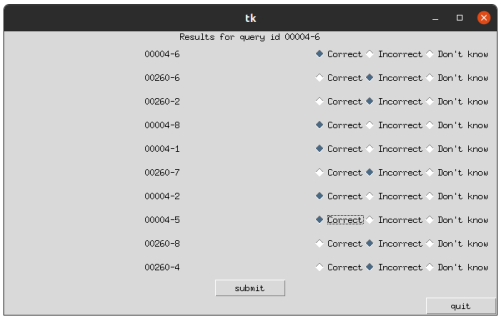


(h) File 4-6 (TF-IDF and SVD)

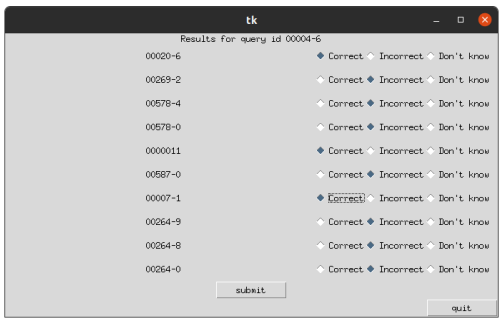
Figure 36: User Interface for getting feedback for PCA and SVD



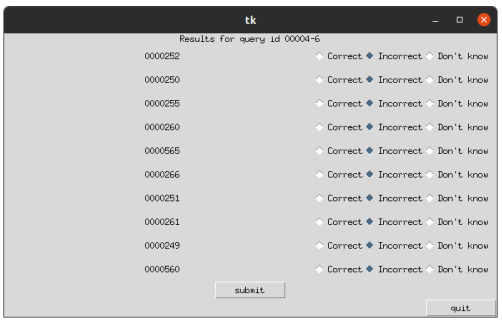
(a) File 4-6 (TF-IDF and NMF)



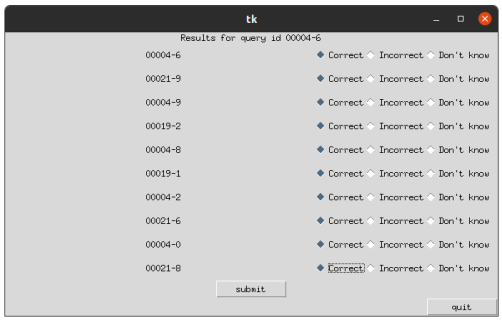
(b) File 4-6 (TF-IDF and NMF)



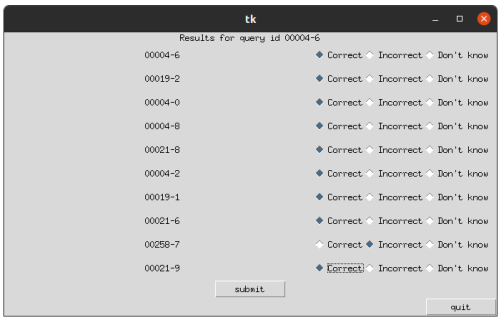
(c) File 4-6 (TF-IDF and NMF)



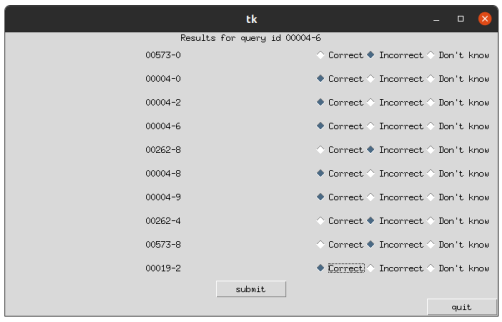
(d) File 4-6 (TF-IDF and NMF)



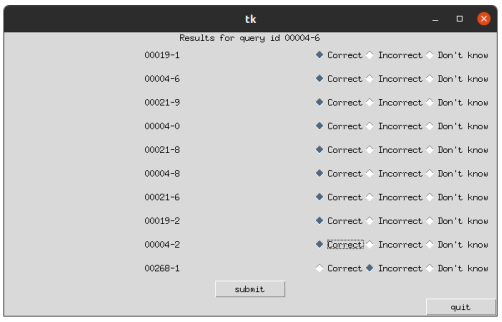
(e) File 4-6 (TF-IDF and LDA)



(f) File 4-6 (TF-IDF and LDA)



(g) File 4-6 (TF-IDF and LDA)



(h) File 4-6 (TF-IDF and LDA)

Figure 37: User Interface for getting feedback for NMF and LDA

References

[1] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu, "Automatic multimedia cross-modal correlation discovery," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004, pp. 653–658.

