swt_505_5

by Vinod kumar

Submission date: 17-Feb-2020 05:18PM (UTC+0530)
Submission ID: 1258869737
File name: Thesis_SWT_505.v4_2.docx (947.13K)
Word count: 3743
Character count: 23164

A

Dissertation On

Automation Assistance

Submitted in Partial Fulfillment of the Requirement

For the Award of Degree of

Master of Technology

In

Software Technology

Submitted By

Manish Jain

University Roll No. 2K16/SWT/505

Under the Esteemed Guidance of

Dr. Vinod Kumar

Associate Professor

Department of Computer Science & Engineering, DTU, Delhi

2017-2020(Jan)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

DELHI – 110042, INDIA

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

CERTIFICATE

This is to certify that the dissertation titled **"Automation Assistance"** is a bonafide record of work done at **Delhi Technological University** by **Manish Jain, Roll No. 2K16/SWT/505**for partial fulfilment of the requirements for the degree of **Master of Technology** Degree in **Software Technology** in the **Department of Computer Science & Engineering**. This project was carried out under my supervision and has not been submitted elsewhere, whether in part or full, for the award of any other degree or diploma to the best of my knowledge and belief.

Dr. Vinod Kumar
Associate Professor,
Department of Computer Science & Engineering,
DTU

Date: _____

ACKNOWLEDGEMENT

I would like to express sincere thanks and respect towards my guide **Dr. Vinod Kumar, Associate Professor, Department of Computer Science & Engineering, Delhi Technological University Delhi,** who have supported and encourage me during the course of the project. This work could not have been accomplished without his guidance.

I consider myself very fortunate to get the opportunity to work with him and for the guidance and support I have received from him. The completion of the project is possible with his regular guidance. Special thanks for not only providing me necessary project information but also teaching the proper way and techniques of representation.

Last but not the least, I am grateful to DTU for providing good support and right resource and environment for this work to be carried out.

Manish Jain
University Roll No: 2K16/SWT/505
M.Tech (Software Technology)
Department of Computer Engineering
Delhi Technological University
Delhi-110042

ABSTRACT

System Testing is the important phase of any project success. These test cases covers the complete functionality of a project with user perspective.

Many industry are working with big projects having huge number of system test cases. Executing this huge number of test cases manually is time consuming, costly and sometime impossible. Automation makes it possible to execute the test cases in less time as compared to manual execution. Automation helps to perform this testing multiple times with less effort.So all industry depends on automation to complete a project on time.

For large projects having 10 thousands of test cases, automating this big number of test cases is itself a challenge.

While doing the automation of a test case, some test steps may be similar to previously developed cases. Instead of developing same test steps again, it is always efficient to reuse the previous developed test script and only write the script for the different test steps.

But in big projects with many members automating the huge set of test cases, it is difficult to remember and reuse the others member automation script.

We should find a tool so that one can easily reuse the others developed test scripts for similar test steps for faster automation.

Artificial Intelligence is widely used in industries now. With the availability of data it is easy to simulate the human intelligence in different activities.

By applying Artificial intelligence model on already developed Automation scripts, it is possible to reuse it in developing new Automated Test Case.

In this project we are building an AI modeled faster Automation solution.
This approach is proved to be very efficient and save lot of efforts in automating the similar test steps. It finds the similarity between the test steps and suggest top similar test scripts based on the similarity index.

Automation Engineer can choose the best suited test step and copy the already developed test script to his code.

We have also discussed its future expansion in Risk based testing and Analysis.

TABLE OF CONTENTS

Chapter 1: INTRODUCTION

1.1 PROBLEM STATEMENT

In the era of Automation industry is looking to reduce manual interventions as much as possible. This trend can be seen globally where companies are trying to reduce manual effort and increase the Automation by automating its process, development and Testing.

Automation for small projects is easy as compared to big projects. For big projects where number of Test cases are more than 10 thousands and multiple number of employees are working to automate, faster automation is a challenge. Inorder to increase the speed and efficiency of Automation we can use the artificial intelligence methods to reuse the already developed scripts in the new test case which is being automated.

So, hereby we trying to provide the suggestion of test scripts which is similar with the test steps of the test case under automation.Enhancing Automation Efficiency by test step recommendation system to support automation of huge test case bank.

1.2 WHAT ARE THE STEPS INVOLVED IN AUTOMATION ASSISTANCE

Automation Assistance involves many test steps.

1. Firstly plotting the input training data labelled as Feasible & Not feasible.
2. Then preprocessing the input data using snowball stemmer method.
3. Copy all the columns to one title column and apply preprocessing on title column.
4. Vectorize the input data using TF-IDF vectorizer.
5. Using KNN, SVC, MNB ML models for classification of vectors.
6. Loading the Testing data and applying the preprocessing on test data before inputting to Classifier.
7. Prediction and plotting of results using KNN.
8. Reading data inputs from Test Bank file.
9. Preprocessing the input test bank using Stemmer.
10. Copy All Columns to Title Column & Apply preprocessing on Title column.
11. Calculate vectors for Test bank using linear kernel function.
12. Input query to find similar testcases.
13. Finding Top 5 Similar testcases to the input query.

1.3 THESIS MOTIVATION AND GOAL

Currently there is a lot of automation activities happening to automate the test cases. Preparation of the automation test cases is always a time consuming activity. It is always helpful to find a way to reduce this time of automation preparation.

