Université d’Ottawa
Faculté de génie

uOttawa
L’Université canadienne
Canada’s university

University of Ottawa
Faculty of Engineering

Group Final Project
Q&A Mental Care Chatbot System

1 Group Member

He Zhixin

Student ID: 300142825

Responsibility: Feature Extraction, Clustering Error Analysis, Recommend Engine Development, Report Writing

Lin Wenqi

Student ID: 300201446

Responsibility: Chatbot GUI Development, Training Model Development, Parameter Optimization, PPT making

Pang Yuxue

Student ID: 300177911

Responsibility: Raw Data Processing, Classification Error Analysis, Confusion Matrix Design, Report Writing

Qin Kun

Student ID: 300188345

Responsibility: Classification Model Prediction, Confusion Matrix Optimization, Chatbot GUI Development, Code Organization

2 Problem Formulation

Mental health is arousing more and more attention these years. In Canada, approximately one in five people suffer from mental health illness. Nearly 50% of citizens from age 40 say that they had or are experiencing mental health problem. The most effective measure to cope with mental issue is to see psychologist. However, it is necessary for people to know self-diagnosis when it is hard to see the doctor at once. By teaching people some basic emotion evaluation, it is also helpful for doctor to quickly understand the current mental status of patients in depth. That is our purpose to design Q&A mental care chatbot. We believe that people will be better at protecting themselves after using the chatbot system. it will give patients a appropriate answer for each question and provide some similar questions for further asking.

3 Prepare the data and process the data:

Firstly, in order to get the original data, our group used mental health dataset provided by Kaggle, which has several csv files. Secondly, our goal is to generate 98 question label groups, each group has 5 similar questions.

To do this, we firstly import 98 questions and their corresponding answers. Then, we extend each question into a group by adding 4 another similar records. In all, the dataset has 98 question group with 490 records. Finally, we used the **word_tokenize** function to get the 490*485 target record words. We also clean the stops words by using stop words list generated before.

To easily process the data of all question records, we defined a function named **“csvDataProcess”** to do that, it has one parameter for verifying the specific name of csv file.

4 Transform

To deal with the records, our group used the Bag of Words (BOW), TF-IDF and LDA (Latent Dirichlet Allocation) methods to realize it. We plan to utilize 485 records to finish the transformation.

4.1 Feature words extraction

For the BOW method, we used **CountVectorizer()** to transform the words in the text in to the matrix of the frequency of words, and used **fit_transform** to calculate the frequency of each word. After transformation, **toarray** function was used to generate a numpy array.

Similar to BOW, in TF-IDF method, TfidfTransformer() class and its related functions were used to build to calculate the value of TF-IDF of each word. In addition, LDA was implemented from the library of sklearn.decomposition, and the parameters of this function was set as follows n_components=98, max_iter=50, learning_method='batch'.

4.2 Building labels for Question Groups

We build a numpy array to storage the labels of each question group, we use **range(98)** to list numbers from 0 to 97 represent for each book, and the length of each number is 5, representing 5 similar questions. So, the total length of this array is 98*5 = 490.

5 Machines Classification and Evaluations with SVM, KNN and DC

After section 4.1 and 4.2, we use train_test_split function to split data into training part and test part. Then training data and test was put into different machines (SVM, KNN and DC) to train and test.

5.1 Classification Algorithm Parameters Setting

The parameters of SVM, KNN and DC were set as below:

```
clf = svm.SVC(C=0.5, kernel='linear', decision_function_shape='ovo')
```

Fig 1. Parameters of SVM using BOW

```
clf = svm.SVC(kernel='linear', C=1)
```

Fig 2. Parameters of SVM using TF-IDF

```
KNN = KNeighborsClassifier(n_neighbors=1,algorithm='kd_tree',weights = 'distance')
```

Fig 3. Parameters of KNN using BOW

```
KNN = KNeighborsClassifier(n_neighbors=2,algorithm='auto',weights = 'distance', p = 3)
```

Fig 4. Parameters of KNN using TF-IDF

```
Tree = tree.DecisionTreeClassifier(splitter='best', criterion = 'gini')
```

Fig 5. Parameters of DC using BOW

```
Tree = tree.DecisionTreeClassifier(splitter='best')
```

Fig 6. Parameters of DC using TF-IDF

5.2 Evaluation for SVM, KNN and DC

5.2.1 Using accuracy to evaluate models

Accuracy was used to evaluate the performance of each machine, which is shown in the below table.

