A REPORT

ON

**ONLINE QUESTION ANSWERING SYSTEM**

BY

AKSHAT SINGH 2019A7PS0074P COMPUTER SCIENCE

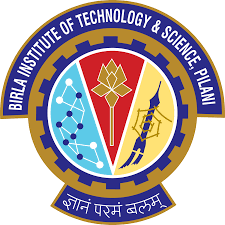
**STUDY ORIENTED PROJECT**

Under the guidance of

**Prof L. RAJYALAKSHMI**

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI**

**JANUARY 2022**

****

**ACKNOWLEDGEMENTS**

I would like to thank AUGSD for providing me with this opportunity to be able to work under seasoned and esteemed professors of BITS Pilani and gain immeasurable knowledge and experience from the same.

I would like to express my very great gratitude to Prof L. Rajyalakshmi for her valuable guidance and constructive suggestions during the planning as well as the development of this project. Her willingness to give her time so generously has been very much appreciated.

I would also like to thank my friends for being incredibly supportive and helpful. Their valuable inputs have been immensely helpful in the project.

**ABSTRACT**

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)**

**Dept of Computer Science and Information Systems**

**Duration:** 1 semester **Date of Start:** 15th January 2022

**Date of Submission:** 16th May 2022

**Title of the Project:** ONLINE QUESTION ANSWERING SYSTEM

**ID No./Name:**  2019A7PS0074P / Akshat Singh

**Discipline of the student:** Computer Science

**Name of the Faculty:** L Rajyalakshmi

**Key Words:** Computer Science, Machine Learning, Natural Language Processing, Community QnA, Efficiency, Verify Result, Algorithms

**Project Areas:** Machine Learning

**Abstract:** Compute and verify the effectiveness of different algorithms that can be used to implement a community QnA system, and return the best result possible.

**TABLE OF CONTENTS**

1. Title Page
2. Acknowledgements
3. Abstract Sheet
4. Table of Contents
5. Introduction
6. Literature Review
7. Workflow
8. Results
9. Conclusions
10. Appendices
11. References
12. **INTRODUCTION**

Over the past few years, the acceptance of online answering services has skyrocketed. With the advent and popularity of sites like Yahoo! Solutions, Cross Validated, Stack Overflow, Quora, and Health Tap, more and more people are turning to these online forums to answer their questions. Individuals can use these forums to post questions online and ask a variety of experts from around the world to respond, and have the opportunity to provide thoughts or expertise to help. This, in turn, fosters more engagement and gratitude. [[1]](https://www.analyticsvidhya.com/blog/2021/11/end-to-end-question-answering-system-using-nlp-and-squad-dataset/) The most common type of knowledge retrieval is the question-and-answer system, which is recognized by the user's information requests expressed in natural language sentences or queries and is a type of natural interaction between humans and machines. In question answers, where the whole thing is considered relevant to the requested data, specific pieces of data appear to be returned in response to a query. The user of the question-and-answer system must find a concise and understandable answer with an appropriate answer, which can refer to a paragraph, a word, a picture, a sentence, a paragraph, audio, or an entire document.

Today, online social networks are growing rapidly, providing many options for expressing thoughts in natural language. Question Answering System (QAS) is a web forum that allows users and experts to ask and answer questions in natural language. The basic goal of a question-answer system is to find the correct answer that satisfies the user for a given query on the web and a collection of documents. There are several ways to find the right answer to the question asked, but the public question-answering services available today provide large lists of articles that may be relevant without specifying the importance of the question. This system results in a quick response to a specific request. Therefore, one of the most important tasks for consumers or knowledge users is to identify correct response data, which involves getting direct, specific, and relevant responses to a query. Giving clear and precise answers is important for having a good arrangement of relevant knowledge.

When we try to find answers to queries in the community question and answer system, we use repositories where we can find them on a theoretical basis. However, finding questions and where they might relate to different answers can be time-consuming, and finding related answers requires sifting through a large number of responses.

