**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

“Jnana Sangama”, Belagavi-590014, Karnataka



**An Internship Report**

Submitted in the partial fulfillment of the Academic requirement for the Second Year of

**Bachelor of Engineering**

**In**

**Electronics & Communication**

Submitted by:

**S HARINI**

**[1EE21EC039]**

Under the guidance of



1968

EAST WEST

**Department of Electronics and Communication Engineering**

**EAST WEST COLLEGE OF ENGINEERING, BANGALORE-560064**

**2023-2024**

# **Detect Questions using Python**

## **ABSTRACT**

There are several ways to detect questions using Python, and the appropriate approach depends on the specific requirements and constraints. Here are a few options:

* **Rule-based approaches**: One way to detect questions is to use handcrafted rules to identify question words, such as "what," "when," "where," "why," and "how". These rules can be combined with other linguistic features, such as sentence structure and punctuation, to increase accuracy.
* **Machine learning models**: Another option is to train machine learning models on labeled datasets to classify sentences as questions or statements. Supervised machine learning algorithms, such as logistic regression, decision trees, random forests, and support vector machines, can be used for this purpose. Deep learning models, such as recurrent neural networks (RNNs) and transformer-based models, can also be employed to capture complex patterns in language.
* **Pre-trained language models**: Recent advances in natural language processing (NLP) have led to the development of powerful pre-trained language models, such as BERT and RoBERTa, which can be fine-tuned for various NLP tasks, including question detection. These models can achieve high accuracy without requiring extensive feature engineering or domain-specific knowledge.
* **Hybrid approaches**: Finally, hybrid approaches combining rule-based and machine learning techniques can also be explored. For example, rule-based methods can be used to identify candidate questions, which are then fed into a machine learning model for final classification.

In this abstract, Detecting Questions using Python requires careful consideration of the specific requirements and constraints. A combination of rule-based and machine learning techniques can provide robust and accurate solutions.

# **Detect Questions using Python**

# **INTRODUCTION**

Detecting questions in natural language text involves identifying whether a sentence is interrogative or not. In Python, we can achieve this by leveraging various techniques, including regular expressions, machine learning, and natural language processing (NLP). Here's a brief overview of each approach:

* **Regular Expressions (Regex):** Regex can be used to identify questions based on their grammatical structure. For instance, we can look for sentences that begin with question words like "what", "when", "why", "who", or "how". However, this approach has limitations, as it cannot handle complex questions or sentences that don't follow standard grammar rules.
* **Machine Learning:** Machine learning algorithms can be employed to classify sentences as questions or non-questions. We can train a classifier using labeled datasets, where each sentence is annotated with its corresponding label (question or non-question). Common machine learning approaches for this task include decision trees, random forests, Support Vector Machines (SVM), and neural networks.

* **Natural Language Processing (NLP):** NLP provides a more sophisticated way to detect questions. We can use NLP libraries like NLTK or spaCy to perform tasks such as tokenization, part-of-speech tagging, named entity recognition, and dependency parsing. These processes allow us to analyze the syntax and semantics of a sentence, enabling us to identify questions more accurately.

In Python, we can implement question detection using the following steps:

**Step 1:** Data Collection Gather a dataset of labeled sentences, where each sentence is annotated as a question or non-question. This dataset will serve as the foundation for our machine learning or NLP models.

**Step 2:** Preprocessing Clean and preprocess the collected data by performing the following operations:

**a.** **Tokenization:** Split the sentences into individual words or tokens.

**b. Stopword removal**: Remove common stopwords like "the", "and", "is", etc., which don't contribute much to the meaning of the sentence.

**c. Lemmatization:** Convert words to their base or dictionary form (e.g., "running" becomes "run").

**d. Part-of-speech tagging:** Identify the parts of speech (noun, verb, adjective, adverb, etc.) for each word in the sentence.

**Step 3:** Feature Extraction Extract relevant features from the preprocessed data that can help distinguish between questions and non-questions. Some common features used in question detection include:

**a. Word frequency:** Calculate the frequency of each word in the sentence. Questions tend to have higher frequencies of question words like "what", "when", "why", etc.