Table1 Evaluation using accuracy

Transform Method / Training Machine	BOW	TF-IDF
SVM	0.71	0.69
KNN	0.68	0.75
DC	0.66	0.63

For a further evaluation, we used five times Cross Validation to compare the models, the results are shown below, and we can see that SVM using TF-IDF shows the best accuracy this time, which is different from the above verification. Because cross-validation can bring more reliable results, we think that TF-IDF with SVM tends to have the best accuracy among all models.

```
scores of 5 times cross validation using BOW+DC
Cross-validation scores: [0.81632653 0.76530612 0.71428571 0.75510204 0.68367347]
mean of Cross-validation:
0.746938775510204
scores of 5 times cross validation using BOW+KNN
Cross-validation scores: [0.78571429 0.75510204 0.69387755 0.67346939 0.75510204]
mean of Cross-validation:
0.7326530612244898
scores of 5 times cross validation using BOW+SVM
Cross-validation scores: [0.80612245 0.76530612 0.76530612 0.79591837 0.81632653]
mean of Cross-validation:
0.789795918367347
scores of 5 times cross validation using TF_IDF+DC
Cross-validation scores: [0.75510204 0.75510204 0.69387755 0.67346939 0.68367347]
mean of Cross-validation:
0.7122448979591837
scores of 5 times cross validation using TF_IDF+KNN
Cross-validation scores: [0.78571429 0.76530612 0.70408163 0.74489796 0.76530612]
mean of Cross-validation:
0.753061224489796
scores of 5 times cross validation using TF_IDF+SVM
Cross-validation scores: [0.84693878 0.83673469 0.81632653 0.80612245 0.82653061]
mean of Cross-validation:
0.826530612244898
```

Fig 7. Five times cross validation of each model

5.2.2 Prediction of input data

First, we randomly chose a question in our question list, then it was transformed into an array of vector, and put it in the prediction of each model, as a result, every model can tell the answer of the question. The paragraph shown below is the test data and result:

"What is mental health?" =>We all have mental health which is made up of our beliefs, thoughts, feelings and behaviors.

We can see that the all the six machines can tell us the correct answer of the chose question, which means that our SVM, KNN and DC models are accurate enough to do this work.

6 Machines Clustering and Evaluations with K-Means, EM and AC

After section 4.1 and 4.2, the original input data was generated for those K-Means, EM and HC models. **However, when we were testing the EM algorithm, the RAM of the virtual machine of CoLab overloaded, due to the high dimensions of the input data.** Therefore, in order to make sure the program could run correctly, we decreased the dimension of the original data through the PCA method for TF-IDF and BOW.

6.1 Clustering Algorithm Parameters Setting

The parameters of K-Means, EM and AC functions were set as below:

```
kmeans  =  KMeans(n_clusters  =  5,  init='k-means++',  n_init=1,  max_iter=200,
                                 tol=0.0001,  precompute_distances=True,  verbose=0,  random_state=9,
                                 copy_x=True,  n_jobs=None,  algorithm='auto')
```

Fig 1. Parameters of K-Means

```
gmm = GaussianMixture(n_components=5, covariance_type='full', tol=0.0001, reg_covar=1e-06,
                      max_iter=300, n_init=200,weights_init=None, means_init=None,
                      precisions_init=None, random_state=None, warm_start=False,
                      verbose=0, verbose_interval=10)
```

Fig.2. Parameters of EM

```
#"euclidean", "l1", "l2", "manhattan", "cosine", or "precomputed"
#linkage : {"ward", "complete", "average", "single"},
ac = AgglomerativeClustering(n_clusters=15, affinity='cosine', linkage='average')
```

Fig.3. Parameters of AC

6.2 Evaluation for KM, EM and HC

For the evaluation of the clustering, Kappa, Consistency, Coherence and Silhouette were used in our program in order to compare the different clustering methods. The result of these evaluation methods was shown below.