It is essential to have a clear answer to a particular query that is both relevant and updated. Furthermore, since information seekers have to wait a long time to receive responses from other users, discovering comparable questions and answers in earlier texts helps. They save time. Community Question Answering (CQA) Forums are a type of social network that provides an alternative to traditional web searches. Instead of retrieving results from a web search network, forum users post desired information as valid natural language queries and received direct answers written by people or experts. Natural language material is available in a variety of quality categories, such as questions and answers, from high-quality content to low-quality content, irrelevant content, and even derogatory information. As a result, voting for the best answer becomes more complicated, and choosing the answer with the highest quality becomes more important.

1. **LITERATURE REVIEW**

The Bloom filter includes an algorithm that has a spatial advantage over many different types of data structures such as binary trees, binary search trees, hash tables, arrays, and linked lists. [[2]](https://www.educative.io/edpresso/what-is-a-bloom-filter) These data structures can hold a large amount of data, from bits to integers. They can even contain lists of arrays or arrays of arrays, essentially indicating that they are compatible with each other. So, depending on the situation, an array consisting of other data structures can be used to save space and improve efficiency.

The Onion Routing Protocol is a technology that allows users to communicate totally anonymously over the internet. [[3]](https://www.researchgate.net/publication/221615105_Routing_questions_to_appropriate_answerers_in_Community_Question_Answering_services) This is accomplished using layer-by-layer encryption. It's termed an onion routing method because messages are encased in multiple layers, one on top of the other, just like an onion. This data is sent through routers, which split all of the messages layer by layer and then reassemble them for decryption once they arrive at their destination. The users' identities are still hidden because just the message's location and address are accessible, and only for the most recent message layer. This aids in the better analysis of time.

[[4]](https://arxiv.org/abs/1408.5882) proposes the use of a Convolution Neural Network (CNN) to encode sentences. The initial strategy targeted for deep learning development is word representation. This technique is known as word embedding. It is insufficient for addressing questions. Another method for language modeling has been proposed: the neural sentence model. It is used to create a fixed-length vector from a variable-length phrase. The author has studied the incorporation of Word, convolution, and MaxPooling layers into this design.

The author proposes a response sequence learning model in [[5]](https://www.researchgate.net/publication/314198465_Answer_Selection_in_Community_Question_Answering_via_Attentive_Neural_Networks) by combining a cumulative neural network (CNN) and long short term memory units (LSTM) for response selection. Based on a cyclic neural network, this approach can represent a question-and-answer grammar objective (RNN). This approach is used in the AQC program to improve the performance of response selection. After using CNN to link the question-and-answer representatives, the author used LSTM to classify the similarity quality of each answer.

1. **WORKFLOW**

In this project, we try to verify and improve the working of one such existing model. This model uses the Dice metric and a modified version of the Levenshtein Distance formula to compare question subjects. [6] This model is compared with other models that use KNN classifier, Naïve Bayes, etc. [7] Their performance is evaluated, and results are compared.

The dataset used for implementing these models consists of 2000 question and answer pairs from Yahoo! Answers between 2009 and 2010. [[8]](https://sourceforge.net/projects/yahoodataset/) The QA pairs consist of questions and answers from different categories like Facebook, Google, Youtube, Wikipedia, etc.

The metrics that are used to measure results are Precision, Recall, and F-measure.

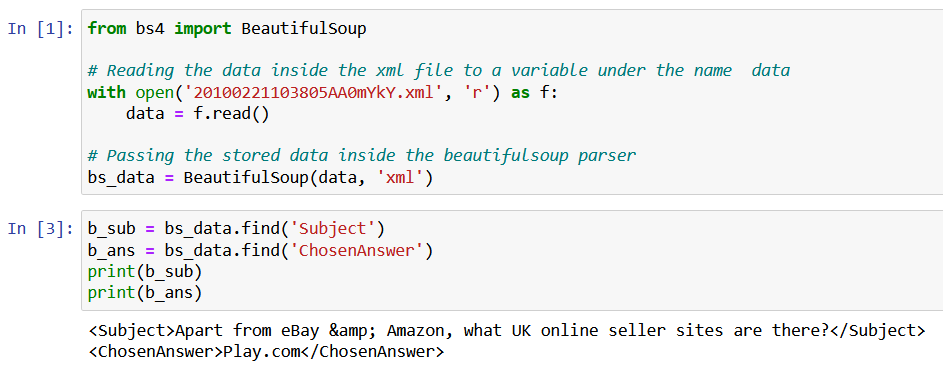
Recall = = *P()*

Precision = = *P()*

F-measure =

* 1. **Extracting QA pairs**

The dataset comprises of XML files. We used a python library called BeautifulSoup to extract QA pairs from these files. The code is provided in Appendix4.



