Automatically Categorizing Software Technologies

By

Suleman Khan

00000274810

Supervisor

Dr. Wasi Haider Butt

Department of Computer and Software Engineering

College of Electrical and Mechanical Engineering

National University of Science and Technology (NUST)

Islamabad, Pakistan

September 2022

Automatically Categorizing Software Technologies

By

Suleman Khan

00000274810

Supervisor

Dr. Wasi Haider Butt

A thesis submitted in conformity with the requirements for the degree of *Master of Science* in Software Engineering

Department of Computer and Software Engineering

College of Electrical and Mechanical Engineering

National University of Sciences and Technology (NUST)

Islamabad, Pakistan

September 2022

DECLARATION

I verify that this research exertion titled "*Automatically Categorizing Software Technologies"* is my own work. It has not been presented anywhere else for valuation or publication. The facts and figures which have been used in this research from other sources have been properly referred.

_____

Student's Signature

Suleman Khan

MS-18 CSE

PLAGIARISM CERTIFICATE (TURNITIN REPORT)

Turnitin report sanctioned by Supervisor is attached.

_____

Student's Signature

Suleman Khan

00000274810

Supervisor's Signature

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

I am beholden to Allah Almighty to have shown me the way all over this research at every step and for every new thought undeniably. Whosoever helped me throughout the course of my thesis, whether my parents, or any other individual was Your will, so indeed none be worthy of praise but You.

I would also like to express special gratitude to my supervisor Dr. Wasi Haider Butt for his assistance all over this journey, and to GEC members Dr. Muhammad Umar Farooq and Dr. Arslan Shaukat for their guidance. Lastly, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my thesis.

ABSTRACT

The following research aims to create an automatic detection of programing language. Taking source codes of a programing language as an input and giving output the names of the programming languages of the respective input data. Due to unstructured form of the literature and data available on internet and data repositories regarding this topic, it is hard for researchers and users to manage that kind of data easily. Categorization is the best way to develop formal knowledge base of unstructured data in a formal way by considering source code of different programming languages and relationship of data available. Detection of programing languages related research slightly increases in recent years as it is not easy to develop expert systems or artificial intelligence-based systems using raw data of huge amount of source codes. In this research Resource description framework data regarding the source code of programming languages such as C, C++, Python etc. has been collected and used for developing a system which not only detect the name of the language whose source code is entered, but also ensures maximum attainable level of accuracy.

Existing available detection tools rather focuses on the lack of informal language and software technology standard taxonomy makes it impossible to analyze technology trends on forums and other online sites. Furthermore, it defines its function (commercial, PHP). By extension, this method can dynamically compile the list using all technologies of a given type.

Keywords: *IDE, PHP, WITT, WebIsADb, WiBiTaxonomy*

CONTENT

LIST OF FIGURES

1st Chapter

Introduction

CHAPTER 1: INTRODUCTION

Knowledge is inadequate and useless if it is not presented and managed in an efficient manner. Efficient sharing of knowledge helps in achieving innovations and it results in development of humankind. In past decade, there is a bang of knowledge available on the internet and research publications. Specially, in the domain of software tools and technologies, huge amount of research been carried-out and sharing this categorization knowledge made an important instrumental step towards development. However, a lot of software knowledge is stored in the form which is not easy to access for everyone because of its unstructured nature. Massive research on this topic is available on internet but it is difficult to obtain the required results. If data related to the hypernyms, source codes and tags related to different software tools and technologies is easily accessible to everyone then there would be many advantages accomplished including, accurate knowledge sharing and reusability.

At present, people are intrigued about the concept of automatic detection of the names of programing languages, automatically categorization of software technologies and the usage of artificial intelligence knowledge. They are more worried about their time and want to have a system which give them complete knowledge about the best available tool. The method in use for detection of programing languages is Glasslang and for software categorization is MUDABlue. This method would categorize an enormous collection of software platforms automatically. It does not only organize computer software structures but also find out clusters from the classifications group inevitably. This technique can sort starved of any information about the software in search. Moreover, they employed an interface to this method, a category-based package repository browsing system. This method permits scanning a depository considered, where a system can fit into numerous classes [1].

These people use search engines for search of their particular information. These type of searches over the internet increases on daily basis by persons do not belong to software domain. Particularly, the data or information in software categorization domain available in unstructured manner on and on the web [2]. Most commonly used search engines like Google have not able to address this issue until now. Main cause of this issue is the lack of semantic orientation of data available online on publication or web [3]. If

data related to software tools easily accessible to everyone then there would be many advantages accomplished including, accurate information sharing and reusability.

1.1 Background and Drive

There is a huge amount of literature on software applications or tools is created in last few decades, key reason for this explosion is the development of new tools and technologies every other day. Now it becomes tangible that this huge textual form of data can be efficiently used through automated text extraction approaches [4]. Huge amount of software data or information is available on internet in an unstructured form, but the one cannot easily extract required results [5]. In software domain, there is a lot of information present in unstructured form in publications. This problem is not yet addressed by commonly used search engines and text extraction tools and techniques [4]. One of the main reasons for this issue is lack of semantic orientation of available data on internet and publications, which makes search engines unable to get required results. If this information will be easily available to the people, then people can use it for their benefit in an efficient way.

Semantic web terminology is usually profited without rich consideration of the associations and backgrounds data. It is an extension of the World Wide Web (www) that are associated in such a way that they can easily be processed by machines or computers instead of human operators [6]. Thus, PCs can make significant translations like the way human interaction data to accomplish their objectives. This nature of semantic web permits putting away information in underlying structure. Converting conventional web toward semantic is really challenging task. Thus, this leads our goal towards creating a system which may have organized data and will help in detection of programming languages from their source codes.

Programming order is officially characterized as "Express association of programming in a gathering. These gatherings permit programming to be figured out with regards to those classifications, rather than particularities of each bundle. Different arrangement plans think about various parts of order." [7]. Software depositories sustain programs that are usually classified to boost the usefulness of several conservation responsibilities. Appropriately classified programs permit collaborators to detect necessities interrelated to their programs and envisage conservation glitches in software packages. Manual

cataloguing is exorbitant, monotonous, and arduous. Therefore, automatic categorization methods are attaining prevalent significance. Regrettably, for unalike legitimate and organizational details, the software's source code is every so often unavailable, hence making it hard to robotically sort these software's. In addition, the authors recommend an innovative method in which they use these requests from independent libraries for the programmed sorting of programming systems which utilize these requests.

Natural language as well as the non-existence of a set of nomenclature for applications prepare it hard to dependably evaluate automation courses on debate mediums as well as alternative networking platforms. Authors suggest an automated method known as Witt for software package technologies categorization (an extended form of the hypernym unearthing problem). This method obtains as input a sentence unfolding an application technology or notion and gives back a standard class that elaborates it (e.g., IDE), together with features that further certify. In addition, the method empowers the run-time formulations of catalogues of entire technologies of a specified kind. Furthermore, Stack Overflow as well as Wikipedia are the two main sources of this technique. In addition to this, it encompasses several novel field transformations and a resolution to standardizing spotted hypernyms' issue. Authors compare Witt with six autonomous classification technologies and figure out that, once applied to software terms, this technique established improved exposure compared to all gauged substitute methods, without conforming to dilapidation in false-positive rate [8].

Software development is growingly dependent on off-the-shelf elements in the shape of frames, libraries, coding languages as well as instruments to practice them. Natural language and the want of a typical classification for programs concoct it hard to dependably examine high-tech trends in dialogue mediums as well as additional ubiquitous places. Witt is recommended for an automatic classification of software methods. A simple sentence is acquired by the system which describes a package notion and gives back a common class which explains it (e.g., an IDE), accompanying characteristics that certify it even further. By means of augmentation, the method permits the run-time formation of catalogue of all applications of a specified nature [9].

1.2 Aims & Objectives

The abundance of data which come from various sources may hinder the retrieving

process of useful knowledge. Due to that reason, detection and categorization approach in data integration has attracted the attention due to its ability in doing research work. Henceforth, the available study on the idea of automatic detection of programing languages has lot of potential and study related to an automatic categorization of software technologies could be studied in more detail and offered in a holistic form for individuals as well as corporations. So that they can get the best viable options when they are going to search online. The better the categorization technology the better the results. Moreover, it will not only give required results but also save individuals' valuable time. Consequently, it is necessary to conduct the systematic literature review to figure out the details about the available software technologies in a market for automatic detection.