Kmeans

	0
kappa	0.041237
consistency	0.182656
silScore	0.066978
chScore	3.206853

Fig.3. The Evaluation of Kappa, Consistency, Coherence and Silhouette with the BOW-KMeans

GaussianMixture

	0
kappa	0.041237
consistency	0.284635
silScore	0.066765
chScore	3.178924

Fig.4. The Evaluation of Kappa, Consistency, Coherence and Silhouette with the BOW-GM

AgglomerativeClustering

	0
kappa	0.047423
consistency	0.306309
silScore	0.017640
chScore	2.590824

Fig.5. The Evaluation of Kappa, Consistency, Coherence and Silhouette with the BOW-AC

Kmeans

	0
kappa	0.041237
consistency	0.249760
silScore	0.030693
chScore	3.953466

Fig.6. The Evaluation of Kappa, Consistency, Coherence and Silhouette with the TF-IDF-KMeans

GaussianMixture

	0
kappa	0.041237
consistency	0.270914
silScore	0.029805
chScore	5.206064

Fig.7. The Evaluation of Kappa, Consistency, Coherence and Silhouette with the TF-IDF-GM

AgglomerativeClustering

	0
kappa	0.051546
consistency	0.276610
silScore	0.052813
chScore	3.191267

Fig.8. The Evaluation of Kappa, Consistency, Coherence and Silhouette with the TF-IDF-AC

From the picture shown above, we can see that the TF-IDF has the overall better clustering with K-Means, EM and AC than BOW. However, the general evaluation result is not very good. For example, the kappa always keeps in a very low level. We suppose that it should be related to the dataset our choose. The dataset does not have a very big volume. During the clustering, the machine does not have sufficient resources to finish the training. This might cause that the accuracy is always in a low level. Another reason is that our dataset focuses on the mental health. There is no doubt that each question has a limited correlation with others. After data cleaning, it is possible that some questions have similar feature words but asking in different aspects. When the machine confronts this situation, it is likely to regard them as similar group. Therefore, the evaluation result naturally cannot meet our expectations.

7 Error analyzation

批注 [DH1]: 钰雪可以从这里开始

7.1 High frequency words

To realize this part, first, we used the test set as the input of the **error_analysis** function, then the machine predicts answers about input questions through the clustering algorithms. If the predicted answers do not match the actual questions. This answer is a wrongly predicted answer, and then the word frequency of the feature words is calculated for this wrong answer. Then the top 10 most frequent feature words were found, which means these feature words are easily misjudged by machines.

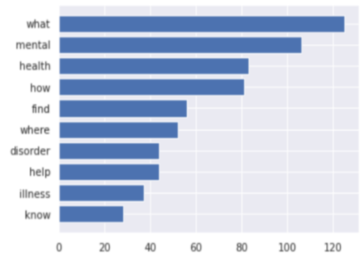

Fig 16. Top 10 frequency words through BOW+KMeans

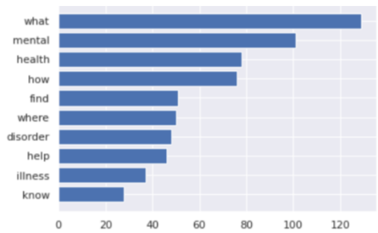

Fig 17. Top 10 frequency words through BOW+EM

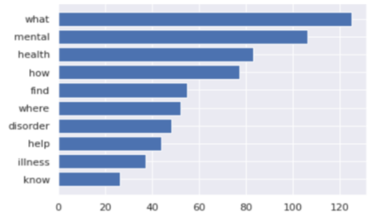

Fig 18. Top 10 frequency words through BOW+HC

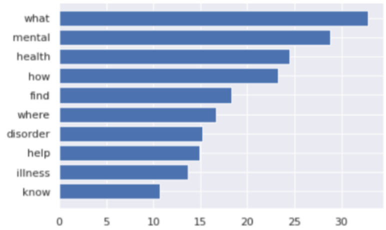

Fig 19. Top 10 frequency words through BOW+KMeans

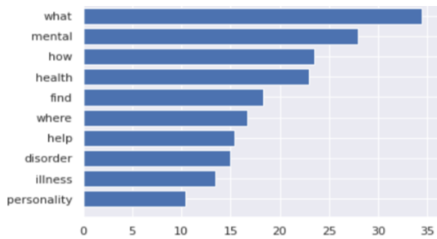

Fig 20. Top 10 frequency words through BOW+EM

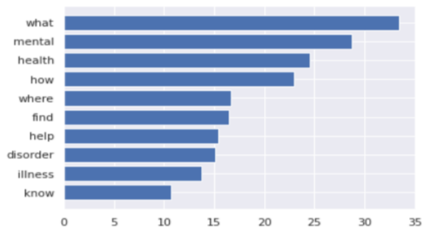

Fig 21. Top 10 frequency words through BOW+HC

Like the picture shown above, some of the words are pretty common when we used the BOW transform, like "what", "mental", "health" and etc., which we thought these words do not have the particular characteristic for all those questions, and we consider those words should also be stop-words, then we filtered those word, and redefine them as the new stop-words in addition to the stop-words form the NLTK library. After the filtering work, we run the clustering model again, in order to compare the "common words" effect.