**b. Sentiment analysis:** Analyze the sentiment of the sentence. Questions tend to have a neutral or curious sentiment.

**c. Dependency parsing:** Analyze the grammatical structure of the sentence, focusing on the relationships between words. Questions typically have a different dependency structure than statements.

**Step 4:** Model Training Train a machine learning or NLP model using the extracted features and labeled dataset. Popular choices for question detection include logistic regression, decision trees, Random Forest, SVM, and neural networks.

**Step 5:** Evaluate the performance of the trained model using metrics like accuracy, precision, recall, F1-score, and AUC-ROC. Fine-tune the model as needed to improve its performance.

**Step 6:** Deployment Deploy the trained model in a suitable application or platform, such as a chatbot, virtual assistant, or sentiment analysis tool.

**Here we import the Regular Expressions (regex/re) to write a code on Detecting Questions using Python.**

# **Detect Questions using Python**

## **COMPANY PROFILE**

* 1. **Formation of Company:**

AQMENZ Automation Private Limited is a private incorporated on 15th October 2018. It is classified as non-Govt Company and is registered at Registrar of companies, Bangalore.

* 1. **Brief history of company:**

AQMENZ Automation Pvt Ltd (AAPL)was started on October 2018. It is situated in northern part of Bangalore, RT Nagar Karnataka. AAPL provides Mechanical Design & Automation solutions to their client companies. AAPL also involved in Open-source Robotics and developed different varieties of Robots.

AQMENZalso started INDOSKILL, a separate platform for the students to get training and work on various Real Time Industrial Projects. Indoskill offers skill-oriented Handson training through an online platform.

Field of Expertise**:** Open-source Robotics, Industrial Automation, Product Design, Python and Deep Learning and Embedded Systems.

* 1. **Major Milestones:**

We have under gone many industrial projects. Our major clients are BIAL(Bangalore International Airport Limited), GE (General Electric) and Amics technologies.

**1.4 Vision and Mission:**

Our Motto and Vision are to create awareness & training young generation to current and future jobs demands and also help to current and future jobs demands; meanwhile help the students and employees to meet the mandatory necessities of future human resources and skill demands. We are in the 4th industrial revolution. The technological revolution is catastrophic like never before, hence continues awareness for the up-gradation environment is much essential. Aqmenz Automation Pvt. Ltd. is working to help and enhance the potential of students and employees. So that future human resources will be very beneficial, purposeful and profitable to the nation.

## **Objectives:**

* AAPL had a trust in Skill India mission & vision, hence our utmost priority is to add skill to the young Generation and make them Profitable and productive for the nation.
* We aim in Providing Industrial Automation Training Skill module kits to Institution University’s & Collage Lab Facilities with Lowest Possible Price for Benefits of Technical Students.
* Identifying young entrepreneurs and motivate, training them to establish Start-up to create Employment as well as prosperity for the nation.
* Consultation, Sourcing and supplying highly skilled Manpower to Industry for better efficiency and productivity.
* Providing low cast & precise industrial automation solutions.
* Very eager to fetch solution for most complex industrial problems in

a mode.

## **1.6 About the company :**

Organization structure The organization structure is having three different departments such as design department, software department and sales and marketing.