1.3 Structure of thesis:

1st chapter: Includes an Overview of the thesis, derive behind this research work and domain information, motivation for topic selection and objectives of the research and structure of this thesis. 2nd chapter comprises the detailed systematic literature review whereas the 3rd chapter highlights the proposed methodology and implementation of our proposed idea. 4th chapter comprises Results and Validation. 5th chapter comprises conclusion that concludes the thesis along with ideas for further improving the existing work under the heading future work.

2nd Chapter

Literature Review

CHAPTER 2: LITERATURE REVIEW

In this section overview of the already research presented on the subject of the automatic detection of software languages is produced. Related work for the domain of software tools, which are developed by different data sources and their features, are presented comprehensively in this chapter. The shortcoming of these related developed technologies is discussed and how we try to overcome these shortcomings. The features and coverage of general categorization tools are also discussed.

2.1 Overview

The automatic detection of language relations can usually trace back to the development of WordNet [10], which is a hand-made catalogue of connotations, like hypernyms or synonyms. Of particular relevance to researcher's effort is Miller et al. Explanation off's pseudonym 'A concept x is a pseudonym for the concept y because native English speakers accept sentences composed of frames, such as x is a (some kind) y".

Then automatically construct word relations using text extraction: Hearst proposes a series of dictionary syntax patterns that usually represent lower words (e.g., "like X") [11], and Caraballo expands this idea by putting upper words together in a hierarchical structure add [12]. These methods have been improved by using language dependence [7] as well as guided algorithms of deep learning [8]. These methods function by exploiting a plethora of text. One of the up-to-date related technologies is the WebIsADb [9], that extrapolates as of Common Crawl, which accesses billions of web pages.

Earlier research has every so often used Wikipedia as the vital source for the taxonomy construction, as it is considered the immense easily accessible assortment of encyclopedia knowledge [9].

Witt cannot depend on these out-of-date methods since the label info on Stack Overflow is usually brief and every so often lacking; the author cannot accept the existence of related links. Therefore, they implemented a new linking method founded on diverse information and precise field circumstances.

Other researchers are working to extract semantic relationships between Wikipedia articles. e.g., Zhongshan et al. [13] search for any semantic relationship by discovering

related terms and predicates of the relationship. THD efforts to use the link hypernym data set generated from Wikipedia [14], [15] to invent the superordinate of the enquiry. Given the figure of relations and metadata available on Wikipedia, many other methods of extracting information are possible, such as the use of word matches [16], keyword popularity [17] or HTML tables [18]. The Wikipedia Bitaxonomy Project (WiBi) takes out data from Wikipedia articles to complement to the nomenclature of Wikipedia groups, and contrarywise, [19] to improve the value of the consequential facts' assembly.

Lastly, some organized information takes out effort on Wikipedia came to DBPedia [20]. Which is a structured information database, and through various tools and APIs, hyperlinks can also be called [21]. The industry is also making similar efforts [22].

The work contrasts with researchers' efforts to search for purposeful terms in hyperlinks instead of extracting entities from text, so they have to tackle by inadequate background data to discover related articles. Researchers get the better of this want of background by means of domain-specific knowledges.

Some researchers used NLP to take out key notions from the qualifiers demarcated in the code and merge them into a structure similar to WordNet, including their top relationship [23]. Similarly, another researcher uses heuristics to determine wherever notions related to the qualifiers in the source [24] are introduced or described. In both methods, each expression is obtained from code rudiments, and this material is not usually used to classify software technologies.

Labels are frequently used as descriptors of software technology so that they can be used as groups. On the usage of labels in work system management, authors originate that inventor established implied and obvious devices to oversee tag vocabulary [25]. Used for software projects such as free codes to offer a larger label vocabulary on websites, Wang et al. An equality measure has been proposed to derive lexically correlated labels in addition establish a nomenclature [26]. An agglomerated tiered grouping outline is proposed, which depends on the similarity of every two labels, exploiting information as of Ohloh [27]. It should be noted that their effort does not harvest a pecking order of hypernyms: e.g., the term hibernation is grouped as a child node of java.lang.org. In addition, earlier research has anticipated a label commendation method for works on Ohloh and Freecode [28]. Furthermore, a method for finding alike programs founded on Source Forge labels [29].

Tags are employed to specify programming lingoes, outlines, environmental issues, fields, and non-functional matters in Stack Overflow [30]. Numerous methods have been technologically advanced to recommend tags for Stack Overflow reports, together with discriminant prototypical methods [31], Bayesian probability models [31], and a method that combines multiple techniques called TagCombine [32]. Witt method inevitably classifies software applications.

2.2 Categorization

Categorization can be defined as a clear description of a conceptualization. The term first used by Aristotle and borrowed from philosophy. In philosophy it is a systematic version of any existence. In Artificial intelligence existence mean which can be represented efficiently. When domain knowledge is presented in declarative form then the set of objects represented are called "universe of discourse". These sets and relationship between them can be converted into representational knowledge based (KB) program that shows knowledge [33]. In Artificial intelligence, it can be a set of procedures which defines set of representational terminologies. Thus, categorization can also be defined as set of entities which can be grouped in a way which is based upon their uses or purposes. Generally, it is a methodology developing logical sets[34]. Therefore, this type of technique can be used in building many types of applications such as expert systems, and Natural Language Processing (NLP), and could be used as a basis for Semantic Web [35].

2.2.1 Application of Software Categorization

Categorization is the way to act as a base for development of expert and intelligent systems. It also gives an efficient answer to the question which is better tool between the two tools. One of the main advantages of using categorization is increase in efficiency, it can be act as a basis for development of another such tools due to its identical novel approach. Ontology can also provide foundational base for semantic web as ontologies are opposite to closed world traditional databases, these databases are limited to a particular domain knowledge [36]. As it possesses flexible nature that can be used for gathering more and more information, data, and knowledge. It designates the concepts and relation between these ideas which are correlated to a particular area. The application ideas are currently in market are following.

2.2.2 WiBiTaxonomy

The WiBiTaxonomy Project (WiBi) [20] uses NLP technology and existing links between articles and categories to create a better relationship chart from all articles and groups of Wikipedia. For example, WiBi bank on the hypothesis that the primary sentence of a piece of an article explains the theme of that.

2.2.2.1 THD

Targeted Hypernym Discovery (THD) [15], [16] uses manual vocabulary syntax patterns to detect hypernyms from target data foundations.

2.2.2.2 WordNet

WordNet [4] is a vocabulary database that contains information such as top and bottom ratios. The database is handmade and is measured the gold standard for many linguistic programs.

2.2.2.3 DBpedia Spotlight

DBpedia [21] is a crowdfunding database full of organized info from Wikipedia. Entry encompasses, amongst additional gears, the relationship of hypernyms. It is a method for recording manuscripts with DBpedia accesses. It extracts the published text as input and automatically extracts DBpedia entries.

2.2.2.4 WebIsADb

The WebIsADb was created by applying a set of sophisticated and extensive grammatical patterns (similar to the Hearst pattern) to the large network document corpus common crawl. Additionally, to find hypernyms, WebIsADb too practices pre-adjusters and post-modifiers. This concept is comparable to our properties.

2.2.2.5 Google

The Google search engine defines definition operators. It will try to find the word definition from the operator in the case of a search engine. If in any case only single explanation is reckoned, it will seem in a special container at the topmost portion of the network link. They use the definition as a hypernym when label is analyzed as a noun. They only cast-off totaled explanations, or instances.

2.2.2.6 Witt

Researchers considered three variants of Witt (What is Technology) method for evaluation purposes. Their variant reproduces only the original hypernym as explained in up-coming section (WittH). The other option pays back only the name of the main category lacking any extra attributes and treats the pay back category as a hypernym (WittC). For a given label, the third variant returns the corresponding category and all additional properties (WittCA). These variants are desirable to response the second query: What is the effect of the novel hypernym abstraction stage on the grouping equivalent technology?

2.3 Programing languages

Software's are written into many different languages due to their vast domain. The languages which we have used for our initial research work are.

- C
- Python
- Java
- MATLAB
- Csharp
- PHP
- HTML
- SQL

2.3.1 C language

C is a procedural programming language with a static framework that has the usefulness of organized programming, recursion, and lexical variable expansion. C is planned with develops that move well to general equipment guidelines. It has a stretched past of purpose in programs recently inscribed in composite linguistic. It is a machine-free coding language which is predominantly in use to make numerous sorts of consumptions and employed frameworks such as Microsoft-Windows and other complex projects e.g., the Oracle data set, Git, the Python translator, and games, and is regarded as a fundamental software development. during the period used up learning another programming language. Employed frameworks and different application structures for

Personal computers (PCs) models: mainframes to PLCs and implanted frameworks are illustrations of such software's.