As we can see from the below graphs, after refreshing the stop-words, the results of Kappa and Consistency increased dramatically. Which we can draw a conclusion: Removing the high-frequency words (which are also unnecessary words) from the original data, it will improve the clustering results in the BOW model, while in the TF IDF model, after changing the stop words, the evaluation results did not change dramatically, this maybe due to the TF IDF already had excellent clustering result. However, the optimization space is limited. For the LDA model, we can see that the redefine stop word has positive impacts on the kappa coefficient as well as consistency just like the BOW model. However, it will lower the silScore and chScore slightly.

	kappa	consistency	silScore	chScore
KM	0.60250	0.573663	0.015606	4.982611
EM	0.67500	0.644175	0.010780	6.551854
HC	0.62875	0.649735	0.007445	5.439557

Fig 22 Evaluation results before redefine stop word in bow model

	kappa	consistency	silScore	chScore
KM	0.68000	0.724253	0.015698	5.350281
EM	0.92750	0.845246	0.011236	6.337511
HC	0.67625	0.759087	0.008573	5.524254

Fig 22 Evaluation results after redefine stop word in bow model

	kappa	consistency	silScore	chScore
KM	0.96375	0.918709	0.016861	8.136206
EM	0.96375	0.912849	0.017031	8.110744
HC	0.92625	0.859386	0.015890	8.361653

Fig 23 Evaluation results before redefine stop word tf idf model

	kappa	consistency	silScore	chScore
KM	0.95000	0.899207	0.016884	8.159752
EM	0.96000	0.909562	0.017051	8.115528
HC	0.94625	0.889184	0.016401	8.298426

Fig 24 Evaluation results after redefine stop word tf idf model

7.2 Confusion Matrix

After Clustering, our group also used the Confusion Matrix to help us find the relationships among different models.

TF-IDF analyzation:

For the TF-IDF + KMeans model, we can see that, the machine have some problem about distinguishing author among author 3 and author 0 , as well as author 4 and author 0, which means the “melville-moby dick” may have some similarity between ‘chesterton-ball' and 'bible-kjv', this situation also occurred in EM and HC models, while in the HC mode, some chunks were misjudged from ‘melville-moby dick’ to 'edgeworth-parents'. So, it’s easy to see that the machine have difficulty in judge ‘melville-moby dick’ and other books.

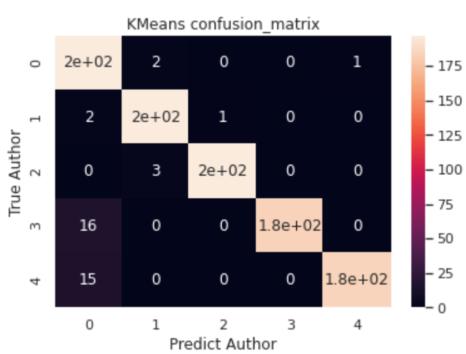

Fig 34 Confusion matrix of TF-IDF+ KMeans

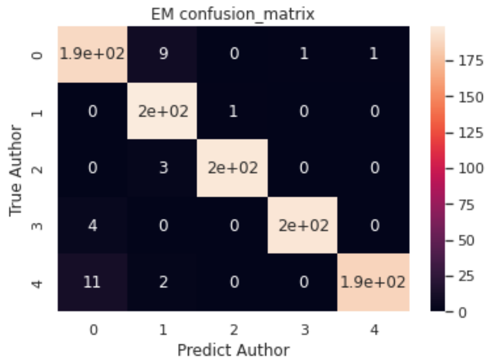

Fig 35 Confusion matrix of TF-IDF+ EM

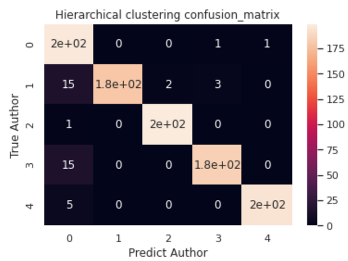

Fig 36 Confusion matrix of TF-IDF+ HC

BOW analyzation:

We can see that when the machine used K-Means clustering, it failed to distinguish author 2. For author 3, the phenomenon is similar with TF-IDF transformation, some chunks were considered as author 0. In the EM model, author 0 and author 4 has relatively good clustering results, compared with author 2 and 3. And the machine has difficulty in distinguishing author 1 and 3.