So it makes sense in developing such system where the assistance for faster automation can be provided to the developer where it first get trained from the developed test cases and later use this data to compare the test steps of the new test cases and provide the suggestion.

1.4THESIS DETAILS IN DEPTH

This thesis is divided into six different parts.

Chapter 1defines the problem statement for the thesis, which istime consuming activity of automation in making the test script.Thismethodhas shown accurate results with the trained data and applied for the sample test cases.

Chapter 2 explains the topics used in detail. In our research we have explain the terms Word Stemming, KNN model, TFITF, Cosine Similarity concepts and vectorization.

Chapter 3 isexplaining the work flow and solution. Step by step functioning is explained in this section from the training data to applying it on the new test case.

Chapter 4 illustrates the Topfour similar test cases found matching with the current test case.

Chapter 5 is the conclusion of the thesis. It describes the Small and big AI implementation can be a basic working methodology in the industry. We can use similar recommendation systems in the industry which enhance the performance and efficiency. Also the future work which can be done on the current work.

Chapter 2: RESEARCH BACKGROUND

2.1 WORD STEMMING – SNOWBALL STEMMER

Stemming refers to the methodology of downsizing downsized terms to their own original form from where the term has originated, which is usually in its original textual type . Of all the available search engines,majority of them declares text of the same original form such as those having similar meanings as that of this.

The stem may be different to the architectural base form of the text. The assumption made here is that the words that are in relationship with each other points out to such a stem that almost signifies a similar meaning , even if this stem is not in itself a correct root. Many Algorithms are designed for stemming past 50 years.

It has made the process of machine learning possible for text and prediction can be made easily on top of the data set.

5

Snowball is a small string processing language designed for creating stemming algorithms for use in Information Retrieval. The Snowball compiler translates a Snowball script into another language - currently ISO C, C#, Go, Java, Javascript, Object Pascal, Python and Rust are supported.

2.2 WORD VECTORIZATION

Extracting meaningful message or information from the text, a sentence using machine learning and deep learning techniques requires the text needs to be converted into a set of real numbers (a vector) — Word Embeddings.

Word Embeddings or Word vectorization is a methodology in NLP to map words or phrases from vocabulary to a corresponding vector of real numbers which used to find word predictions, word similarities/semantics.

Types

- **Count Vectorizer**
    - The most basic way to convert text into vectors is through a Count Vectorizer.
    - **Step 1**: Mark unique words in the complete data.
    - **Step 2:** A Common length array of zeros is created for every sentence.
    - **Step 3:** Total occurrence of the first word is calculated for each sentence reading one at a time. After calculating the number of occurrence in that sentence, position of the word in the list above is identified and replace the same zero with this count at that position. This is repeated for all words and for all sentences
- **TF-IDF Vectorizer** (Explained in detail in next sections)
    - While Count Vectorizer converts each sentence into its own vector, it does not consider the importance of a word across the complete list of sentences.
- **Hashing Vectorizer**
    - This vectorizer is very useful as it allows us to convert any word into it's hash and does not require the generation of any vocabulary.
    - Step 1: Define the size of vector to be created for each sentence
    - Step 2: Apply the hashing algorithm (like MurmurHash) to the sentence
    - Step 3: Repeat step 2 for all sentences
- **Word2Vec**
    - These are a set of neural network models that have the aim to represent words in the vector space. These models are highly efficient and perform at in understanding the context and relation between words. Similar words are placed close together in the vector space while dissimilar words are placed wide apart.
    - It is so amazing to represent words that it is even able to identify key relationships

TF-IDF Vectorizer

TF-IDF is a product of two parts:

TF (Term Frequency) — It is defined as the number of times a word appears in the given sentence.

IDF (Inverse Document Frequency) — It is defined as the log to the base e of number of the total documents divided by the documents in which the word appears.

Step 1: Identify unique words in the complete text data. In our case, the list is as follows (17 words):

```
['ended', 'everyone', 'field', 'football', 'game', 'he', 'in',
'is', 'it', 'playing', 'raining', 'running', 'started', 'the',
'towards', 'was', 'while']
```

Step 2: Generate an array of zeros with the same length as above for each sentence.

Step 3: For each word in each sentence, we'll calculate the TF-IDF value and update the corresponding value in the vector of that sentence

Example

We'll first define an array of zeros for all the 17 unique words in all sentences combined.

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

I'll take the word he in the first sentence, He is playing in the field and apply TF-IDF for it. The value will then be updated in the array for the sentence and repeated for all words.

```
Total documents (N): 4
cuments in which the word appears (n): 2
Number of times the word appears in the first sentence: 1
Number of words in the first sentence: 6Term Frequency(TF) =
1Inverse Document Frequency(IDF) = log(N/n)
                                 = log(4/2)
                                 = log(2)TF-IDF value = 1 * log(2)
             = 0.69314718
```

Updated vector:

[0, 0, 0, 0, 0, 0.69314718, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

The same will get repeated for all other words. However, some libraries may use different methods to calculate this value. For example, sklearn, calculates the Inverse Document Frequency as:

IDF = (log(N/n)) + 1
Thus, the TF-IDF value would be:

TF-IDF value = 1 * (log(4/2) + 1)
= 1 * (log(2) + 1)
= 1.69314718
The process when repeated would represent the vector for first sentence as:

[0, 0, 1.69314718, 0, 0, 1.69314718, 1.69314718, 1.69314718, 0, 1.69314718, 0, 0, 0, 1, 0, 0, 0]

TF-IDF Summary

term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection

Document 1	
Term	Term Count
this	1
is	1
a	2
sample	1

Document 2	
Term	Term Count
this	1
is	1
another	2
example	3

$$\mathrm{tf}(''\text{example}'', d_1) = \frac{0}{5} = 0$$

$$\mathrm{tf}(''\text{example}'', d_2) = \frac{3}{7} \approx 0.429$$

$$\mathrm{idf}(''\text{example}'', D) = \log\left(\frac{2}{1}\right) = 0.301$$

Finally,

$$\mathrm{tfidf}(''\text{example}'', d_1, D) = \mathrm{tf}(''\text{example}'', d_1) \times \mathrm{idf}(''\text{example}'', D) = 0 \times 0.301 = 0$$

$$\mathrm{tfidf}(''\text{example}'', d_2, D) = \mathrm{tf}(''\text{example}'', d_2) \times \mathrm{idf}(''\text{example}'', D) = 0.429 \times 0.301 \approx 0.129$$

Fig 2.1: TF-IDF Summary

2.3 COSINE SIMILARITY

Cosine similarity is used to measure the similarity of two vectors. These vectors are that of inner product space. This inner product space measures the cosine of the angle between them. Here vectors should be non zero. The cosine of 0° is 1. Any angle between 0 and π radians has less than 1 cosine. Therefore Cosine similarity is an orientation similarity and not the magnitude. Therefore we can say that 2 vectors having same orientation have 1 as their cosine similarity. 2 vectors with 90° orientation with each other have 0 as cosine similarity. 2 vectors opposite orientation have a -1 as cosine similarity.

The cosine similarity is particularly used in positive space where the outcome is neatly bounded in {\displaystyle[0,1]}. In the case of direction cosine the unit vectors are fully similar if they're parallel and fully dissimilar if they're perpendicular.

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

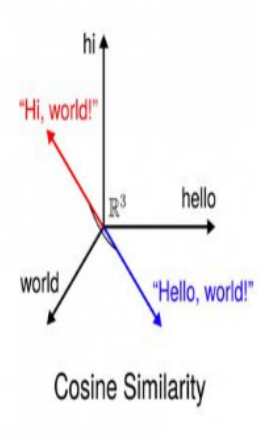


Fig 2.2 : Cosine Similarity

Cosine Similarity Visualization

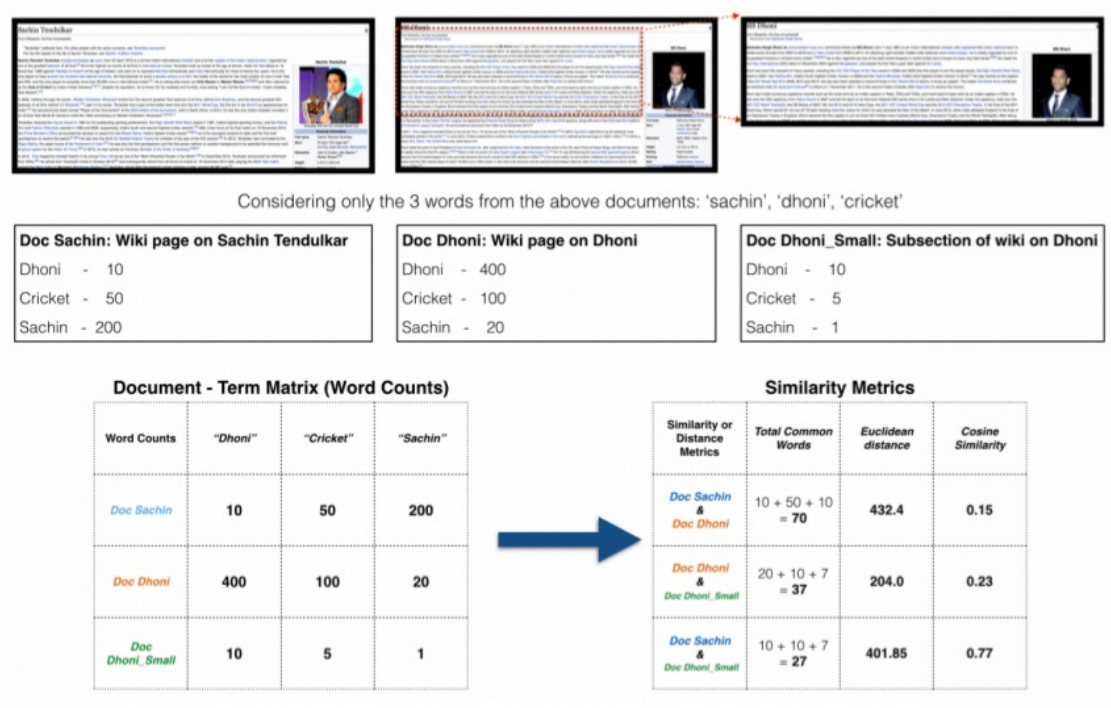


Fig 2.3 : Three document of Similar Matrices

2.4 CLASSIFICATION MODELS

Supervised Machine Learning: The majority of practical machine learning uses supervised learning. Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output Y = f(X). The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

Techniques of Supervised Machine Learning algorithms include linear and logistic regression, multi-class classification, Decision Trees and support vector machines. Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. For example, a classification algorithm will learn to identify animals after being trained on a dataset of images that are properly labeled with the species of the animal and some identifying characteristics.