2.3.2 Python

Python is a significant level, broadly useful, deciphered, object-situated programming language. Like PERL, It is also a well-known software development language with software engineers expertise in C++ and Java. By working in this language, clients may decipher articulations on different working frameworks, together with UNIX-based frameworks, Mac OS, MS-DOS, OS/2, and various renditions of Microsoft Windows 10 and Windows 11.

2.3.3 Java

Java is a universally useful, software development language projected to partake less organization dependance. It is a useful phase for the improvement of software application's development. Accordingly, it is fast, safe as houses, and trustworthy. It is sketchily operated for the advancement of Java applications on Personal computers, server farms, game control center, rational workstations, cell phones etc.

2.3.4 MATLAB

It is a product improvement language created by MathWorks. It started as a framework program composing language where customary variable-based number related composing PC programs was essential. It might be run both under savvy gatherings and as a bundle work. This educational activity gives you powerfully a sensitive show of MATLAB programming language. It is expected to give students experience with MATLAB programming language. Issue based MATLAB models have been given in direct and straightforward way to make your getting on rapidly and reasonable.

This software is integrated into hardware as part of larger systems to control its various functions. This type of software is embedded in the system ROM (Read Only Memory). For example, the keyboard control software embedded in a microwave or washing machine where it has to analyze data and make decisions and actions that allow the product to function as desired. This software is also called smart software because of its performance.

2.3.5 Csharp

Csharp is a generally helpful, present day and thing arranged programming language enunciated as "C Sharp". Microsoft made it drove by Anders Hejlsberg and his gathering inside the .NET drive and was upheld by the European PC Producers Affiliation (ECMA) and Global Principles Association (ISO). C# is among the tongues for Normal Language Foundation. C# is an incredible arrangement like Java semantically and is straightforward for clients who have some familiarity with C, C++, or Java.

2.3.6 PHP

PHP is an open-source server-side prearranging language that numerous versions use for web advancement. It is likewise a broadly useful language that you can use to make loads of tasks, including Graphical User Interfaces (GUIs).

2.3.7 HTML

HTML grants web clients to make and configuration portions, areas, and associations using parts, names, and qualities. Regardless, it's very huge that HTML isn't seen as a programming language as it can't make dynamic convenience.

2.3.8 SQL

It has hanged around an unfailingly recognized pronouncement for dataset throughout the long term, generally due to its usability and the profoundly productive way it questions, controls, totals information, and plays out many different capabilities to change assortments. enormous measures of organized information in valuable. data.

2.4 REASONS FOR DEVELOPING PROGRAMING LANGUAGE DETECTION TOOL:

The foremost purpose for developing this is information sharing for the investigators and researchers, which required information related to similar domain [37]. Some other reasons are listed below,

- Sharing the information required in similar domain.
- Reusing previously created ontologies related to same domain.
- Extracting domain knowledge efficiently from general or operational knowledge.

2.5 RELATED WORK

Existing devices like Google Code Prettify [38] are accessible that permit features language structure in source code pieces utilizing heuristics. Sentence structure for all of the upheld programming dialects are now predefined in the application. Yet typically this cannot distinguish the programming language of piece of code. There are additionally devices like SyntaxHighlighter [39] or Feature [40] that grammar label bits in blog entries utilizing predefined set of watchwords accessible. Devices like SourceClassifier [41] utilize Credulous Bayes classifiers for recognition programming language of an asset. As affirmed by our outcomes in this paper it is by all accounts very lacking as their precision strategies are a lot of lower than what might be viewed as satisfactory by and by. Little et al. [42] proposes a measurable method for programming language recognition

in view of the location of blocks or remark strings of the asset and execution measurable examination for exceptional characters, for example, sections, the main word in a line, last person, administrators, accentuation and so on. This technique performs better compared to execution of Source Classifier yet is still very unsatisfactory practically speaking with a precision that falls well beneath half. Source code web search tools like SearchCode [43] and Codase [44] have filed vast number of source codes and give accessibility to explicit catchphrases. Notwithstanding, they primarily utilize quick ordering strategies catchphrases for delivering results. They do not be guaranteed to distinguish the programming language. These strategies can function admirably for getting assets in an exceptional programming language, yet will not work for applications, for example, programmed sentence structure featuring

3rd Chapter

Proposed Methodology & Implementation

CHAPTER 3: PROPOSED METHODOLOGY & IMPLEMENTATION

3.1 Problem Definition

There are many bits of source code shared by clients' famous internet-based discussions devoted to investigating and tackling programming issues related questions like Stack Exchange [45]. Notwithstanding, most web crawlers appear to fundamentally track down assets in unambiguous programming dialects because of the absence of programmed source language discovery. Frequently, most sites and gatherings use dependable guidelines to track down the language by utilizing development and additionally marker-based techniques to recognize catchphrases. However, with substantial number of online discussions accessible, the source language for most assets stays anonymous and they merely show up as pure writing records. It will be very valuable on the off chance that efficient means are utilized to decide the programming language wherein an asset or even a piece of code is composed. This, obviously, would prompt greater code reuse and better web search tool perceivability. Looking for source code motors like SearchCode [43] or Codase [44] can enormously profit from this model having the option to record assets by confining the language. In this paper we propose a Bayesian learning-based model for the revelation of fundamental programming language from a source or piece of code.

This examination plans to foster an instrument for programmed recognition of programming dialects utilizing MATLAB. After thought of numerous information wellspring of programming dialects we develop our own dataset comprising generally around 1000 source codes since we want to play out this undertaking according to novel viewpoint. It contains information including the name of some of programming dialects, source codes, and labels. The total examination technique is made sense of in this part [46].

3.2 Research Methodology

In order to create automatic detection tool for programing languages firstly data from Google, IEEE, ACM, and other internet sources is collected, refined, and saved in an excel file. After that data is loaded into MATLAB tool. Then data is distributed into

test data and train data sets. Moreover, tokenization is done and classifiers' training using knn and ecoc and testing using prediction model is performed [47]. In addition to, confusion matrix is generated to validate the results. At the last phase, the name of programing languages is automatically detected. Research methodology at abstract level is describes in figure below. Each step is then further explained in detail in this chapter.

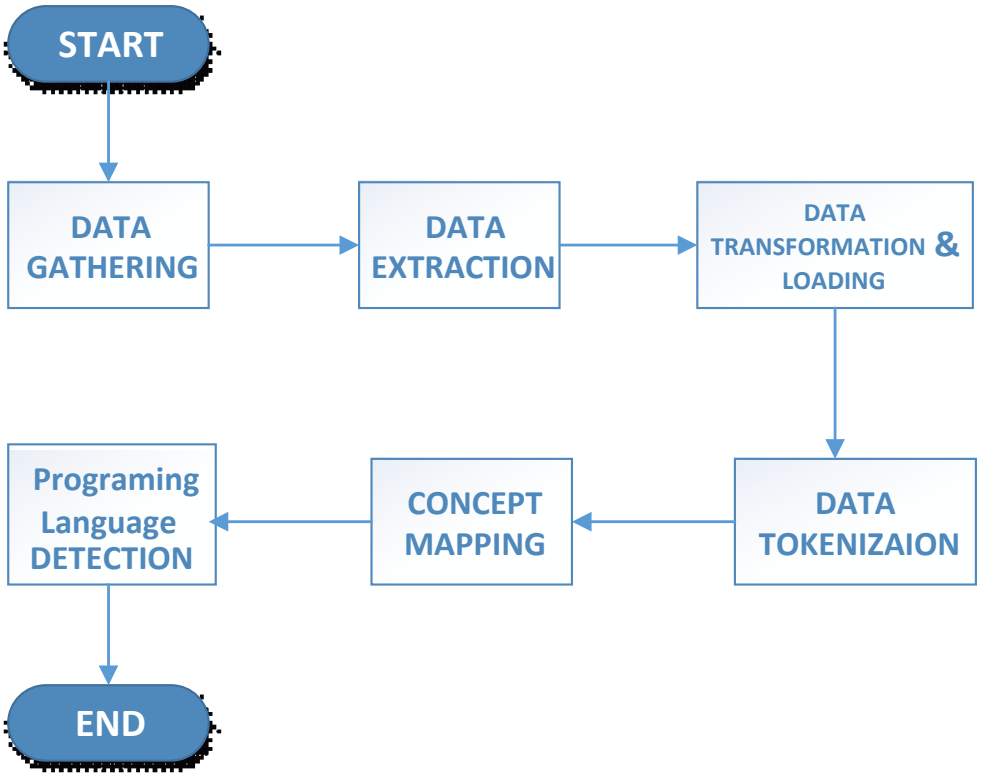

Figure 3.1 (Research Methodology)

3.2.1 Data Gathering

As mentioned above we use Google Scholar, IEEE, ACM, Google, and other internet sources as a data source to generate dataset for this research. Google search serves as a core source as for collection of source codes of different programing languages used in our research. Detail of steps carried out to obtain desirable data from these data banks. This, clearly, would incite more prominent code reuse and better web search apparatus detectable quality. Searching for source code engines like SearchCode [43] or Codase [44] can gigantically benefit from this model having the choice to record resources by restricting the language. In this paper we propose a Bayesian learning-based model for the disclosure of key programming language from a source or piece of code.