We can then move on to implement the models on them.

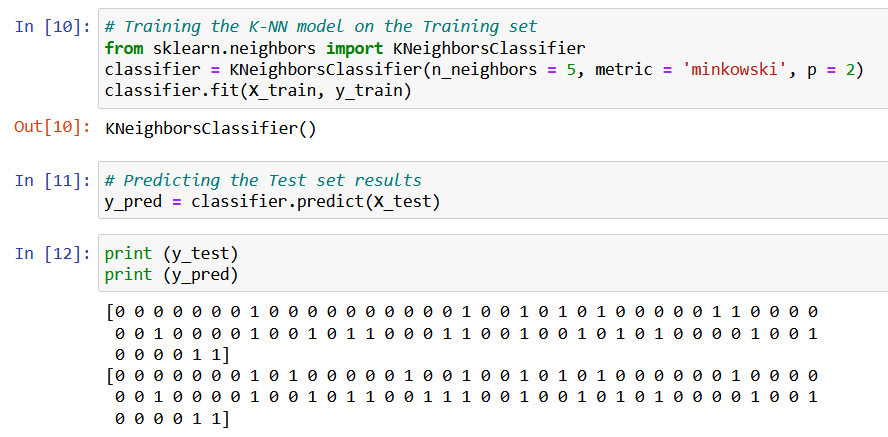
* 1. **K-nearest Neighbours**

The k- nearest neighbors (KNN) algorithm is a supervised machine learning algorithm that can be used to solve classification and regression problems.

KNN captures the idea of similarity and closeness to calculate the distance between points on a graph. [[9]](https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761)

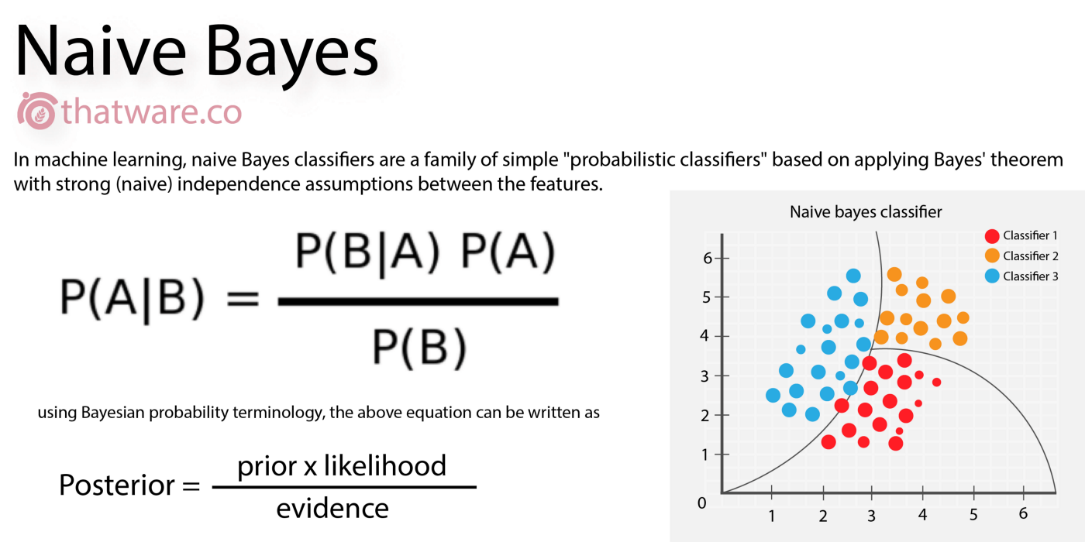


The implementation process begins with loading the data and initializing k to the desired number of neighbors. Then we apply the classifier to make a prediction as y\_pred. The actual result expected is stored as y\_test. We compare y\_pred and y\_test to achieve the results. The code is provided in Appendix1.



