**Automatic question tagging system**

**Purpose**

The automatic question tagging system is designed to categorize and label questions based on their content and topic.

### Functionality

The system utilizes natural language processing and machine learning algorithms to analyze the text of questions and assign relevant tags.

**Preprocessing:**

### Text Cleaning

* Remove any unnecessary characters, such as punctuation marks and special symbols.
* Convert all text to lowercase to ensure consistency in word representation.

### Tokenization

* Split the text into individual words or tokens.
* This allows for further analysis and processing on a word-level basis.

### Stop word Removal

* Remove common words, such as 'a', 'the', and 'is', that do not provide much meaning to the context of the question.
* This helps reduce noise and improve the efficiency of the tagging system.

### Lemmatization

* Convert words to their base or dictionary form.
* This helps in reducing the dimensionality of the data and improving the accuracy of the tagging system.

**Feature Extraction**

### Tokenization

The system uses tokenization to break down questions into individual words or tokens, which serve as the basic units for feature extraction.

### Stop Word Removal

Common words like 'the', 'is', and 'and' are removed from the tokenized questions as they do not carry significant meaning for question tagging.

### Lemmatization

The system applies lemmatization to reduce words to their base or root form, ensuring that different forms of the same word are treated as one feature.

### Part-of-Speech Tagging

Each token is assigned a part-of-speech tag, such as noun, verb, adjective, or adverb. This helps capture the grammatical structure and semantic meaning of the question.

### Evaluation and Validation

### Accuracy Assessment

* To evaluate the performance of the automatic question tagging system, a sample dataset of pre-tagged questions was used as a benchmark.
* The system's predicted tags were compared against the ground truth tags to calculate accuracy.

### Precision and Recall

* Precision and recall measures were also calculated to assess the system's performance.
* Precision measures the proportion of correctly tagged questions out of all predicted tags.
* Recall measures the proportion of correctly tagged questions out of all actual tags.

**Integration and Deployment**

### Seamless Integration

* Our automatic question tagging system can be easily integrated into existing platforms and systems.
* We provide comprehensive documentation and support to ensure a smooth integration process.

### Customization and Configuration

* Our system can be customized and configured to meet the specific needs and requirements of your organization.
* We work closely with your team to understand your tagging criteria and implement the necessary configurations.

### Deployment Options

* Our system can be deployed on-premises or on the cloud, depending on your preferences and infrastructure.
* We offer flexible deployment options to ensure compatibility with your existing technology stack.

### Training and Support

* We provide comprehensive training and support to ensure your team is equipped with the knowledge and skills to effectively use our system.
* Our support team is available 24/7 to address any issues or questions that may arise.

**CODE:**

**IMPLEMENTATION OF CODE**:

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.multioutput import MultiOutputClassifier

from sklearn.ensemble import RandomForestClassifier

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

from sklearn.pipeline import Pipeline

from sklearn.metrics import accuracy\_score, f1\_score

# Sample data (questions and corresponding tags)

data = {

'Question': ['What is the capital of France?', 'How do I cook pasta?', 'What is the population of Tokyo?'],

'Tag1': ['geography', 'cooking', 'geography'],

'Tag2': ['city', 'recipe', 'city']

}

df = pd.DataFrame(data)

# Splitting data into features (X) and labels (y)

X = df['Question']

y = df.drop('Question', axis=1)

# Splitting data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Define a TF-IDF vectorizer and a random forest classifier

tfidf\_vectorizer = TfidfVectorizer()

rf\_classifier = RandomForestClassifier()

# Create a multi-output classifier pipeline

pipeline = Pipeline([

('tfidf', tfidf\_vectorizer),

('clf', MultiOutputClassifier(rf\_classifier))

])

# Train the model

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

# Predict the tags for test data

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

# Flatten the multi-output labels into a single list

y\_test\_flattened = pd. DataFrame(y\_test.values.ravel())

y\_pred\_flattened = pd.DataFrame(y\_pred.ravel())

# Evaluate accuracy

accuracy = accuracy score(y\_test\_flattened, y\_pred\_flattened)

print("Accuracy:", accuracy)

# Calculate F1 score

f1 = f1\_score(y\_test\_flattened, y\_pred\_flattened, average="micro")

print("F1 score:", f1)

**OUTPUT:**

ACCURACY: 1.0

F1 SCORE : 1.0