3.2.2 Data Extraction

In the wake of setting up the information, Google program used to extricate information from other web assets. There are two delivery arrangements of Google. Transformation framework incorporates both as result of assets. In our review we pick unstructured as result data. Designers and specialists rouse to utilize unstructured data in light of the fact that it offers significant benefits in source jargon straightforwardness and address the total semantics of each source jargon [48]. Additionally, more suitable portrayals of idea name, source, and progressive data (connection). We removed information in unstructured utilizing program to pick our subset. Information extraction brings about GBs of unstructured records. To remove information from Google and convert that information into literary records that will be viable for stacking it into data set, following advances ought to be done.

3.2.3 Data Transformation and Loading

As we have tremendous size extricated information from different data sets which is in unstructured configuration. We have use ETL (Extract, Load, Transform) way to deal with manage Google and different data sets information. The separated information ought to be stacked into a legitimate configuration for recognition. For this reason, we have made dataset and compose a bunch content to stack information from unstructured configuration to organized design. The course of information stacking is displayed in Figure.

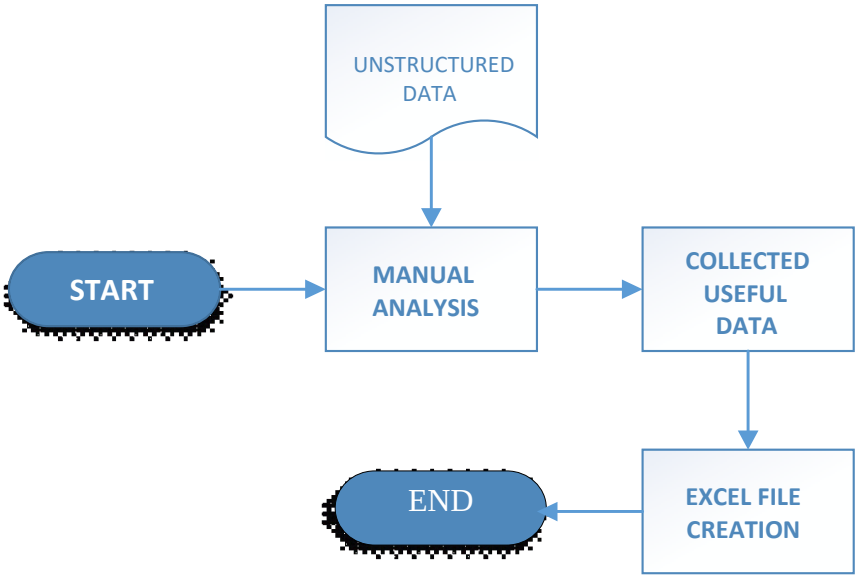

Figure 3.2 (Data transformation & Loading)

3.2.4 Data indexing

In the wake of stacking information into Excel document, stacked records are in thousands and it ought to be filed. Record design of dataset further develops the information recovery procedure on .csv document. Files activity rapidly search the expected information from table without looking through each record of dataset. For this reason, manual investigation has been performed.

3.3 Importing Excel data file into MATLAB tool

To import data into MATLAB tool for automatic detection of programing languages from source code following steps are followed.

3.3.1 Open MATLAB code file in MATLAB Tool.

In MATLAB tool, go to Open file icon and press a click on it. Then go to the folder of source code and select the code file and press enter. In this way code file is opened tool, and ready to run. Before running the file, one needs to the add path of code file to the same destination where tool is running. Now, code can be run and produce desired results.

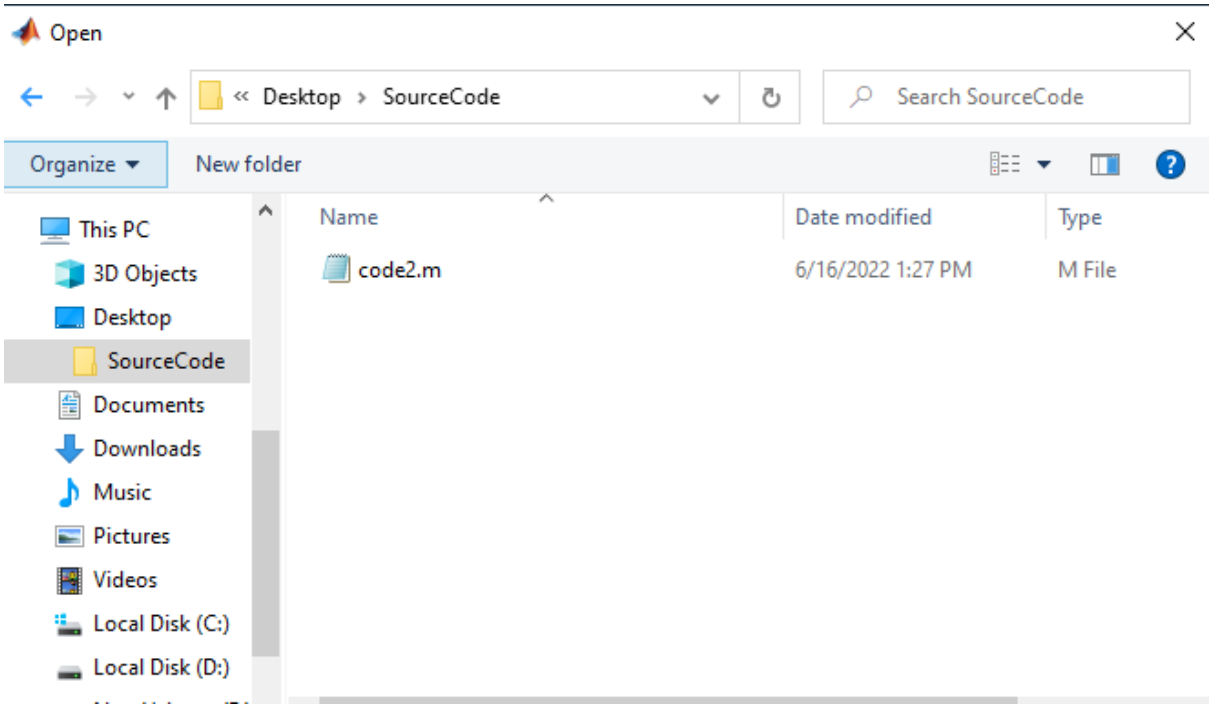

Figure 3.3 (Open code in MATLAB)

3.3.2 How dataset is loaded into the MATLAB

The dataset is loaded by using the data read function of MATLAB language. It is depicted in the following diagram

```
filename = "dataset.csv";
data = readtable(filename,'TextType','string');
head(data)
```

Figure 3.4 (Dataset Loading)

4th Chapter

Results & Discussions

CHAPTER 4: RESULTS & DISCUSSION

After applying all steps of research methodology automatic detection of programing languages' names from their source codes are finally become possible. All required steps: First, loading of data. Secondly, partitioning of datasets into test and training data. Thirdly, transforming the data for hypernyms. Fourthly, tokenized document. Fifthly, usage of remove Stopword function to remove extra spaces for better efficiency. Sixthly, use the bagofWords function. Seventhly, use training and testing classifiers for data classification. Eighthly, plot confusion matrixes to find out the accuracy of the classification. In the end, detect the name of the programing language by entering source code.