* 1. **Naïve Bayes**

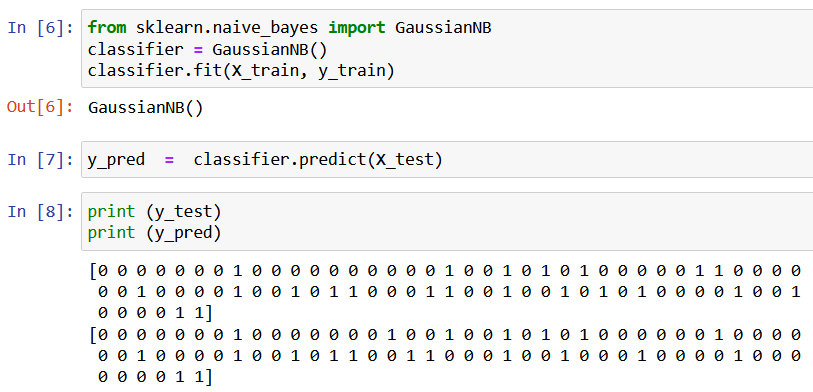
A Naïve Bayes classifier is a machine learning model that's used for classification task. The core of this classifier is based on the Bayes theorem in probability.



Bayes theorem: P(A|B) =

It is used to find the probability of A happening, given that B has occurred. Here, B is the evidence, and A is the hypothesis. The classifier is called 'Naïve' Bayes because here, an assumption is made that one particular feature does not affect the other. [[10]](https://towardsdatascience.com/introduction-to-na%C3%AFve-bayes-classifier-fa59e3e24aaf)

The implementation process begins with loading the data and calculating the gaussian distance. Then we apply the classifier to make a prediction as y\_pred. The actual result expected is stored as y\_test. We compare y\_pred and y\_test to achieve the results. The code is provided in Appendix2.



* 1. **Dice Metric and Modified Levenshtein Distance**

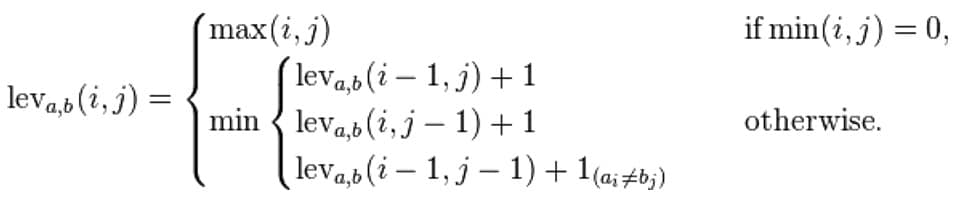
The Dice similarity coefficient, also known as Sorensen -Dice index, is a tool to measure the similarity between two sets of data. The equation for calculating similarity is:

2\* | X ∩ Y | / (| X | + | Y |)

Where X and Y are the two sets.

The Levenshtein distance is a string metric for measuring the difference between two sequences. [[11]](https://www.cuelogic.com/blog/the-levenshtein-algorithm#:~:text=The%20Levenshtein%20distance%20is%20a,one%20word%20into%20the%20other) Informally, it is the minimum number of single-character edits required to change one word into the other.

Mathematically,

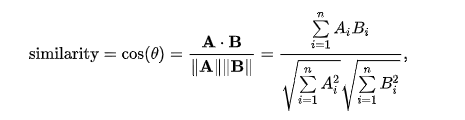


We apply our modified Levenshtein Distance formula to question statements from the database. Our algorithm evaluates the similarity of sentences compared to the original algorithm used to compare words. The code is provided in Appendix3.