Supervised learning problems can be further grouped into Regression and Classification problems. Both problems have as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for regression and categorical for classification.

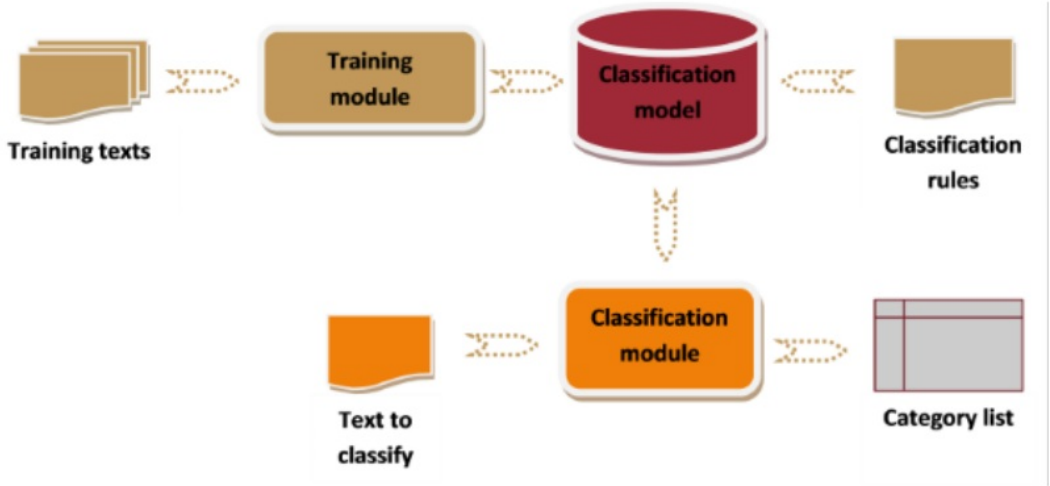


Fig: 3.1: K-Nearest Neighbours

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

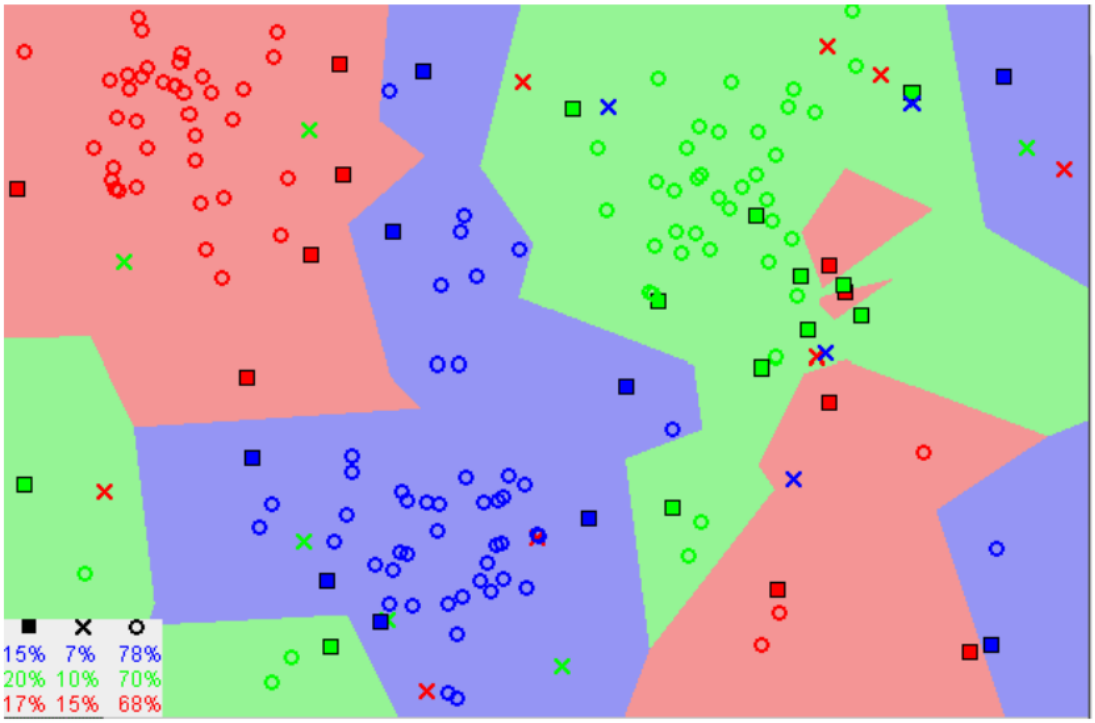


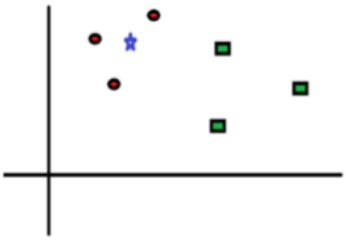
Fig: 3.2: K-Nearest Neighbours Working model

Similar data points are nearby each other in the above image. In KNN algorithm the distance of similarity is calculated between points on a graph.