4.1 Automatic Detection of Programing Languages and other steps

The figure below is showing the working of our algorithm of automatic detection of programing languages from source code in MATLAB tool.

```
File  Edit  Format  View  Help
Filename = "dataset.csv";
data = readtable(filename,'TextType','string');
head(data)
%%
cvp = cvpartition(data.Class,'Holdout',0.3);
dataTrain = data(cvp.training,:);
dataTest = data(cvp.test,:);

textDataTrain = dataTrain.Text;
textDataTest = dataTest.Text;
YTrainC = dataTrain.Class;
YTestC = dataTest.Class;
YTrainH = dataTrain.Hypernym;
YTestH = dataTest.Hypernym;
%%
documents = tokenizedDocument(textDataTrain);
Documents1 = removeStopWords(documents);
Documents = removeWords(Documents1,"a");
%%
bag = bagOfWords(Documents);
train_features = bag.Counts;
 train_features = full(train_features);
 %%
 documents_text = tokenizedDocument(textDataTest);
 XTest = encode(bag,documents_text);
XTest = full(XTest);

%%
%Classifier Training
tic
 model1 = fitcecoc(train_features,YTrainC);
 model2 = fitcknn(train_features,YTrainH);
toc
%  tic
%    md1 = TreeBagger(50,train_features,YTrain,'OOBPrediction','On',...
%    'Method','classification')
%  toc
%%
%Classifier testing

YPredC = predict(model1,XTest);
YPredH = predict(model2,XTest);
testing_time = toc
sprintf('Testing Time = %f',testing_time)
averge_test_time_per_instance =  testing_time/66;
sprintf('%f',averge_test_time_per_instance)
accC = sum(YPredC == YTestC)/numel(YTestC);
accH = sum(YPredH == YTestH)/numel(YTestH);
%%
plotconfusion(categorical(YTestC),categorical(YPredC))
figure,
plotconfusion(categorical(YTestH),categorical(YPredH))
%%
text1 = ('#include <stdio.h> int main() { int data[100], i, n, small, big;  printf("Enter your input for n:"); scanf("%d", &n); printf("Enter your data inputs:\n"); for (i = 0; i < n; i++)  scanf("%d", &data[i]);  small = big = data[0];   for (i = 0; i < n; i++) { if
tok1 = tokenizedDocument(text1);
tok1x = full(encode(bag,tok1));
testlabel = predict(model1,tok1x)
testhyper = predict(model2,tok1x)
```

Figure 1 Detection Algorithm

4.2 Results of all the functions except Program language detection

Following figure shows the results of all the functions such as bag, cvp, data, dataTest, dataTrain, documents, Documents, Documents1, documents_text, dataset loading, model1, model2, testhyper, testing_time, and testlabel. These are the results of functions which we have used in our code to set a platform the detection of programing language in an efficient way.

bag	1x1 bagOfWords
cvp	1x1 cvpartition
data	110x3 table
dataTest	33x3 table
dataTrain	77x3 table
documents	77x1 tokenizedDocum...
Documents	77x1 tokenizedDocum...
Documents1	77x1 tokenizedDocum...
documents_text	33x1 tokenizedDocum...
filename	"dataset.csv"
model1	1x1 ClassificationECOC
model2	1x1 ClassificationKNN
testhyper	1x1 cell
testing_time	0.4133
testlabel	1x1 cell

Figure 2 Secondary Functions

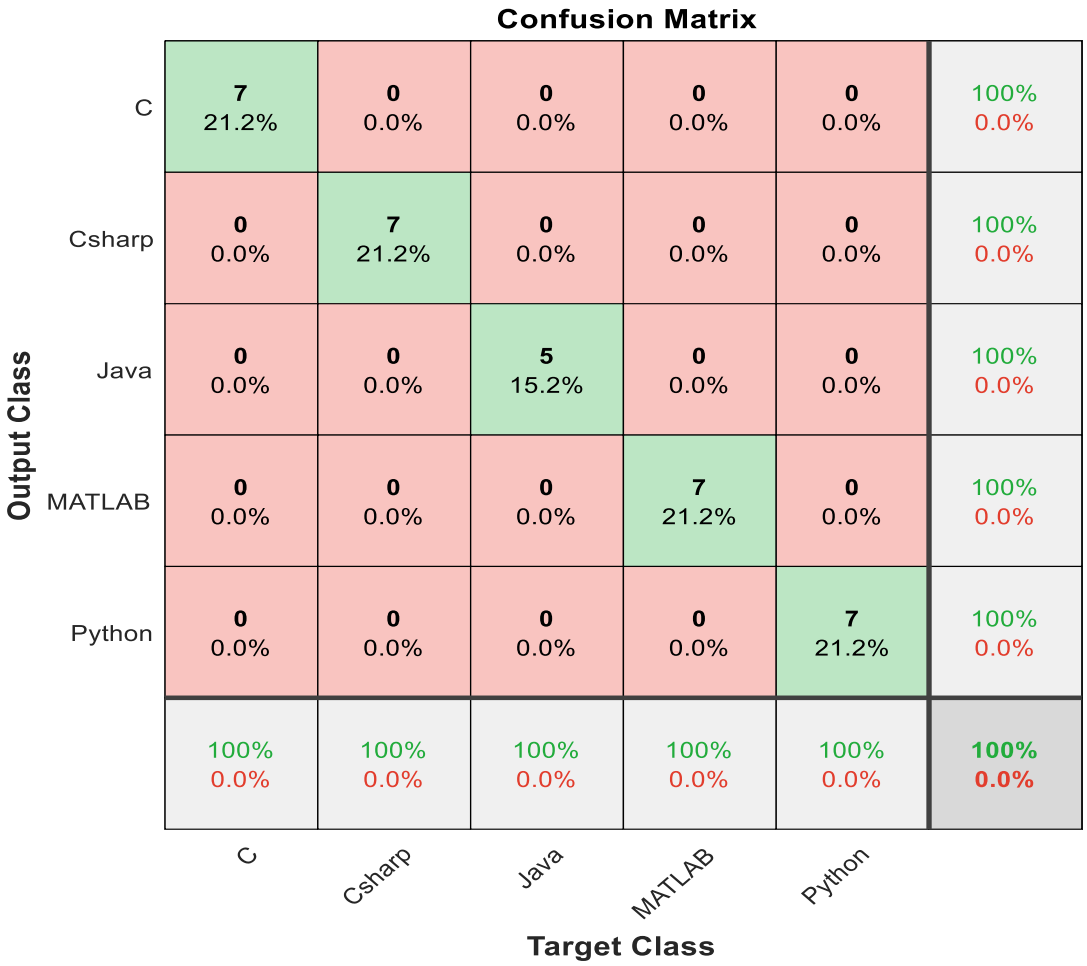

Figure 3 Confusion Matrix: Languages

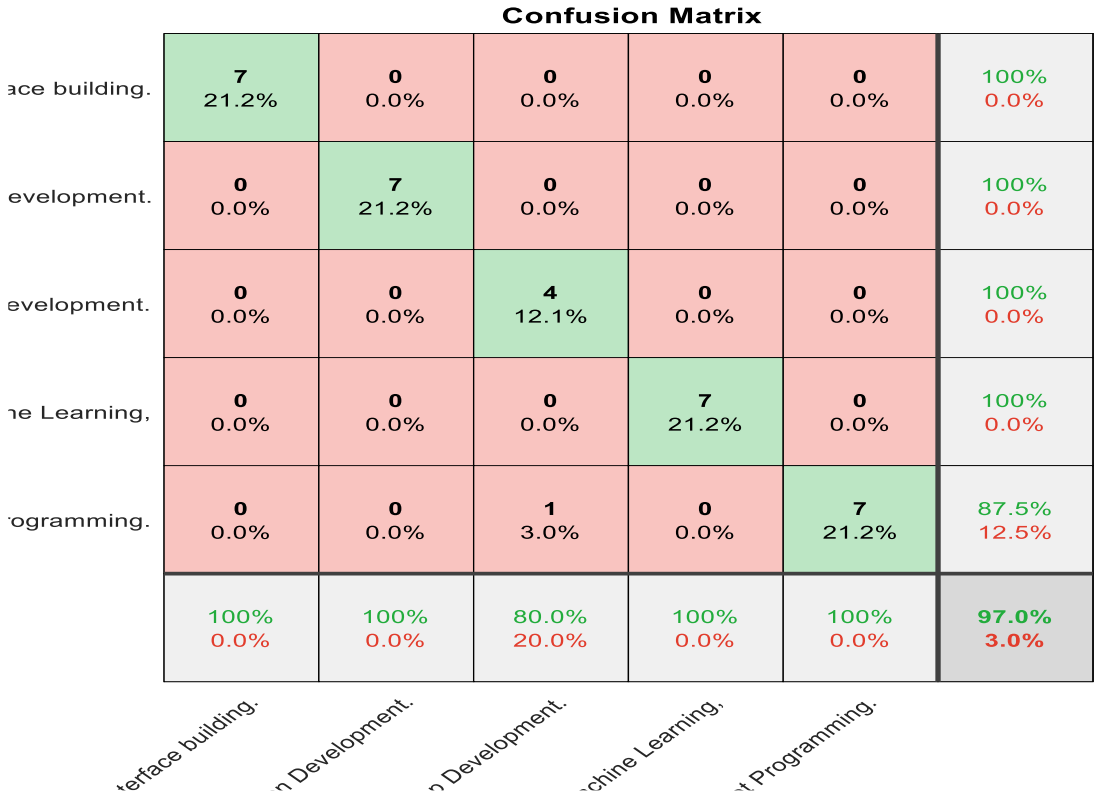

Figure 4 Confusion Matrix: Platforms

4.3 Result for Detection of Programing Language

After showing all the above-mentioned function. Finally, following two figures shows the result for automatic detection of programing language from a source code and second figure shows the accuracy of this algorithm.

```
testlabel =

  1×1 cell array

    {'C'}


testhyper =

  1×1 cell array

    {'Operating System, Robotics, Computer Graphics and Desktop Application Development.'}
```

Figure 5 Detection of Programing Language

accC	0.9967
accH	0.9934

Figure 6 Accuracy for hypernym detection

4.4 Discussion & Comparison

For building our model we collected over 1000 source codes of eight programing languages (C, C#, Python, Java, MATLAB, PHP, HTML, and SQL) from publicly available repositories as Google Scholar, IEEE, ACM, GitHub, and other internet sources. Using these resources, we have created our dataset [49]. A manual testing was finished on the sources to change the unstructured type of information into organized structure. Following two figures show the sample of our final dataset.