For the HC model, the result is not good, we can see that the machine thought the true author1 and author 3 were author 0.

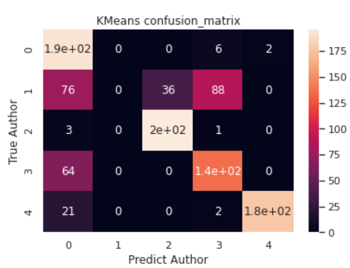

Fig 37 Confusion matrix of BOW +KMeans

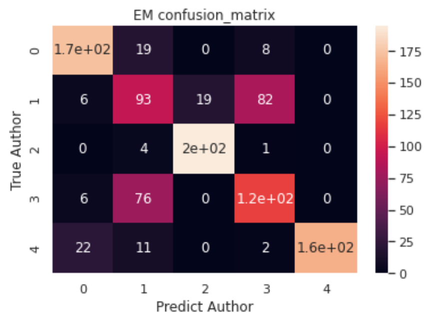

Fig 38 Confusion matrix of BOW+EM

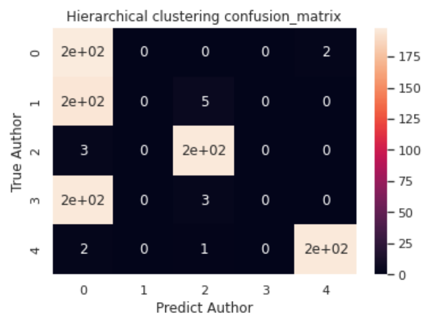

Fig 39 Confusion matrix of BOW+HC

LDA analyzation:
Different from former analyzation, through the K-Means, EM and HC model, the machines cannot precisely judge the author 0 and author 1, one of the reason we think is that during the transformation progress, we need input the n_components, which has already decreased the dimension of the original data, it may lost some valid data . This is different from BOW and TF-IDF,

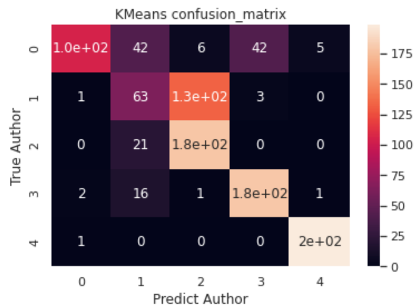

Fig 40 Confusion matrix of LDA+KMeans

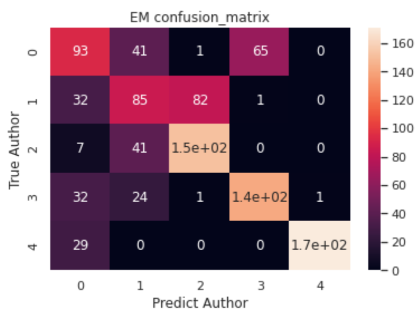

Fig 41 Confusion matrix of LDA +EM

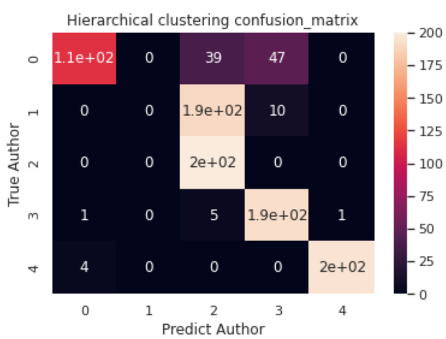

Fig 42 Confusion matrix of LDA+ HC

8 Recommend Engine

From all classification and clustering models, we know that SVM with TF-IDF can provide results with the best accuracy, so in this part, we use confusion matrix generated in the error analysis part as input data of recommend engine. By choosing predicted questions which similarity is higher or equals to 1, we can generate a file named SVM_confusion.csv which have three columns named ‘trueQuestion’ ‘predictQuestion’ and ‘similarity’ separately.

After that we imported relevant algorithms such as SVD and NMF to build recommend engine, the table below shows some running parameters of different algorithms.