* 1. **Cosine Similarity**

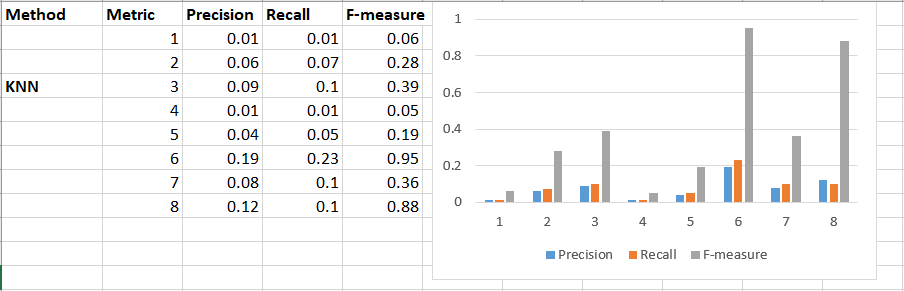
Cosine similarity is a measurement that quantifies the similarity between two or more vectors. It is described mathematically as the division between the dot product of vectors and the product of the Euclidean norms or magnitude of each vector. [[12]](https://towardsdatascience.com/understanding-cosine-similarity-and-its-application-fd42f585296a)



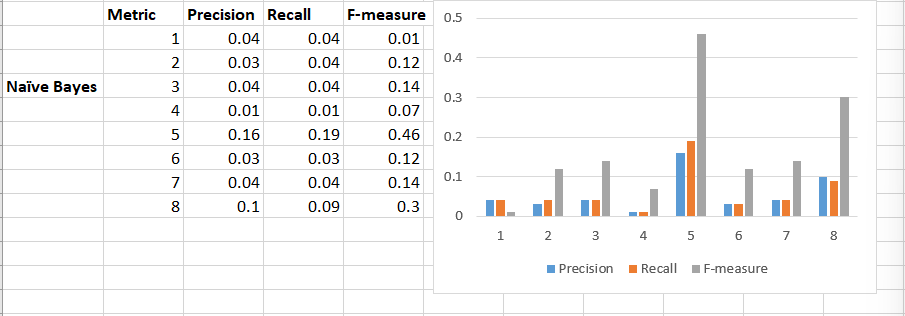
1. **RESULTS**

All of these methods were implemented on the dataset in different combinations, and the observations were recorded.

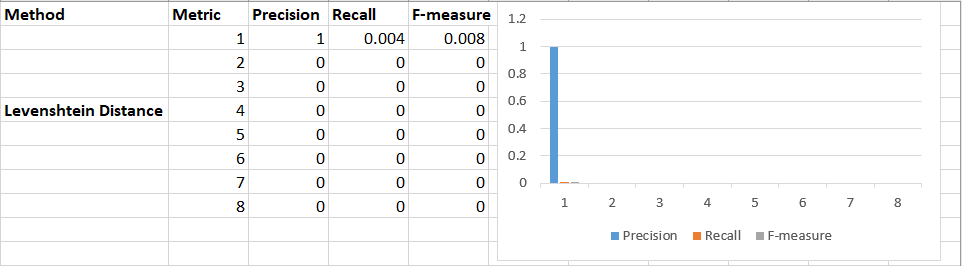
* 1. **Testing on individual databases prior to official evaluation –**
     1. **KNN classifier**