A	B	C	D	E	F	G	H	I	J	K
C	#include	Operating System, Robotics, Computer Graphics and Desktop Application Development.								
C	#include	Operating System, Robotics, Computer Graphics and Desktop Application Development.								
C	#include	Operating System, Robotics, Computer Graphics and Desktop Application Development.								
C	#include	Operating System, Robotics, Computer Graphics and Desktop Application Development.								
C	#include	Operating System, Robotics, Computer Graphics and Desktop Application Development.								
C	#include	Operating System, Robotics, Computer Graphics and Desktop Application Development.								
C	#include	Operating System, Robotics, Computer Graphics and Desktop Application Development.								
C	#include	Operating System, Robotics, Computer Graphics and Desktop Application Development.								
C	#include	Operating System, Robotics, Computer Graphics and Desktop Application Development.								
C	#include	Operating System, Robotics, Computer Graphics and Desktop Application Development.								
C	#include	Operating System, Robotics, Computer Graphics and Desktop Application Development.								
C	#include	Operating System, Robotics, Computer Graphics and Desktop Application Development.								
Python	import	Web Services, Data Analysis, Robotics and Machine Learning,								
Python	import	Web Services, Data Analysis, Robotics and Machine Learning,								
Python	import	Web Services, Data Analysis, Robotics and Machine Learning,								
Python	import	Web Services, Data Analysis, Robotics and Machine Learning,								
Python	import	Web Services, Data Analysis, Robotics and Machine Learning,								
Python	import	Web Services, Data Analysis, Robotics and Machine Learning,								
Python	import	Web Services, Data Analysis, Robotics and Machine Learning,								
Python	import	Web Services, Data Analysis, Robotics and Machine Learning,								
Python	import	Web Services, Data Analysis, Robotics and Machine Learning,								

MATLAB	gpa =	Maths, Algorithm develop
MATLAB	a =	Maths, Algorithm develop
MATLAB	vec1 = [3	Maths, Algorithm develop
MATLAB	char =	Maths, Algorithm develop
MATLAB	a = 15;	Maths, Algorithm develop
MATLAB	a=13;b=1	Maths, Algorithm develop
MATLAB	numbers	Maths, Algorithm develop
MATLAB	a=5	Maths, Algorithm develop
MATLAB	a=5;	Maths, Algorithm develop
MATLAB	vec =	Maths, Algorithm develop
MATLAB	a = [12	Maths, Algorithm develop
MATLAB	a = [1 4 4	Maths, Algorithm develop
Csharp	using	Web Sever Side Scripting,
Csharp	using	Web Sever Side Scripting,
Csharp	Console.	Web Sever Side Scripting,
Csharp	using	Web Sever Side Scripting,
Csharp	using	Web Sever Side Scripting,
Csharp	using	Web Sever Side Scripting,
Csharp	using	Web Sever Side Scripting,
Csharp	using	Web Sever Side Scripting,
Csharp	using	Web Sever Side Scripting,

Figure 7 Dataset Sample

Moreover, the following figure again shows our algorithm

```
filename = "dataset.csv";
data = readtable(filename,'TextType','string');
head(data)
%%
cvp = cvpartition(data.Class,'Holdout',0.3);
dataTrain = data(cvp.training,:);
dataTest = data(cvp.test,:);

textDataTrain = dataTrain.Text;
textDataTest = dataTest.Text;
YTrainC = dataTrain.Class;
YTestC = dataTest.Class;
YTrainH = dataTrain.Hypernym;
YTestH = dataTest.Hypernym;
%%
documents = tokenizedDocument(textDataTrain);
Documents1 = removeStopWords(documents);
Documents = removeWords(Documents1,"á");
%%
bag = bagOfWords(Documents);
train_features = bag.Counts;
 train_features = full(train_features);
 %%
 documents_text = tokenizedDocument(textDataTest);
 XTest = encode(bag,documents_text);
XTest = full(XTest);

%%
%Classifier Training
tic
 model1 = fitcecoc(train_features,YTrainC);
 model2 = fitcknn(train_features,YTrainH);
toc
%  tic
%    md1 = TreeBagger(50,train_features,YTrain,'OOBPrediction','On',...
%  'Method','classification')
%  toc
%%
%Classifier testing

YPredC = predict(model1,XTest);
YPredH = predict(model2,XTest);
testing_time = toc
sprintf('Testing Time = %f',testing_time)
averge_test_time_per_instance =  testing_time/66;
sprintf('%f',averge_test_time_per_instance)
accC = sum(YPredC == YTestC)/numel(YTestC);
accH = sum(YPredH == YTestH)/numel(YTestH);
%%
plotconfusion(categorical(YTestC),categorical(YPredC))
figure,
plotconfusion(categorical(YTestH),categorical(YPredH))
%%
text1 = ('#include <stdio.h> int main() { int data[100], i, n, small, big;
tok1 = tokenizedDocument(text1);
tok1x = full(encode(bag,tok1));
testlabel = predict(model1,tok1x)
testhyper = predict(model2,tok1x)
```

Figure 8 Algorithm

5th Chapter

Detection Evaluation

CHAPTER 5: DETECTION ASSESSMENT

5.1 Expert assessment:

For the assessment of any software detection algorithm commonly used methods are expert opinion and automated evaluation. But, for automated detection of the names of programing languages the state-of-the-art system should be available for particular domain. In case of our algorithm there is system available. So, we chose to evaluate our model with that model named as Glasslang. And at the same time also we conducted surveys from experts. 10 software experts were chosen from different software houses of Pakistan and a google form-based questionnaire is forwarded to them to record their opinion. 5-point like scale (Strongly Agree, Agree, Neutral, Disagree, Strongly Agree) is used to evaluated expert opinion about the newly developed detection algorithm. This evaluation covered the accuracy results of classifier and confusion matrixes.

5.1.1 Comparison Evaluation:

Now will briefly provide the comparison with another detection tool. Although this tool is built on large dataset as compared to our dataset, but its accuracy is bit on the lower side then our predicted model. Moreover, Naïve Bayes classifier, Bayesian Network, and Multinomial Naïve Bayes classifiers with accuracy of 82.48%, 89.59%, and 93.48%. Whereas the accuracy of our algorithm is almost 98% as shown in the Figure 6,9,10,and 11 show the confusion matrixes for the algorithm which we are using for comparison with our model and Figure 12 shows the accuracy result of the compared algorithm.

c	java	python	cpp	ruby	html	javascript	objective_c	cs	perl	classified as
168	0	0	11	0	0	1	2	9	2	c
0	290	0	5	0	0	0	1	1	0	java
0	0	231	3	52	0	1	6	6	1	python
0	0	1	175	5	0	11	0	34	4	cpp
0	0	3	1	374	0	9	0	0	1	ruby
0	0	0	0	0	127	0	0	167	0	html
0	2	0	1	3	0	151	0	44	4	javascript
0	0	1	1	0	0	1	217	1	0	objective_c
0	0	0	10	0	0	1	0	140	0	cs
0	0	0	2	0	0	7	0	4	100	perl

Figure 9 Confusion Matrix (for comparison)

c	java	python	cpp	ruby	html	javascript	objective_c	cs	perl	classified as
145	0	0	20	0	0	2	5	11	10	c
1	284	0	2	0	0	1	1	8	0	java
0	0	254	0	26	0	3	1	3	13	python
17	0	1	174	1	0	0	0	31	6	cpp
0	0	4	1	382	0	1	0	0	0	ruby
0	0	0	0	0	294	0	0	0	0	html
0	0	0	1	1	0	138	0	16	49	javascript
1	0	0	0	0	0	2	212	3	3	objective_c
1	0	0	2	0	0	0	0	148	0	cs
0	0	0	1	0	0	0	0	0	112	perl