Table 2. running parameters of different recommend algorithms

Algorithm	test_rmse	fit_time	test_time
NMF	0.119335	0.364949	0.018368
KNNBasic	0.121338	0.008475	0.358545
SVD	0.130906	0.372379	0.022844
NormalPredictor	0.145139	0.008424	0.020608

Then a function called svm_prediciton was built to process raw input question and its main work is to give a predict question by using SVM with TF-IDF model to recommend engine. After recommend engine receives input question, it will output five recommend questions according to our similarity question list and its own algorithm. To evaluate the engine,we randomly chose a question called “What is mental illness”, and its 5 recommend questions can be seen as below:

'Who can be influenced by mental illness',
'What increases the possibility of mental illness',
'What may make mental illness happen',
'Give me some information about different mental health professionals',
'How can I find a mental health expert for my friends'

From the above test results, we can see that the engine tells us five relevant questions and they are all related to the actual question which means that the recommend engine is accurate enough to do this work.

9.GUI Front End System Development

In this part, our group used the ***anvil*** online platform to realize the GUI development. Anvil is a platform that can be connect with google colab to do amazing front end display. After the programming as well as the web page design of the anvil, this platform can generate an authoration code, which can help the colab connect with anvil.

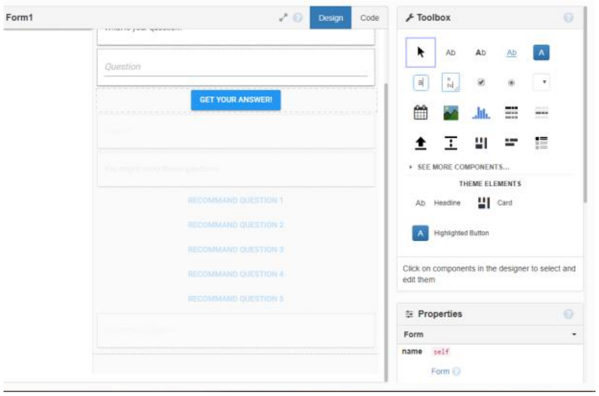

Fig.43 Web page design of Anvil

```python
def Question_button_click(self, **event_args):
  """This method is called when the button is clicked"""
  Mental_chatrobot_index,Mental_chatrobot = anvil.server.call('svm_prediction',self.Question.text)

  # If a Answer is returned set our species
  if Mental_chatrobot:
    self.answer_card.visible = True
    self.Answer.visible = True
    self.Answer.text = Mental_chatrobot
  Recommend_question =  anvil.server.call('get_predictions',Mental_chatrobot_index)
  if Recommend_question:
    self.tip_card.visible = True
    self.tip.visible =True
    #self.recommand_card.visible= True
    #self.label_2.visible =True
    #self.Recommand.visible = True
#      self.Recommand.text = Recommend_question[0]
#      self.Recommand1.text = Recommend_question[1]
#      self.Recommand2.text = Recommend_question[2]
#      self.Recommand3.text = Recommend_question[3]
#      self.Recommand4.text = Recommend_question[4]
    self.Recommand_button0.visible = True
    self.Recommand_button1.visible = True
    self.Recommand_button2.visible = True
    self.Recommand_button3.visible = True
    self.Recommand_button4.visible = True
    self.Recommand_button0.text = Recommend_question[0]
    self.Recommand_button1.text = Recommend_question[1]
    self.Recommand_button2.text = Recommend_question[2]
    self.Recommand_button3.text = Recommend_question[3]
    self.Recommand_button4.text = Recommend_question[4]
  pass
```

Fig.44 Anvil web page Python programming

Then, whatever function we want to call in the colab, the only thing we need to do is add this code top of the function ***@anvil.server.callable*** to further develop it in the anvil we used the function ***"anvil.sever.call"*** to call the colab function.

SVM Call function

```python
!pip install anvil-uplink
import anvil.server
anvil.server.connect("DNOXCO5CLUGFCVMED5SR6P5H-QOSXMM7EFF76S4TW")
```

```python
@anvil.server.callable
def svm_prediction(question):
    Predict_part=docs_new = [question]
    X_new_counts = bow_vect.transform(Predict_part)
    Predict_counts = tfidf_transformer.transform(X_new_counts)
    Predict_input = Predict_counts.toarray()
    Predict_result = clf.predict(Predict_input)
    return Predict_result[0]+1,answers[Predict_result[0]]
```

```python
@anvil.server.callable
def get_predictions(trueQuestion):
        results = tmp_transpose.loc[trueQuestion]
        recommended_question_ids=[]
        for x in range(0, n):
            recommended_question_ids.append(results[x][0])
            recommended_questions=[]
        for question_id in recommended_question_ids:
            recommended_questions.append(all_questions[question_id])
        recommended_questions
        return recommended_questions
```