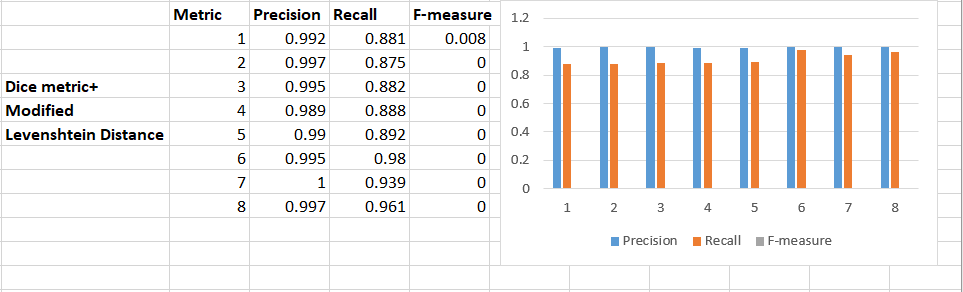
* + 1. **Naïve Bayes**



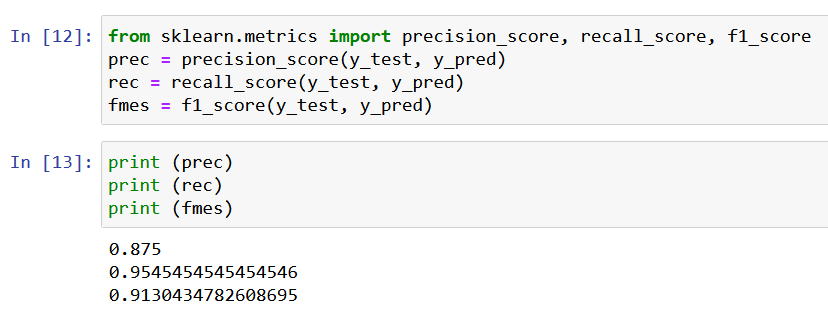
* + 1. **Unmodified Levenshtein Distance**

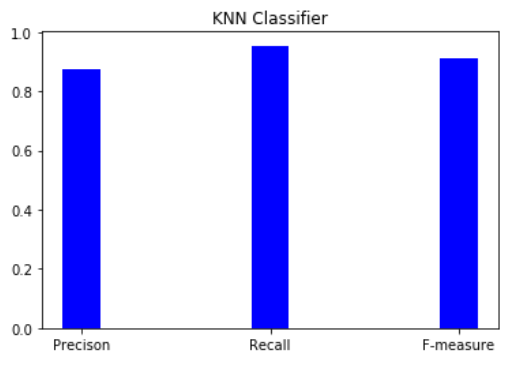


* + 1. **Dice Metric + Modified Levenshtein Distance**

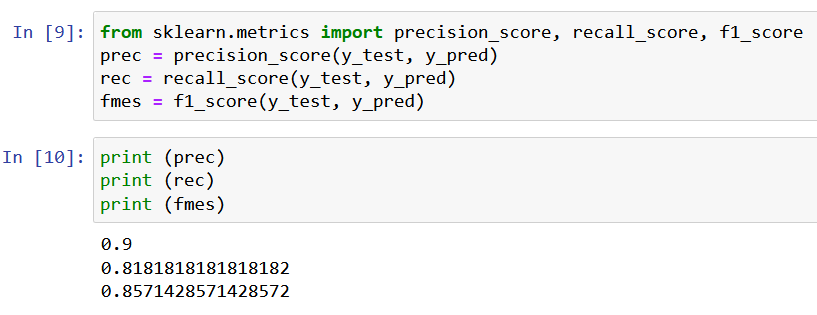


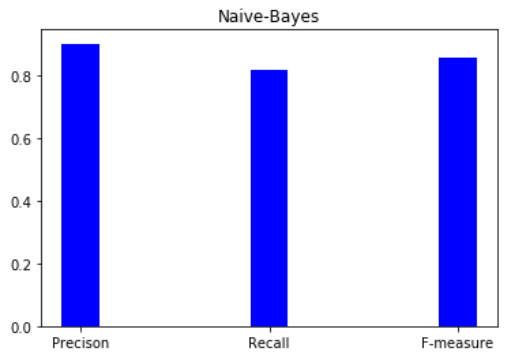
* 1. **Results after implementing on QnA database –** 
     1. **KNN classifier**