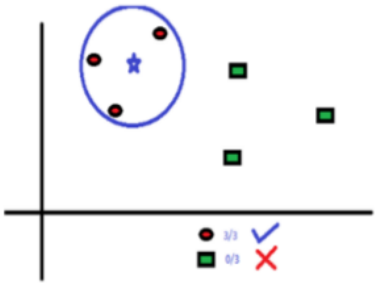
This distance can be calculated in many ways depending on the problem we are handing or the data set of the given set. Most popular one is the Euclidean distance which is a straight line distance.

KNN can be used for classification predictive problems

Figure shows a spread of red circles (RC) and green squares (GS) :



We intend to find out the class of the blue star (BS) . BS can either be RC or GS and nothing else. The "K" is KNN algorithm is the nearest neighbors we wish to take vote from. Let's say K = 3. Hence, we will now make a circle with BS as center just as big as to enclose only three data points on the plane.



The three closest points to BS is all RC. Hence, with good confidence level we can say that the BS should belong to the class RC. Here, the choice became very obvious as all three votes from the closest neighbor went to RC. The choice of the parameter K is very crucial in this algorithm.

Advantages

Its implementation and usage is quite easy than others.

We can directly use it without building any model or tuning parameters. It can be used without making any extra assumptions.

This algorithm can be used for various problems like search, regression and classification.

Disadvantages

Only disadvantage in this algorithm is it gets very slow with the increase of variables.

Chapter 3: WORKFLOW AND SOLUTION

3.1WORKFLOW

(refer fig 1)

a. Training the data – Select the Test Cases for which scripts are already developed. Mark these test cases as Feasible Also mark historical non feasible test cases.
b. Use Snowball stemmer to preprocess these test cases to get the root words of all the selected test cases.
c. Generate Word Vectors – Now use TFIDF technique to generate word vectors from training corpus.
d. Use KNN classification model to train the word vectors.
e. Now take new test cases which you want Automate.
f. First Model will predict its feasibility and then apply Cosine similarity with corpus if predicted feasibility is true.
g. List of top 5 similar test cases will be shown as result.

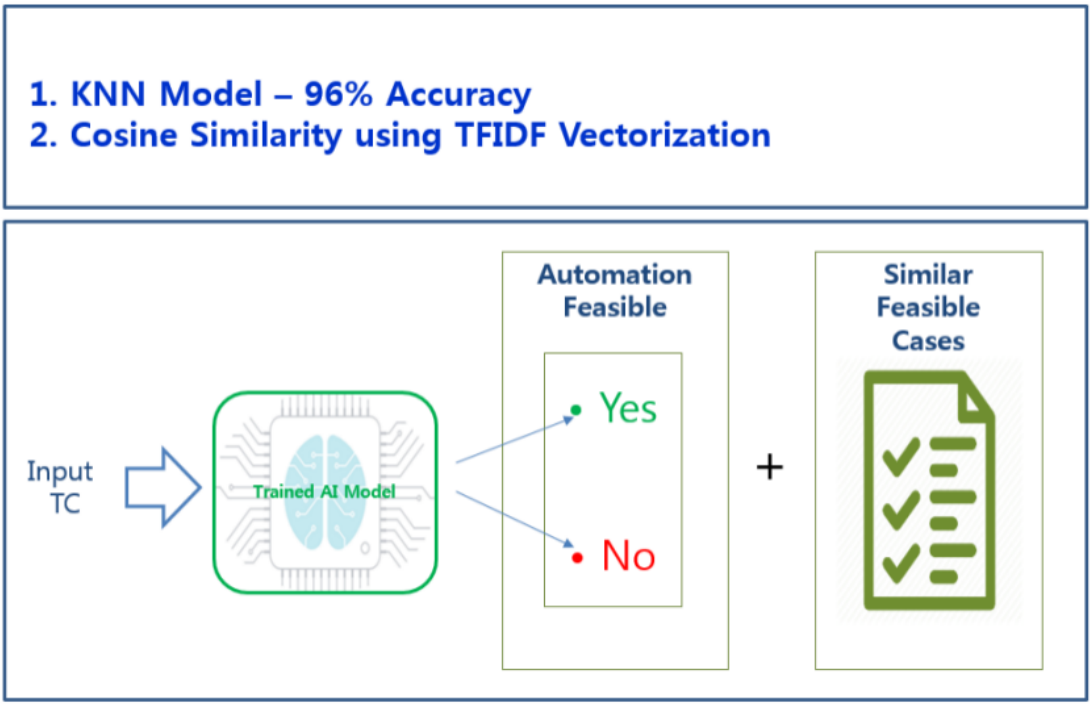


Fig: 3.3: K-Nearest Neighbours Accuracy

3.2DETAILED STEPS

The very first step involved in building training data.