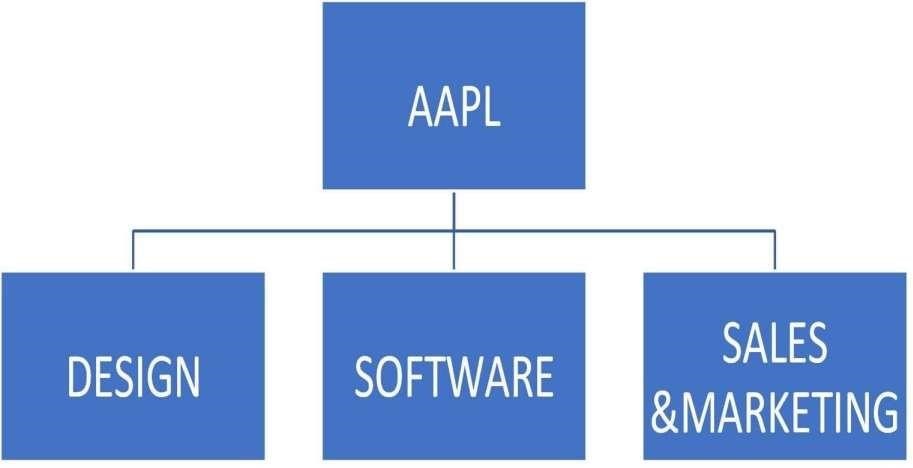


Figure 1.1 Organization structure

## **1.7 Services offered:**

* Provides Design & Automation solutions.
* All type of automation projects to companies using PLC’s, SCADA embedded systems.
* We provide robots and robotic solutions to small and medium scale companies.
* Embedded solutions to companies like GE.
* We conduct technical skill-oriented training programs to engineering colleges.
* We also provide robotics and automation lab equipment’s for colleges.

Number of people working in company and their responsibilities

There are 20 people in this company, out of which:

* Shamanna Mohan, Chief Executive Officer (CEO)
* Mohammed Azhar Hussain, Chief Technology Officer (CTO)

**1.8 Ongoing projects :**

* Automation related projects
* CNC Machines
* Open-source Custom Robots
* Garment Industry slider project

# **Detect Questions using Python**

## **SYSTEM ANALYSIS**

## **Existing System:**

* **NLTK library:** NLTK (Natural Language Toolkit) is a popular Python library used for natural language processing tasks, including question detection. It provides various tools and resources for tokenizing, stemming, lemmatization, and tagging words in a sentence. Developers can use NLTK to develop algorithms that detect questions based on grammatical patterns, keyword extraction, or machine learning models.
* **spaCy library:** spaCy is another modern Python library for natural language processing that offers high-performance, streamlined processing of text data. It includes features such as tokenization, part-of-speech tagging, named entity recognition, and dependency parsing. Developers can leverage spaCy's capabilities to build question detection models that focus on syntax and semantic patterns.
* **Stanford CoreNLP:** While not exclusively a Python tool, Stanford CoreNLP is a widely-used Java library for natural language processing that can be integrated with Python scripts. It encompasses various NLP tools, including question detection. By leveraging Stanford CoreNLP's APIs in their Python code, developers can implement question detection functionality with ease.
* **Pycker:** Pycker is a lightweight Python library designed specifically for building question answering systems. Although it doesn't provide direct question detection functionalities, Pycker offers utilities for handling FAQ pages, extracting questions from text, and generating answers via a knowledge graph. Developers can incorporate Pycker into their projects to construct robust question-answering systems.
* **Pre-trained language models:** Recent advancements in transformer-based language models, such as BERT, RoBERTa, and XLNet, have demonstrated remarkable performance in various NLP tasks, including question detection. These models can be fine-tuned and integrated into Python scripts to classify questions vs. non-questions or even generate answers. Utilizing pre-trained language models allows developers to leverage the knowledge learned from extensive datasets and enhance their question detection capabilities.

These libraries and frameworks offer efficient ways to implement question detection in Python. Developers can choose the most suitable option based on their project requirements and complexity.

**Limitations of this existing system includes:**

* Limited accuracy.
* Contextual Dependence.
* Limited Domain Language.
* Maintenance and Updates.

## **Proposed System:**

* **Preprocessing:** Clean the text data by removing punctuation, converting to lowercase, and tokenizing the text into individual words or phrases.
* **Tokenization:** Split the text into individual tokens, such as words or phrases, to analyse each unit separately.
* **Dependency parsing:** Analyse the grammatical structure of each sentence and identify the relationships between tokens, such as subject-verb-object relationships.
* **Named entity recognition:** Identify named entities in the text, such as names of people, places, organizations, dates, times, etc.
* **Sentiment analysis:** Analyse the sentiment of the text, such as positive, negative, neutral, to determine the tone of the question.
* **Question detection:** Use a machine learning model, such as a Support Vector Machine (SVM), Random Forest, or Convolutional Neural Network (CNN), to classify each token or phrase as a question or non-question. Train the model using a labelled dataset of questions and non-questions.

**The advantages of this proposed system include:**

* Improved accuracy.
* Efficient processing.
* Scalability.
* Customizability.
* User-friendliness.

**The potential risks and challenges of this proposed system include:**

* Complexity.
* Ambiguity.
* Contextual dependence.
* Limited domain knowledge.
* Ethical considerations.