Fig.45 Colab Connects with Anvil

When user inputs questions, the Anvil will read the question and send it as the local varible to the colab prediction function, then the predicion function will return the answer index and the anwser text, the anvil will first display the answer, after that, it will send the index number to the recommend engine(in colab), then the engine will return 5 reommended questions and display them with the format of 5 buttons, if the user clicks the one of the buttons, the anvil will call the prediction function again, and display the recmmendation answer.

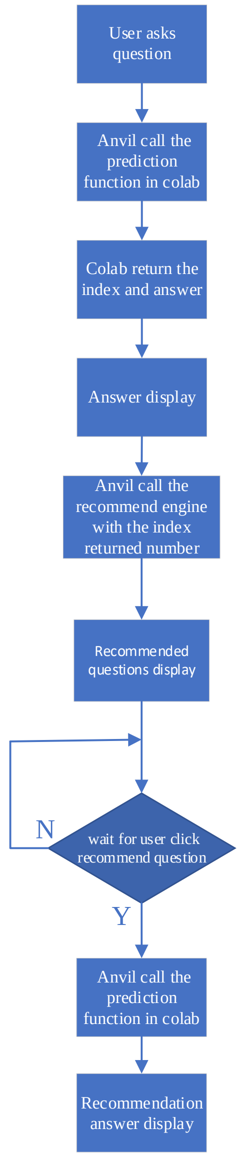

Fig. 46 Flow chart of the GUI system

The final GUI front end is like this:

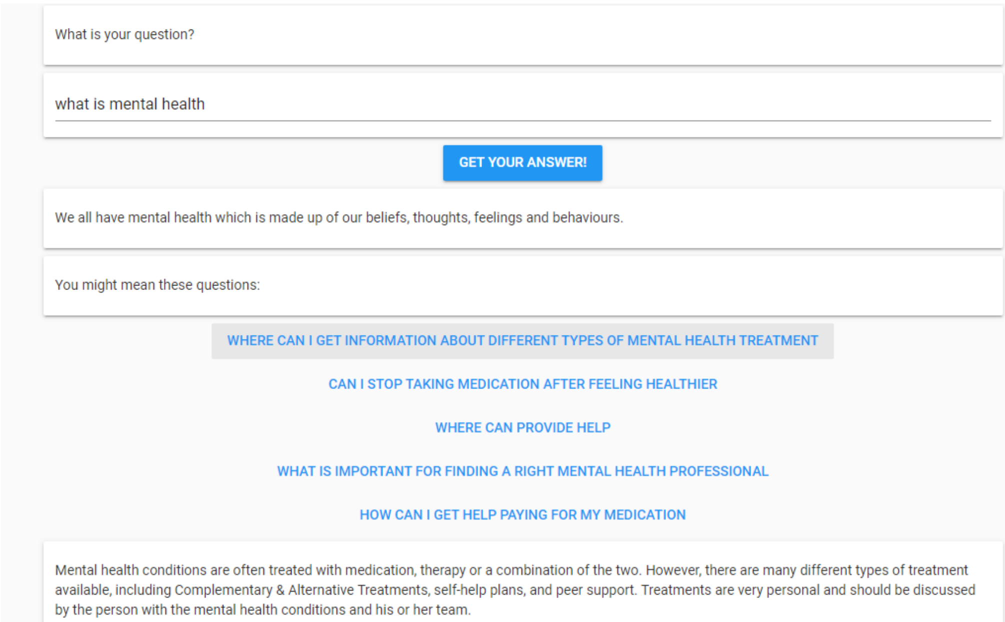

Fig.47 GUI system display

10 Conclusion

From the former chapters, the conclusion can be drawn as follows:

(1) The accuracy of the clustering results will be lower as the dimension of the input data decreased.

(2) After adding the top10 high frequency words into the stop words, the accuracy of BOW and LDA transformation will improve dramatically.

(3) Among all the machine training models, the TF-IDF transformation has the best clustering result.

(4) Different parameters of the clustering algorithms' function will have different result, after several times parameters optimization, we get the results which have been above. Due to the large amount of parameters, we did not shown in our paper.