Plotting input training data labelled as Feasible & Not feasible.

```
importmatplotlib. pyplotasplt
%matplotlib inline
count_Class=pd.value_counts(trainData["LABLE"],sort=True)
count_Class.plot(kind='pie',autopct='%1.0f%%')
plt.title('Pie chart')
plt.ylabel('')
plt.show()
```

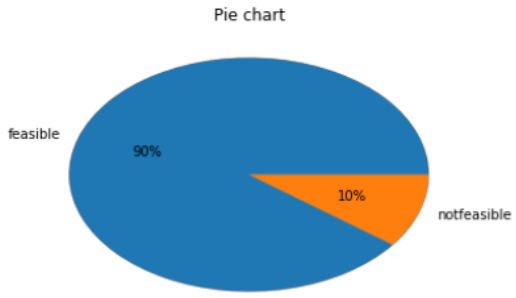


Fig: 3.4: Pie chart of feasibility

Preprocessing input data using snowball stemmer

```
importnltk
fromnltk.stemimportSnowballStemmer
fromnltk.corpusimportstopwords
defpre_process(text):
#print("Before",text)
#print(type(text),text)
text=str(text)
text=text.translate(str.maketrans('','',string.punctuation))
text=[wordforwordintext.split()ifword.lower()notinstopwords.words('english')]
text=[wordforwordintextifword.lower()notin['1','2','3','4','5','6','7','8','test','verify']]
#print("After",text)
words=""
foriintext:
stemmer=SnowballStemmer("english")
words+=(stemmer.stem(i))+" "
returnwords
```

Copy all the columns to one title column and apply preprocessing on titel column

```
trainData['TITLE']=trainData['TITLE']+trainData['PRECONDITION']+trainData['STEPS']+train
Data['EXPECTED_RESULT']
textFeatures=trainData['TITLE'].copy()
labels_train=trainData['LABLE'].copy()
textFeatures=textFeatures.apply(pre_process)
```

Vectorize the input data using TF-IDF vectorizer

```
fromsklearn.feature_extraction.textimportTfidfVectorizer
vectorizer1=TfidfVectorizer("english")
features=vectorizer1.fit_transform(textFeatures)
```

Using KNN, SVC , MNB ML models for classification of vectors

```
#accuracy calculate
fromsklearn.model_selectionimporttrain_test_split
fromsklearn.metricsimportaccuracy_score
fromsklearn.svmimportSVC
features_train,features_test,labels_train,labels_test=train_test_split(features,trainData['LABLE'],test_size=0.3,random_state=111)
print("Accuracy :")
fromsklearn.neighborsimportKNeighborsClassifier
fromsklearn.model_selectionimporttrain_test_split
knn=KNeighborsClassifier(n_neighbors=1)
knn.fit(features_train,labels_train)
prediction=knn.predict(features_test)
print("KNN ---> ",accuracy_score(labels_test,prediction)*100)

svc=SVC(gamma=1.0)
svc.fit(features_train,labels_train)
prediction=svc.predict(features_test)
print("SVC ---> ",accuracy_score(labels_test,prediction)*100)

fromsklearn.naive_bayesimportMultinomialNB
```

```
mnb=MultinomialNB(alpha=0.2)
mnb.fit(features_train,labels_train)
prediction=mnb.predict(features_test)
print("MNB ---> ",accuracy_score(labels_test,prediction)*100)
```

Accuracy :

KNN ---> 97.5609756097561

SVC ---> 95.12195121951220

MNB ---> 97.5609756097561

As per above result, KNN gives the best accuracy as compared to SVC and MNB.

Loading the Testing data and applying the preprocessing on test data before inputting to Classifier

```
TEST_COLUMNS=['ID','TITLE','PRECONDITION','STEPS','EXPECTED_RESULT']
testData=pd.read_csv('D:\\A2\\notfeasible\\notfeasible\\RegressionTestData.csv',header=None,names=TEST_COLUMNS)[:3000]
testData['TITLE']=testData['TITLE']+testData['PRECONDITION']+testData['STEPS']+testData['EXPECTED_RESULT']
testTextFeatures=testData['TITLE'].copy()
testTextFeatures=testTextFeatures.apply(pre_process)
testFeatures=vectorizer1.transform(testTextFeatures)
```

Prediction and plotting of results using KNN

```
fromsklearn.neighborsimportKNeighborsClassifier
prediction=knn.predict(testFeatures)
#print(type(prediction),prediction)
importmatplotlib.pyplotasplt
%matplotlib inline
count_Class=pd.value_counts(prediction,sort=True)
count_Class.plot(kind='pie',autopct='%1.0f%%')
plt.title('Pie chart')
plt.ylabel('')
plt.show()
```

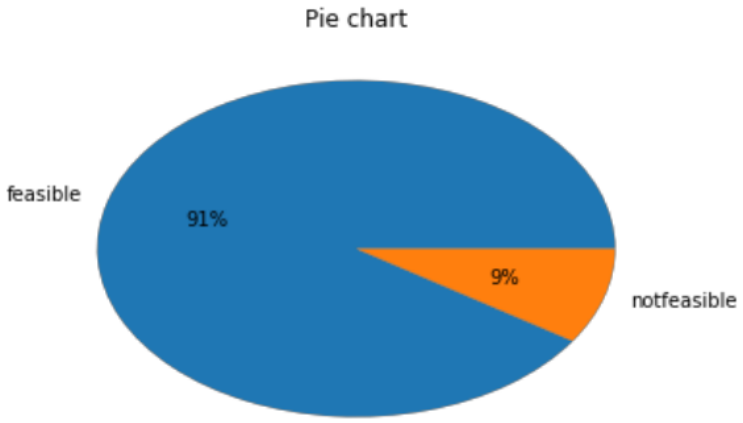


Fig: 3.5: Pie chart of feasibility

Reading data inputs from Test Bank file

```
fromsklearn.feature_extraction.textimportTfidfVectorizer,ENGLISH_STOP_WORDS
fromsklearn.metrics.pairwiseimportlinear_kernel
importpandasaspd
importstring
```

```
TRAINING_COLUMNS=['ID','TITLE','PRECONDITION','STEPS','EXPECTEDRESULT']
trainData=pd.read_csv('D://A2//Recommendation//Recommendation//data//RecomdationBankCSV.csv',header=None,names=TRAINING_COLUMNS,encoding="latin1",engine='python')[:3000]
rawData=trainData;
```