# **Detect Questions using Python**

## **PROJECT WORKING AND FLOW**



Figure 1.2

1. This program is written in Python and it imports library: re (Regular Expression operations).
2. The function **is\_question** takes a string argument, **sentence**.
3. Inside the function, we first check if the sentence ends with a question mark using the **endswith** method. If it does, we return **True**.
4. Next, we create a list of wh-words (words that start a question). We iterate over this list and check if any of these words are present in the sentence (ignoring the case). If any of these words are found, we return **True**.
5. If the sentence passes none of the above conditions, the function returns **False**, indicating that the sentence is not a question.
6. The function is then tested with different strings to verify its accuracy.

# **Detect Questions using Python**

## **SOURCE CODE WITH EXPLANATION OF EACH BLOCK OR STEP**

1. **Importing ‘re’:** The program starts by importing ‘Regular Expression (re)’ (fig.1.3) which is widely used and is one of the type of Natural Language Programming (NLP).  It is used to locate or validate specific strings or patterns of text in a sentence, document, or any other character input.



Figure 1.3

1. **Using ‘def ()’:** In Python, defining ‘def’ (fig.1.4) is the keyword for defining a function which is then followed by parameter(s) in the parenthesis ().



Figure 1.4

1. **Using ‘if ()’:** In a Python program, the ‘if’ statement is used to perform decision-making conditions. In the below program, if statement checks whether the given condition is TRUE or FALSE.

* If a given sentence is ending with a interrogation mark (?), then the output returns “TRUE” (fig.1.5).

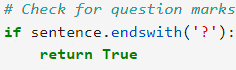
****

Figure 1.5

* There is also an another condition followed by ‘for loop’ if a given sentence starts with or even contains any of the interrogation words [‘who’, ‘where’, ‘when’, ‘whom’, 'how’, ‘what’] then the output returns “TRUE”. The interrogation words are defined as shown in the figure 1.6.
* Otherwise, returns “FALSE”.

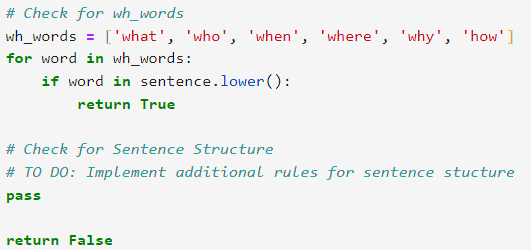


Figure 1.6

1. **Using looping statement:** Looping statement allows programmers to modify the flow of the program so that rather than writing the same code, again and again, programmers are able to repeat the code a finite number of times.

Here ‘for loop’ is used which repeats the execution of the program until the sentence contains the interrogation words (from “wh\_words"). If the words are present from the defined “wh\_words” in the sentence, then it returns “TRUE”, otherwise the looping statement/ the condition applied in the looping statement passes out and returns “FALSE”.

These conditions can be observed in the figure 1.6.

# **Detect Questions using Python**

## **RESULTS**

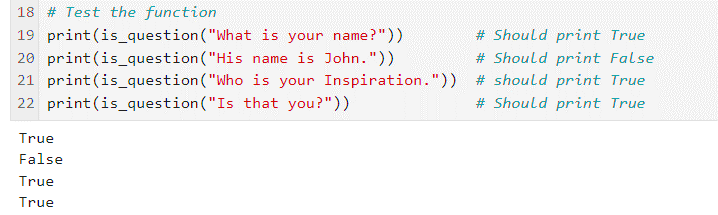


Figure 1.7

The result of the program is shown in the fig. 1.7. After using all the required libraries, looping statements and decision making statements we need to check the working and the flow of the program whether the results are matching to the expected results or not.

* As we see in line 19 (fig.1.7), we are checking for the first condition that is whether the string ends with interrogation mark (?). When we enter the string “What is your name?”, the program starts executing by checking each steps of it. After going through the program it prints “TRUE” due to the presence of the interrogation mark. Here we have the presence of both interrogation word and symbol.
* When we look at the line 20 (fig. 1.7), we have given a sentence “His name is John”. Again this string is crosschecked. After the process, it results “FALSE” due to the absence of both question mark and the interrogation words.
* Now in the line 21 (fig.1.7), even we don’t have the interrogation symbol, the result will be “TRUE” due to the presence of the interrogation word [Who].
* Now this condition is opposite of the above condition that is, even we don’t have the interrogation word due to the presence of the interrogation symbol the result will be printed as “TRUE” (Refer line 22, fig. 1.7).
* Overall, this is how the program is traced and executed.