Figure 10 Confusion Matrix (for comparison)

c	java	python	cpp	ruby	html	javascript	objective_c	cs	perl	classified as
158	8	3	16	0	0	2	4	1	1	c
0	297	0	0	0	0	0	0	0	0	java
0	2	293	2	3	0	0	0	0	0	python
11	7	2	202	0	0	0	0	7	1	cpp
1	1	1	1	378	3	0	0	0	3	ruby
0	0	0	0	0	294	0	0	0	0	html
0	1	5	2	5	3	175	12	0	2	javascript
1	6	4	0	0	0	0	210	0	0	objective_c
0	5	0	19	0	0	7	0	120	0	cs
0	0	0	0	2	2	0	0	0	109	perl

Figure 11 Confusion Matrix (for comparison)

It shows the disorder network for the Naive Bayes Classifier. The confusion grid shows the quantity of sources that were precisely portrayed for each language what is more, the quantity of sources that were misclassified. Out of 2392 source archives which we used as test data; 1973 records were precisely described giving a precision of 82.48%. Likewise, shows the confusion lattice for the Bayesian Network Classifier. The Bayesian Network Classifier is out and out more definite over the Naïve Bayes Model. Out of 2392 records used for test data, 2143 source archives have been precisely recognized and 249 records have been incorrectly perceived giving a precision of 89.59%. Also, they show the disorder framework resulting to using Multinomial Credulous Bayes Classifier. The results show that Multinomial Naive Bayes Classifier has performed better contrasted with Bayesian Network Classifier model. This consequently spreads out the transcendence of Multinomial Naive Bayes model over Naive Bayes model. Out of 2392 archives which were used as test data, MNB Classifier is skilled to precisely organize 2236 source records, where only 156 reports have been mistakenly portrayed. This furnishes us with the precision of 93.48%, which is the best result among all the Bayesian classifier models used in their preliminary [50].

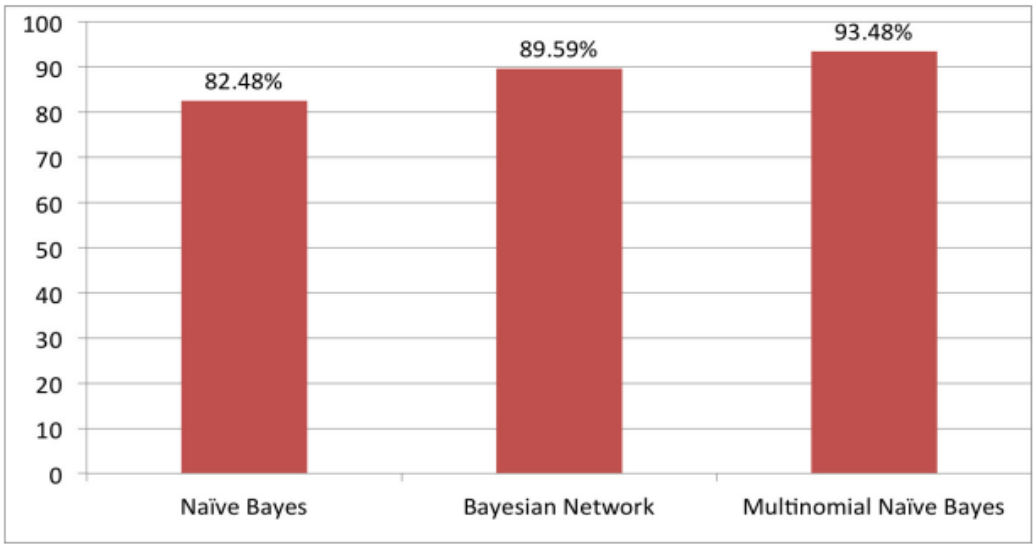

Figure 12 Accuracies (for comparison)

On the other hand, we used KNN and COC classifiers for the development of our models. The following figure shows accuracy of our models

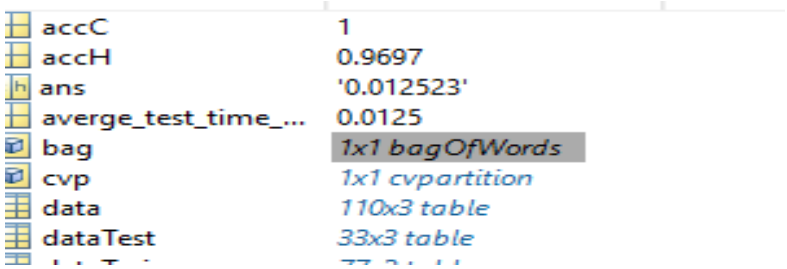

Figure 13 Accuracies

Although, our models are showing better accuracy as compared to above mentioned classifiers, but we know that our dataset is much smaller than the dataset used for the research with which we are comparing. In addition to, our research is also doing tokenization and remove stopword which are not only beneficial for automatic detection of programing languages from source code, but their main advantage will be shown in our future research on automatic categorization of software technologies. Confusion matrixes of our model are already provided under the chapter of results and discussion.

6th Chapter

Conclusion & Future Work

CHAPTER 6: CONCLUSION & FUTURE WORK

6.1 Conclusion

In this paper we introduced a calculation for programmed location of programing dialects from source code. Taking source codes of a programing language as an information and giving result the names of the programming dialects of the particular information. Because of unstructured type of writing and information accessible on web and information archives in regard to this subject, it is hard for scientists and clients to handily deal with such an information. Order is the most effective way to foster conventional information base of unstructured information in a proper manner by considering source code of various programming dialects and relationship of information accessible. Identification of programing dialects related research somewhat increments as of late as it is not difficult to foster master frameworks or man-made reasoning-based frameworks utilizing crude information of colossal measure of source codes. In this exploration Resource portrayal structure information with respect to the source code of programming dialects, for example, C, C++, Python and so on has been gathered and utilized for fostering a framework which not just recognize the name of the language whose source code is placed, yet in addition guarantees most extreme feasible degree of precision.

6.2 Future Work

As we do not guarantee that our answer is all around unrivaled to existing scientific classification devices. For sure, it was created with the objective of performing great for the product area, and therefore it encodes numerous product explicit guidelines. By the by, the assessment gives us certainty that to consequently classify programming advancements, Witt is presently the most ideal choice that anyone could hope to find. In future the developed detection algorithm can be used to develop an algorithm for much bigger dataset then what we have collected and developed for our research work. Another future dimension is incorporation of multiple languages for expanding the scope of our detection algorithm for helping the people who want to use automatic detection algorithm for programing languages. Furthermore, we will also work on another aspect of this research which is automatic categorization of software technologies. Here in this paper,

we just give very brief touch of this topic. For this purpose, we will integrate our developed dataset with other bigger chunk of data.

References

REFERENCES

[1] S. Kawaguchi, P. G. (2005). MUDABlue: an automatic categorization system for open-source repositories. 11th Asia-Pacific Software Engineering Conference. Busan, Korea (South): IEEE.

[2] Ramona-Cristina, P., Vasilateanu, A., & Goga, N. (2016). Ontology based multi-system for SME knowledge workers. 2016 IEEE International Symposium on Systems Engineering (ISSE). doi: 10.1109/syseng.2016.7753132

[3] Weikum, Gerhard, and Martin Theobald. "From information to knowledge: harvesting entities and relationships from web sources." Proceedings of the twenty-ninth ACM SIGMOD- SIGACT-SIGART symposium on Principles of database systems. ACM,(2010).

[4] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, “Introduction to Wordnet: An on-line lexical database,” International Journal of Lexicography, vol. 3, no. 4, pp. 235–244, 1990

[5] Rabie, O., & Norcio, A. (2013). Discussion of Some Challenges Concerning Biomedical Ontologies. Human-Computer Interaction. Applications And Services, 173-180. doi: 10.1007/978-3-642-39262-7_20

[6] Bouchiha, D., & Malki, M. (2010). Towards re-engineering Web applications into Semantic Web Services. 2010 International Conference on Machine and Web Intelligence. doi: 10.1109/icmwi.2010.5648057

[7] Software categories - Wikipedia. (2022). Retrieved 7 July 2022, from https://en.wikipedia.org/wiki/Software_categoriesFlouris, Giorgos, et al. "Ontology change: Classification and survey." Knowledge Engineering Review 23.2 (2008): 117-152.