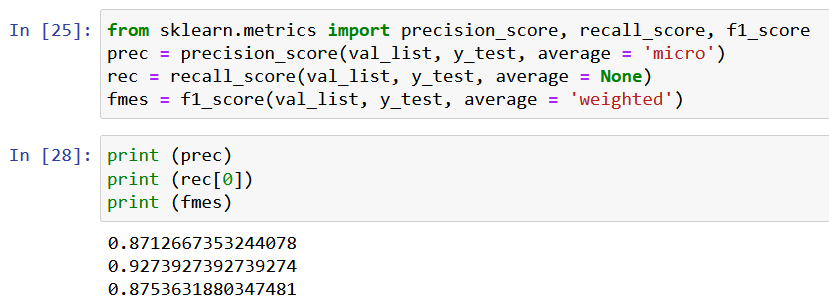


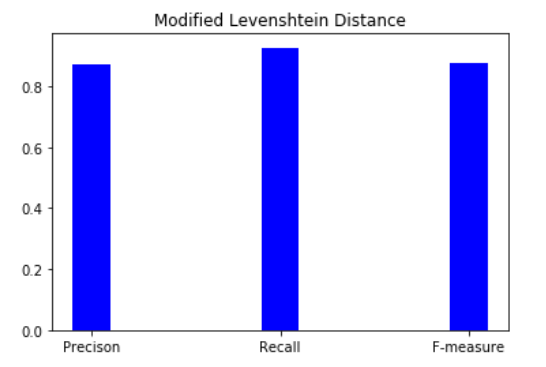
* + 1. **Naïve Bayes**





* + 1. **Modified Levenshtein Formula**





1. **CONCLUSION**

In this report, we have discussed and reviewed different models that can be implemented in Online Question Answering systems to increase their effectiveness and decrease the time lag between the users asking questions and receiving answers. The proposed methods have been properly evaluated, and the results have been appropriately documented to conclude with substantial proof of the efficiency of respective models.

1. **APPENDICES**
   1. **APPENDIX 1**

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import sklearn

# Importing the dataset

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

X = dataset.iloc[:, [1, 2, 3]].values

y = dataset.iloc[:, -1].values

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X[:,0] = le.fit\_transform(X[:,0])

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20, random\_state = 0)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Training the K-NN model on the Training set

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2)

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

from sklearn.metrics import precision\_score, recall\_score, f1\_score

prec = precision\_score(y\_test, y\_pred)

rec = recall\_score(y\_test, y\_pred)

fmes = f1\_score(y\_test, y\_pred)

print (prec)

print (rec)

print (fmes)

x = [1,2,3]

y = [prec, rec, fmes]

label = ['Precison', 'Recall', 'F-measure']

plt.bar(x,y, tick\_label=label,width = 0.2, color= 'blue')

plt.title('KNN Classifier')

plt.show()

* 1. **APPENDIX 2**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import sklearn

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

X = dataset.iloc[:, [1, 2, 3]].values

y = dataset.iloc[:, -1].values

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

X[:,0] = le.fit\_transform(X[:,0])

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20, random\_state = 0)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

from sklearn.metrics import precision\_score, recall\_score, f1\_score

prec = precision\_score(y\_test, y\_pred)

rec = recall\_score(y\_test, y\_pred)

fmes = f1\_score(y\_test, y\_pred)

from sklearn.metrics import precision\_score, recall\_score, f1\_score

prec = precision\_score(y\_test, y\_pred)

rec = recall\_score(y\_test, y\_pred)

fmes = f1\_score(y\_test, y\_pred)

print (prec)

print (rec)

print (fmes)

x = [1,2,3]

y = [prec, rec, fmes]

label = ['Precison', 'Recall', 'F-measure']

plt.bar(x,y, tick\_label=label,width = 0.2, color= 'blue')

plt.title('Naive-Bayes')

plt.show()

* 1. **APPENDIX 3**

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import sklearn

# Importing the dataset

dataset = pd.read\_csv('youtube.csv')