Preprocessing the input test bank using Stemmer

```
fromnltk.stemimportSnowballStemmer
fromnltk.corpusimportstopwords
defpre_process(text):
#print("Before",text)
#print(type(text),text)
text=str(text)
text=text.translate(str.maketrans("",string.punctuation))
text=[wordforwordintext.split()ifword.lower()notinstopwords.words('english')]
text=[wordforwordintextifword.lower()notin['1','2','3','4','5','6','7','8','test','verify']]
#print("After",text)
words=""
foriintext:
stemmer=SnowballStemmer("english")
words+=(stemmer.stem(i))+" "
returnwords
```

Copy All Columns to Title Column & Apply preprocessing on Title column

```
trainData['TITLE']=trainData['TITLE']+trainData['PRECONDITION']+trainData['STEPS']+train
Data['EXPECTEDRESULT']
textFeatures=trainData['TITLE'].copy()
textFeatures=textFeatures.apply(pre_process)
```

Calculate vectors for Test bank using linear kernel function

```
fromsklearn.feature_extraction.textimportTfidfVectorizer
vectorizer=TfidfVectorizer("english")
vec_train=vectorizer.fit_transform(textFeatures)
cosine_sim=linear_kernel(vec_train[0:1],vec_train).flatten()
```

Input query to find similar testcases

```
query="Test to verify Submit Feedback after fill all data."
```

Finding Top 5 Similar testcases to the input query

```
query=pre_process(query)
vec_query=vectorizer.transform([query])
cosine_sim=linear_kernel(vec_query,vec_train).flatten()
outputIndexes=cosine_sim.argsort()[:-10:-1]

#printing top 5 testcases based on cosine similarity
print(rawData["ID"].as_matrix()[outputIndexes[0]])
print(rawData["TITLE"].as_matrix()[outputIndexes[0]],"\n\n")
print(rawData["ID"].as_matrix()[outputIndexes[1]])
print(rawData["TITLE"][outputIndexes[1]],"\n\n")
print(rawData["ID"].as_matrix()[outputIndexes[2]])
print(rawData["TITLE"][outputIndexes[2]],"\n\n")
print(rawData["ID"].as_matrix()[outputIndexes[3]])
print(rawData["TITLE"][outputIndexes[3]],"\n\n")
print(rawData["ID"].as_matrix()[outputIndexes[4]])
print(rawData["TITLE"][outputIndexes[4]],"\n\n")
```

Chapter 4: RESULTS

Four similar test cases and their script is shown to the tester against the query

<<Test to verify Submit Feedback after fill all data.**>>**

C65749726

Test to verify Submit Feedback after fill all data.None.1. Launch KC.2. Open "FEEDBACK & SUPPORT" tab.3. Enter all fields and click on submit button button.1. KC should be launched successfully.2. "FEEDBACK & SUPPORT" tab should get open successfully.3. Feedback get submit or confirmation popup show on screen.None.1. Launch KC.2. Open "FEEDBACK & SUPPORT" tab.3. Enter all fields and click on submit button button.1. KC should be launched successfully.2. "FEEDBACK & SUPPORT" tab should get open successfully.3. Feedback get submit or confirmation popup show on screen.

C65749728

Test to verify submitting feedback with blank feedback box.None1. Launch KC.2. Open "FEEDBACK & SUPPORT" tab.3. Enter Topic4.Try to submit feedback form with blank feedback box field[by deleting feeback box text].1. KC should be launched successfully.2. "FEEDBACK & SUPPORT" tab should get open successfully.3. Submit should be disable.4. Warning text should display " This field is required. "None1. Launch KC.2. Open "FEEDBACK & SUPPORT" tab.3. Enter Topic4.Try to submit feedback form with blank feedback box field[by deleting feeback box text].1. KC should be launched successfully.2. "FEEDBACK & SUPPORT" tab should get open successfully.3. Submit should be disable.4. Warning text should display " This field is required. "

C65749728

Test to verify submitting feedback with blank feedback box.None1. Launch KC.2. Open "FEEDBACK & SUPPORT" tab.3. Enter Topic4.Try to submit feedback form with blank feedback box field[by deleting feeback box text].1. KC should be launched successfully.2. "FEEDBACK & SUPPORT" tab should get open successfully.3. Submit should be disable.4. Warning text should display " This field is required. "None1. Launch KC.2. Open "FEEDBACK & SUPPORT" tab.3. Enter Topic4.Try to submit feedback form with blank feedback box field[by deleting feeback box text].1. KC should be launched successfully.2. "FEEDBACK & SUPPORT" tab should get open successfully.3. Submit should be disable.4. Warning text should display " This field is required. "

C65749727

Test to verify submitting feedback with blank subject field.None1. Launch KC.2. Open "FEEDBACK & SUPPORT" tab.3. Click on Send Feedback button4. Leave subject field blank write something in feedback box and submit.1. KC should be launched successfully.2. "FEEDBACK & SUPPORT" tab should get open successfully.3. Send Feedback pop up should open4. Feedback form should get submitted successfully.None1. Launch KC.2. Open "FEEDBACK & SUPPORT" tab.3. Click on Send Feedback button4. Leave subject field blank write something in feedback box and submit.1. KC should be launched successfully.2. "FEEDBACK & SUPPORT" tab should get open successfully.3. Send Feedback pop up should open4. Feedback form should get submitted successfully.