# **Detect Questions using Python**

## **ADVANTAGES**

1. **Easy to learn:** Python is a relatively easy language to learn, especially compared to other languages commonly used for natural language processing tasks, such as R or MATLAB. This makes it a great choice for beginners who want to get started with building question detection models.
2. **Large community:** Python has a large and active community of developers who share their knowledge and expertise online. This means that there are many resources available for learning and troubleshooting, making it easier to develop and improve question detection models.
3. **Extensive libraries:** Python has a vast array of libraries and frameworks that make it well-suited for natural language processing tasks. Libraries like NLTK, spaCy, and scikit-learn provide efficient and effective tools for text processing, feature engineering, and machine learning.
4. **Scalability:** Python is capable of handling large amounts of data and performing complex computations, making it suitable for scaling up question detection models to larger datasets.
5. **Integration with other technologies:** Python can be easily integrated with other technologies, such as web scrapers, chatbots, and voice assistants, allowing for seamless integration of question detection models into various applications.
6. **Cost-effective:** Python is an open-source language, which means that it is free to use and distribute. This makes it a cost-effective solution for developing question detection models, especially for individuals and organizations with limited budgets.
7. **Flexibility:** Python is a versatile language that can be used for a wide range of tasks beyond natural language processing, including data analysis, visualization, and machine learning. This flexibility makes it a great choice for building multifaceted models that can handle multiple tasks simultaneously.
8. **Improved readability:** Python's syntax is designed to be readable and easy to understand, which makes it simpler to interpret and debug code, especially for those who are new to programming.

## **DISADVATAGES**

1. **Complexity:** The system requires a significant amount of computational resources and expertise to develop and maintain.
2. **Limited domain knowledge:** The system may not perform well on questions outside of its training data or domain knowledge.
3. **Slow performance:** Python is an interpreted language, which means it can be slower than compiled languages like C++ or Java. This can be a challenge when working with large datasets or complex computations, which may require faster processing times.
4. **High memory consumption:** Python's memory consumption can be high due to its flexible data types, which can lead to memory issues when handling large datasets.
5. **Limited support for low-level operations:** Python is a high-level language and lacks support for low-level operations, which can make it less efficient for certain tasks.
6. **Lack of static typing:** Python is dynamically typed, which means that type checking occurs at runtime rather than compile time. This can lead to errors that only appear during runtime and make it more difficult to catch type-related errors early in the development process.

Overall, there are both advantages and disadvantages for Detecting questions using Python.

**Detect Questions using Python**

## **CONCLUSION**

This project demonstrates a basic approach to question detection using Python.

The article discusses the importance of accuracy and effectiveness in automation testing, particularly in regards to Python regex testing. It highlights the significance of regular expressions (regex) in various automation scenarios and introduces the reader to the basics of Python regex components, including syntax, patterns, and functions.

Overall, the article serves as a helpful resource for those interested in improving their automation testing skills and learning more about the power of Python regex. By leveraging various NLP techniques, we can develop a robust system that can accurately identify questions in natural language text. Future enhancements can focus on improving accuracy, expanding the scope of question categories, and integrating additional NLP techniques.

In conclusion, the provided Python code utilizes regular expressions to identify common question patterns within a given sentence. The ‘is\_question’ function checks for the presence of words indicative of questions, such as "what," "when," "where," etc. The example usage demonstrates how to use the function to determine whether a given sentence is a question or not.

**Detect Questions using Python**

## **REFERENCE**

* <https://thecleverprogrammer.com/>
* <https://towardsdatascience.com/>
* <https://chat.openai.com/>
* <https://datascience.stackexchange.com/>
* <https://stackoverflow.com/>
* <https://github.com/>
* <https://www.geeksforgeeks.org/>
* <https://www.useblackbox.io/pricing>