[2] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *2006 47th annual IEEE symposium on foundations of computer science (FOCS'06)*. IEEE, 2006, pp. 459–468.

[3] G. Salton and C. Buckley, "Improving retrieval performance by relevance feedback," *Journal of the American society for information science*, vol. 41, no. 4, pp. 288–297, 1990.

[4] M. Hasanuzzaman, V. Ampornaramveth, T. Zhang, M. Bhuiyan, Y. Shirai, and H. Ueno, "Real-time vision-based gesture recognition for human robot interaction," in *2004 IEEE International Conference on Robotics and Biomimetics*. IEEE, 2004, pp. 413–418.

[5] V. Sombandith, A. Walairacht, and S. Walairacht, "Hand gesture recognition for lao alphabet sign language using hog and correlation," in *2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. IEEE, 2017, pp. 649–651.

[6] G. Murthy and R. Jadon, "Hand gesture recognition using neural networks," in *2010 IEEE 2nd International Advance Computing Conference (IACC)*. IEEE, 2010, pp. 134–138.

[7] E. Sánchez-Nielsen, L. Antón-Canalís, and M. Hernández-Tejera, "Hand gesture recognition for human-machine interaction," 2004.

[8] F.-S. Chen, C.-M. Fu, and C.-L. Huang, "Hand gesture recognition using a real-time tracking method and hidden markov models," *Image and vision computing*, vol. 21, no. 8, pp. 745–758, 2003.

[9] J. R. New, E. Hasanbelliu, and M. Aguilar, "Facilitating user interaction with complex systems via hand gesture recognition," in *Proceedings of the 2003 Southeastern ACM Conference, Savannah, GA*, 2003.

[10] X. Shen, G. Hua, L. Williams, and Y. Wu, "Dynamic hand gesture recognition: An exemplar-based approach from motion divergence fields," *Image and Vision Computing*, vol. 30, no. 3, pp. 227–235, 2012.

[11] F. Flórez, J. M. García, J. García, and A. Hernández, "Hand gesture recognition following the dynamics of a topology-preserving network," in *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*. IEEE, 2002, pp. 318–323.

7 Appendix

In this section, we briefly describe the individual contributions of each member of our group. Every member had a very unique and distinct outlook of the tasks. Intense sessions of discussion lasting over hours resulted in few consensus over the exact procedure the team members would follow for the tasks. Every member had diverse and countering points towards the understanding, implications and meanings of the task description, the algorithms to use and the interpretation of results. Working together as a group required special efforts, since every task had subdivisions that could be parallelized but there was always an inter-dependency on each other to complete the previous tasks. This added a great deal of understanding and cooperation among the team members to work together and resolve conflicts and doubts in code implementations. Tasks were distributed to every member and git was used as version control to help mandate the process of cohesion.

1. Skanda Suresh - 100. Tasks - task3, task6
2. Trilok Kumar Tourani - 100. Tasks - task3, task6
3. Preethi Raman - 100. Tasks - task2-KNN and task4-vectorized query optimization, results collection and experiments
4. Samyuktha Sridhar - 100. Tasks - task2-Decision Tree, task4-relative importance computation, task6-system intelligence, results collection and experiments
5. Raghavendhar Thiruvoipadi - 100. Tasks - task2-PPR, results collection and experiments
6. Nithish Moudhgalya - 100. Tasks - task1, task4-probabilistic query optimization, task5, task6, combining task codes, results collection and experiments