Chapter 5: CONCLUSION

5.1CONCLUSION

The AI implementation is now increasing its horizon from commercialization to Automation. Now we can enhance our efficiency at each level of product development by using AI.

Small and big AI implementation can be a basic working methodology in the industry.We can use similar recommendation systems in the industry which enhance the performance and efficiency.

5.2FUTURE WORK

1. More accuracy may be achieved using other vectorization models like word2vec, Fasttextetc,

2. Similar approach can be applied to number of areas like Risk based testing in Regression. Changes can be matched with the existing test cases and similar test cases can be picked to verify.

3. First Cut Analysis based on defects fixed history. Automated system can be developed in defect analyzing and suggesting high-level analysis to the developer based on similar issue fixing history.

REFERENCES

[1] Rech, J. and Althoff, K.D., “Artificial intelligence and software engineering: Status and future trends.”, KI, 18(3), pp.5-11, 2004.

[2] Wang, Y., “Convergence of Software Science and Computational Intelligence: A New Transdisciplinary research Field”, In Software andIntelligent Sciences: New Transdisciplinary Findings, pp. 1-13, 2012.

[3] Abreu, R., Zoeteweij, P., & Van Gemund, A. J., “An evaluation of similarity coefficients for software fault localization”. In Dependable Computing, 2006. PRDC'06. 12th Pacific Rim International Symposium, pp. 39-46, 2006.

[4] Larkman, D., Mohammadian, M., Balachandran, B., &Jentzsch, R., “Fuzzy cognitive map for software testing using artificial intelligence techniques.”, in IFIP International Conference on Artificial Intelligence Applications and Innovations, pp. 328-335, October 2010.

[5] Kosko, Bart. "Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence/book and disk." Vol. 1Prentice hall,1992.

[6] Howe, A. E., Von Mayrhauser, A., & Mraz, R. T., “Test case generation as an AI planning problem”, in Knowledge-Based Software Engineering (pp. 77-106). Springer US, 1997.

[7] Sorte, Bhagyashree W., Pooja P. Joshi, and VandanaJagtap. "Use of Artificial Intelligence in Software Development Life Cycle: A state of theArt Review."

[8] A Survey on the Role of Artificial Intelligence in Software Engineering Vol. 5, Issue 4, April 2017

[9] Using artificial intelligence to automatically test GUI by Abdul Rauf, Mohammad N. Alanazi 2014 9th International Conference on Computer Science & Education

swt_505_5

ORIGINALITY REPORT

14%	10%	5%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Delhi Technological University Student Paper	3%
2	Submitted to Aston University Student Paper	1%
3	python.astrotech.io Internet Source	1%
4	Submitted to Indian Institute of Technology, Kanpur Student Paper	1%
5	anaconda.org Internet Source	1%
6	Submitted to University of Wales Swansea Student Paper	1%
7	Submitted to Carnegie Mellon University Student Paper	1%
8	www.techtud.com Internet Source	1%
9	www.adityamangal.com	

	Internet Source	<1%
10	tuprints.ulb.tu-darmstadt.de Internet Source	<1%
11	hdl.handle.net Internet Source	<1%
12	raiith.iith.ac.in Internet Source	<1%
13	Submitted to Rutgers University, New Brunswick Student Paper	<1%
14	Boonyoung, Thanyaporn, and Anirach Mingkhwan. "Semantic ranking based on Computer Science Ontology weight", Ninth International Conference on Digital Information Management (ICDIM 2014), 2014. Publication	<1%
15	G. Burek, A. De Roeck, Z. Zdrahal. "Hybrid Mappings of Complex Questions over an Integrated Semantic Space", 16th International Workshop on Database and Expert Systems Applications (DEXA'05), 2005 Publication	<1%
16	Submitted to University of Hong Kong Student Paper	<1%
17	Submitted to The Hague University	

	Student Paper	<1%
18	www.i-scholar.in Internet Source	<1%
19	Submitted to CSU, San Jose State University Student Paper	<1%
20	Submitted to The Hong Kong Polytechnic University Student Paper	<1%
21	Pasquale Lops, Cataldo Musto, Fedelucio Narducci, Giovanni Semeraro. "Semantics in Adaptive and Personalised Systems", Springer Science and Business Media LLC, 2019 Publication	<1%
22	ethesis.nitrkl.ac.in Internet Source	<1%
23	Submitted to Sri Lanka Institute of Information Technology Student Paper	<1%
24	Submitted to Borah High School Student Paper	<1%
25	Submitted to Visvesvaraya Technological University Student Paper	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		