X = dataset.iloc[:, [1, 2]].values

y = dataset.iloc[:, -1].values

test\_dataset = pd.read\_csv('test.csv')

y\_test = test\_dataset.iloc[:, 1].values

y\_test = y\_test.flatten()

def printDistances(distances, token1Length, token2Length):

for t1 in range(token1Length + 1):

for t2 in range(token2Length + 1):

print(int(distances[t1][t2]), end=" ")

print()

def levenshteinDistanceDP(token1, token2):

if token1[-1] == '?' or '!' or '.':

token1=token1.rstrip(token1[-1])

if token2[-1] == '?' or '!' or '.':

token2=token2.rstrip(token2[-1])

sent1 = token1.split()

sent2 = token2.split()

distances = np.zeros((len(sent1) + 1, len(sent2) + 1))

for t1 in range(len(sent1) + 1):

distances[t1][0] = t1

for t2 in range(len(sent2) + 1):

distances[0][t2] = t2

a = 0

b = 0

c = 0

for t1 in range(1, len(sent1) + 1):

for t2 in range(1, len(sent2) + 1):

if (sent1[t1-1] == sent2[t2-1]):

distances[t1][t2] = distances[t1 - 1][t2 - 1]

else:

a = distances[t1][t2 - 1]

b = distances[t1 - 1][t2]

c = distances[t1 - 1][t2 - 1]

if (a <= b and a <= c):

distances[t1][t2] = a + 1

elif (b <= a and b <= c):

distances[t1][t2] = b + 1

else:

distances[t1][t2] = c + 1

return (int(distances[len(sent1)][len(sent2)]))

dict={}

for ind,tind in zip(dataset.index, test\_dataset.index):

dict.update({dataset['title'][ind] : levenshteinDistanceDP (dataset['title'][ind],test\_dataset['title'][tind])})

key\_list = list(dict.keys())

val\_list = list(dict.values())

from sklearn.metrics import precision\_score, recall\_score, f1\_score

prec = precision\_score(val\_list, y\_test, average = 'micro')

rec = recall\_score(val\_list, y\_test, average = None)

fmes = f1\_score(val\_list, y\_test, average = 'weighted')

print (prec)

print (rec[0])

print (fmes)

x = [1,2,3]

y = [prec, rec[0], fmes]

label = ['Precison', 'Recall', 'F-measure']

plt.bar(x,y, tick\_label=label,width = 0.2, color= 'blue')

plt.title('Modified Levenshtein Distance')

plt.show()

* 1. **APPENDIX 4**

from bs4 import BeautifulSoup

# Reading the data inside the xml file to a variable under the name data

with open('20100221103805AA0mYkY.xml', 'r') as f:

data = f.read()

# Passing the stored data inside the beautifulsoup parser

bs\_data = BeautifulSoup(data, 'xml')

b\_sub = bs\_data.find('Subject')

b\_ans = bs\_data.find('ChosenAnswer')

print(b\_sub)

print(b\_ans)

1. **REFERENCES**

[1] – <https://www.analyticsvidhya.com/blog/2021/11/end-to-end-question-answering-system-using-nlp-and-squad-dataset/>

[2] – <https://www.educative.io/edpresso/what-is-a-bloom-filter>

[3] – B. Li and I. King. “Routing Questions to Appropriate Answers in Community Question Answering Services”

<https://www.researchgate.net/publication/221615105_Routing_questions_to_appropriate_answerers_in_Community_Question_Answering_services>

[4] – Y. Kim, "Convolutional neural networks for sentence classification,"

<https://arxiv.org/abs/1408.5882>

[5] – X. Zhou, B. Hu, Q. Chen, B. Tang, and X. Wang, "Answer sequence learning with neural networks for answer selection in community question answering,"

<https://www.researchgate.net/publication/314198465_Answer_Selection_in_Community_Question_Answering_via_Attentive_Neural_Networks>

[6] – Bolanle Ojokoh, Peter Ayokunle “Online Question Answering System”

[7] – Juee Gosavi, B. N. Jagdale “Answer Selection In Community Question

Answering Portals”

[8] – <https://sourceforge.net/projects/yahoodataset/>

[9] – <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>

[10] – <https://towardsdatascience.com/introduction-to-na%C3%AFve-bayes-classifier-fa59e3e24aaf>

[11] – <https://www.cuelogic.com/blog/the-levenshtein-algorithm#:~:text=The%20Levenshtein%20distance%20is%20a,one%20word%20into%20the%20other>

[12] – <https://towardsdatascience.com/understanding-cosine-similarity-and-its-application-fd42f585296a>