[8] Mathieu Nassif, C. T. (JANUARY 2020). Automatically Categorizing Software Technologies. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 46, NO. 1,.

[9] Sidramappa, K. M. (2019). A Novel Approch Automatically Categorizing Software Technologies. International Research Journal of Engineering and Technology (IRJET) .

[10] [4] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, “Introduction to Wordnet: An on-line lexical database,” International Journal of Lexicography, vol. 3, no. 4, pp. 235–244, 1990

[11] [5] M. A. Hearst, “Automatic acquisition of hyponyms from large text corpora,” in Proceedings of the 14th Conference on Computational Linguistics, 1992, pp. 539–545.

[12] [6] R. Snow, D. Jurafsky, and A. Y. Ng, “Learning Syntactic Patterns for Automatic Hypernym Discovery,” in Proceedings of the 18th Annual Conference on Neural Information Processing Systems, 2004

[13] [10] M. Dojchinovski and T. Kliegr, “Entityclassifier.eu: Real-time classification of entities in text with wikipedia,” in Machine Learning and Knowledge Discovery in Databases, ser. Lecture Notes in Computer Science, H. Blockeel, K. Kersting, S. Nijssen, and F. elezn, Eds. Springer, 2013, vol. 8190, pp. 654–658

[14] [11] T. Kliegr, V. Svatek, K. Chandramouli, J. Nemrava, and E. Izquierdo, “Wikipedia as the premiere source for targeted hypernym discovery,” in Proceedings of the ECML PKDD Workshop Wikis, Blogs, Bookmarking Tools: Mining the Web 2.0, 2008

[15] [12] I. Yamada, K. Torisawa, J. Kazama, K. Kuroda, M. Murata, S. D. Saeger, F. Bond, and A. Sumida, “Hypernym Discovery Based on Distributional Similarity and Hierarchical Structures,” in Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2009, pp. 929–937

[16] [13] Z. Kozareva and E. Hovy, “A semi-supervised method to learn and construct taxonomies using the web,” in Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2010, pp. 1110–1118

[17] [14] B. B. Dalvi, W. W. Cohen, and J. Callan, “WebSets: Extracting sets of entities from the web using unsupervised information extraction,” in Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, 2012, pp. 243–252

[18] [15] T. Flati, D. Vannella, T. Pasini, and R. Navigli, “Two is bigger (and better) than one: the Wikipedia Bitaxonomy Project,” in Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014

[19] [16] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann, “DBpedia - a crystallization point for the web of data,” Web Semantics:

Science, Services and Agents on the World Wide Web, vol. 7, no. 3, pp. 154–165, Sep. 2009

[20] [17] P. N. Mendes, M. Jakob, A. Garc´ıa-Silva, and C. Bizer, “DBpedia Spotlight: Shedding light on the web of documents,” in Proceedings of the 7th International Conference on Semantic Systems, 2011, pp. 1–8

[21] [18] W. Wu, H. Li, H. Wang, and K. Q. Zhu, “Probase: A probabilistic taxonomy for text understanding,” in Proceedings of the ACM SIGMOD International Conference on Management of Data, 2012, pp. 481–492

[22] [19] J.-R. Falleri, M. Huchard, M. Lafourcade, C. Nebut, V. Prince, and M. Dao, “Automatic extraction of a WordNet-like identifier network from software,” in 18th IEEE International Conference on Program Comprehension (ICPC), 2010, pp. 4–13

[23] [20] J. Nonnen, D. Speicher, and P. Imhoff, “Locating the meaning of terms in source code: Research on ”term introduction”,” in Proceedings of the 18th Working Conference on Reverse Engineering, 2011, pp. 99–108.

[24] [21] M. F. Porter, “An algorithm for suffix stripping,” Program, vol. 14, no. 3, pp. 130–137, 1980

[25] [27] C. Treude and M.-A. Storey, “Work item tagging: Communicating concerns in collaborative software development,” IEEE Transactions on Software Engineering, vol. 38, no. 1, pp. 19–34, 2012

[26] [24] S. Wang, D. Lo, and L. Jiang, “Inferring semantically related software terms and their taxonomy by leveraging collaborative tagging,” in Proceedings of the 28th International Conference on Software Maintenance, 2012, pp. 604–607

[27] [28] X. Li, H. Wang, G. Yin, T. Wang, C. Yang, Y. Yu, and D. Tang, “Inducing taxonomy from tags: An agglomerative hierarchical clustering framework,” in Advanced Data Mining and Applications, ser. Lecture Notes in Computer Science, S. Zhou, S. Zhang, and G. Karypis, Eds. Springer Berlin Heidelberg, 2012, vol. 7713, pp.64–77

[28] [29] T. Wang, H. Wang, G. Yin, C. X. Ling, X. Li, and P. Zou, “Tag recommendation for open-source software,” Frontiers of Computer Science, vol. 8, no. 1, pp. 69–82, 2014

[29] [30] D. Lo, L. Jiang, and F. Thung, “Detecting similar applications with collaborative tagging,” in Proceedings of the International Conference on Software Maintenance, 2012, pp. 600–603

[30] [31] C. Treude, O. Barzilay, and M.-A. Storey, “How do programmers ask and answer questions on the web? (NIER Track),” in Proceedings of the 33rd International Conference on Software Engineering, 2011, pp. 804–807

[31] [32] A. K. Saha, R. K. Saha, and K. A. Schneider, “A discriminative model approach for suggesting tags automatically for Stack Overflow questions,” in Proceedings of the 10th Working Conference on Mining Software Repositories, 2013, pp. 73–76

[32] [33] C. Stanley and M. D. Byrne, “Predicting tags for StackOverflow posts,” in Proceedings of the 12th International Conference on Cognitive Modelling, 2013, pp. 414–419

[33] [19] Jia, M., Bing-ru, Y., De-quan, Z., & Wei-cong, S. (2009). Research on Domain Ontology Construction in Military Intelligence. 2009 Third International Symposium on Intelligent Information Technology Application. doi: 10.1109/iita.2009.80

[34] [20] Wisniewski, D., Potoniec, J., Ławrynowicz, A., & Keet, C. (2019). Analysis of Ontology Competency Questions and their formalizations in SPARQL-OWL. Journal Of Web Semantics, 59, 100534. doi: 10.1016/j.websem.2019.100534

[35] [21] Horridge, M., Tudorache, T., Nuylas, C., Vendetti, J., Noy, N. and

[36] [23] Musen MA. AMIA 2008 tutorial T26: ontologies in biomedicine. Washington, DC, November 9; 2008.

[37] SNOMED CT http://www.snomed.org/

[38] Google Code Prettify, http://code.google.com/p/google-code-prettify/

[39] SyntaxHighlighter, http://alexgorbatchev.com/SyntaxHighlighter/

[40] Highlight.js, http://highlightjs.org

[41] SourceClassifier, https://github.com/chrislo/sourceclassifier

[42] Klein, D., Murray, K., Weber, S.: Algorithmic programming language identification. CoRR abs/1106.4064 (2011)

[43] SearchCode, http://searchcode.com

[44] Codase, http://codase.com

[45] Stack Exchange, http://stackexchange.com

[46] Sousa, D., & Couto, F. (2020). BiOnt: Deep Learning Using Multiple Biomedical

Ontologies for Relation Extraction. Lecture Notes in Computer Science, 367-374. doi: 10.1007/978-3-030-45442-5_46

[47] Maurice, P., Dhombres, F., Blondiaux, E., Friszer, S., Guilbaud, L., Lelong, N., Khoshnood, B., Charlet, J., Perrot, N., Jauniaux, E., Jurkovic, D. and Jouannic, J. (2017). Towards ontology-based decision support systems for complex ultrasound diagnosis in obstetrics and gynecology. Journal of Gynecology Obstetrics and Human Reproduction, 46(5), pp.423-429.

[48] Runnan Liu, Puwei Wang, & Shixian Zheng. (2012). A Framework for Personalized Management of Domain Ontologies. 2012 IEEE/ACIS 11Th International Conference on Computer and Information Science. doi: 10.1109/icis.2012.6

[49] Silva, F., & Girardi, R. (2014). An Approach to Join Ontologies and Their Reuse in the Construction of Application Ontologies. 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) And Intelligent Agent Technologies (IAT). doi: 10.1109/wi-iat.2014.66

[50] Ianni, G., Martello, A., Panetta, C., & Terracina, G. (2008). Efficiently Querying RDF(S) Ontologies with Answer Set Programming. Journal Of Logic and Computation, 19(4), 671-695. doi: 10.1093/logcom/exn043