Automatic Questions Tagging System

"Identifying tags from question text of Stack Overflow Q&A about the Python programming language"

16

Naufal Halim (19/446784/TK/49889)
Samatha Marhaendra P. (19/444071/TK/49267)
Anas Syahirul Alim (19/439809/TK/48539)

Background & Purpose

Stack Overflow merupakan platform dan komunitas *online* terbesar yang terpercaya di lingkup *developer* di dunia teknologi. Mereka di sana memiliki tujuan yang sama untuk berbagi pengetahuan tentang *programming*, saling membantu menyelesaikan masalah, dan membangun karir.

Manual question Tagging issues:

- *Membutuhkan waktu yang lebih banyak (Time consumming)*
- *Dapat menimbulkan tidak konsistennya tag(May lead to unconsistency)*

Karena terdapat berbagai topik yang dibahas, dilakukan tagging untuk mengelompokkan pertanyaan-pertanyaan yang ada ke dalam topik-topik tertentu. Dengan adanya tag pada tiap pertanyaan, hal ini akan memudahkan developer untuk mencari topik-topik tertentu.

Memprediksi tags yang erat kaitannya
dengan subjek pertanyaan

"Seberapa penting implementasi project ini untuk stackoverflow ?"

Benefit :

- pertanyaannya akan lebih cepat sampai dan tepat sasaran
- pertanyaan akan lebih mudah di identifikasi tags-nya
- Mempermudah memberikan notifikasi kepada pengguna yang akan menjawab dan menargetkan yang punya kompetensi dibidangnya
- Dari sisi penanya, cepat mendapatkan jawaban dan dijawab oleh pengguna yang kompeten
- Dari sisi penjawab, stack overflow akan merekomendasikan pertanyaan-pertanyaan yang relate dengan pertanyaan yang sering dijawab atau memang bidangnya

Dengan demikian, keuntungannya ialah stack overflow dapat membantu manambahkan tags secara otomatis, meminimalisasi tags yang keliru, mempercepat distribusi jawaban dan pertanyaan serta berdampak pada pengalaman pengguna disana

Overview Project

Dataset & Library

Dataset :
Python Question from Stack Overflow.
Full text of Stack Overflow Q&A about the Python programming language

Library :
NLTK (Natural Language Toolkit)

Pre-Processing

1. Merge The Datasets into a Single Dataset
2. Filter Dataset Based on 'Score' column and Most Frequently used Tags
3. Clean The Data
4. Lemmatization
5. TF-IDF Vectorization

Method Used

K-Nearest Neighbors

Datasets Overview

- Questions
- Answers
- Tags

```
[ ]  questions.head()
```

	Id	OwnerUserId	CreationDate	Score	Title	Body
0	469	147.0	2008-08-02T15:11:16Z	21	How can I find the full path to a font from it...	<p>I am using the Photoshop's javascript API t...
1	502	147.0	2008-08-02T17:01:58Z	27	Get a preview JPEG of a PDF on Windows?	<p>I have a cross-platform (Python) applicatio...
2	535	154.0	2008-08-02T18:43:54Z	40	Continuous Integration System for a Python Cod...	<p>I'm starting work on a hobby project with a...
3	594	116.0	2008-08-03T01:15:08Z	25	cx_Oracle: How do I iterate over a result set?	<p>There are several ways to iterate over a re...
4	683	199.0	2008-08-03T13:19:16Z	28	Using 'in' to match an attribute of Python obj...	<p>I don't remember whether I was dreaming or ...

```
[ ]  answers.head()
```

	Id	OwnerUserId	CreationDate	ParentId	Score	Body
0	497	50.0	2008-08-02T16:56:53Z	469	4	<p>open up a terminal (Applications->Utilit...
1	518	153.0	2008-08-02T17:42:28Z	469	2	<p>I haven't been able to find anything that d...
2	536	161.0	2008-08-02T18:49:07Z	502	9	<p>You can use ImageMagick's convert utility f...
3	538	156.0	2008-08-02T18:56:56Z	535	23	<p>One possibility is Hudson. It's written in...
4	541	157.0	2008-08-02T19:06:40Z	535	20	<p>We run <a href="http://buildbot.net/trac">B...

```
tags.head()
```

	Id	Tag
0	469	python
1	469	osx
2	469	fonts
3	469	photoshop
4	502	python

Merging

- The Datasets

```
# Joining answers grouped by 'Id'
grouped_answers = answers.groupby('Id')['Answer'].apply(lambda answer: ' '.join(answer))
grouped_answers = grouped_answers.to_frame().reset_index()

# Joining tags grouped by 'Id'
grouped_tags = tags.groupby('Id')['Tag'].apply(lambda tag: ' '.join(tag))
grouped_tags = grouped_tags.to_frame().reset_index()

# Merging 'Questions' dataframe with 'Answers' dataframe, then with 'Tags' dataframe
df = questions.merge(grouped_answers, how='left', on='Id')
df = df.merge(grouped_tags, how='left', on='Id')

df.drop(columns=['Id', 'OwnerUserId', 'CreationDate'], inplace=True)
```

Data Cleaning

- Punctuation Removal
- HTML Tags
- Lemmatization
- Stopwords Removal

```
# Dropping 'answer' column since it can't be imputed (because this column values is neither categorical nor continuous in nature)
df.drop(columns=['answer'], inplace=True)

# Defining a function to remove punctuation
def punctuation_remover(text):
    for punctuation in string.punctuation:
        text = text.replace(punctuation, '')
    return text

# Changing the data type of 'title' and 'question' columns to string
df['title'] = df['title'].astype(str)
df['question'] = df['question'].astype(str)

# Removing HTML tags on 'question' column values
df['question'] = df['question'].apply(lambda question: re.sub('<[^<]+?>', '', question))

# Applying 'punctuation_remover' function on 'title' and 'question' columns
df['title'] = df['title'].apply(punctuation_remover)
df['question'] = df['question'].apply(punctuation_remover)

# Changing text into lowercase
df['title'] = df['title'].str.lower()
df['question'] = df['question'].str.lower()

# Splitting the text into words
df['title'] = df['title'].str.split()
df['question'] = df['question'].str.split()

lematizer = WordNetLemmatizer()

# Defining lemmatizer function
def word_lemmatizer(text):
    lemma_text = [lematizer.lemmatize(word) for word in text]
    return lemma_text

# Applying lemmatizer function to 'title' and 'answer' columns
df['title'] = df['title'].apply(lambda title: word_lemmatizer(title))
df['question'] = df['question'].apply(lambda question: word_lemmatizer(question))

# Removing stopwords
df['title'] = df['title'].apply(lambda title: [word for word in title if word not in stopwords.words('english')])
df['question'] = df['question'].apply(lambda question: [word for word in question if word not in stopwords.words('english')])
```

TF-IDF

- Vectorization

```
'''
TF-IDF equation:
TF = (# of times of specific word in a doc) / (# of words in doc)
IDF = log((# of docs) / (# of docs that contains specific word))
TF-IDF = TF * IDF
'''

vectorizer = TfidfVectorizer()

# Changing the data type of 'title' and 'question' columns to string
df['title'] = df['title'].astype(str)
df['question'] = df['question'].astype(str)

X1 = vectorizer.fit_transform(df['title'].str.lower())
X2 = vectorizer.fit_transform(df['question'].str.lower())
```

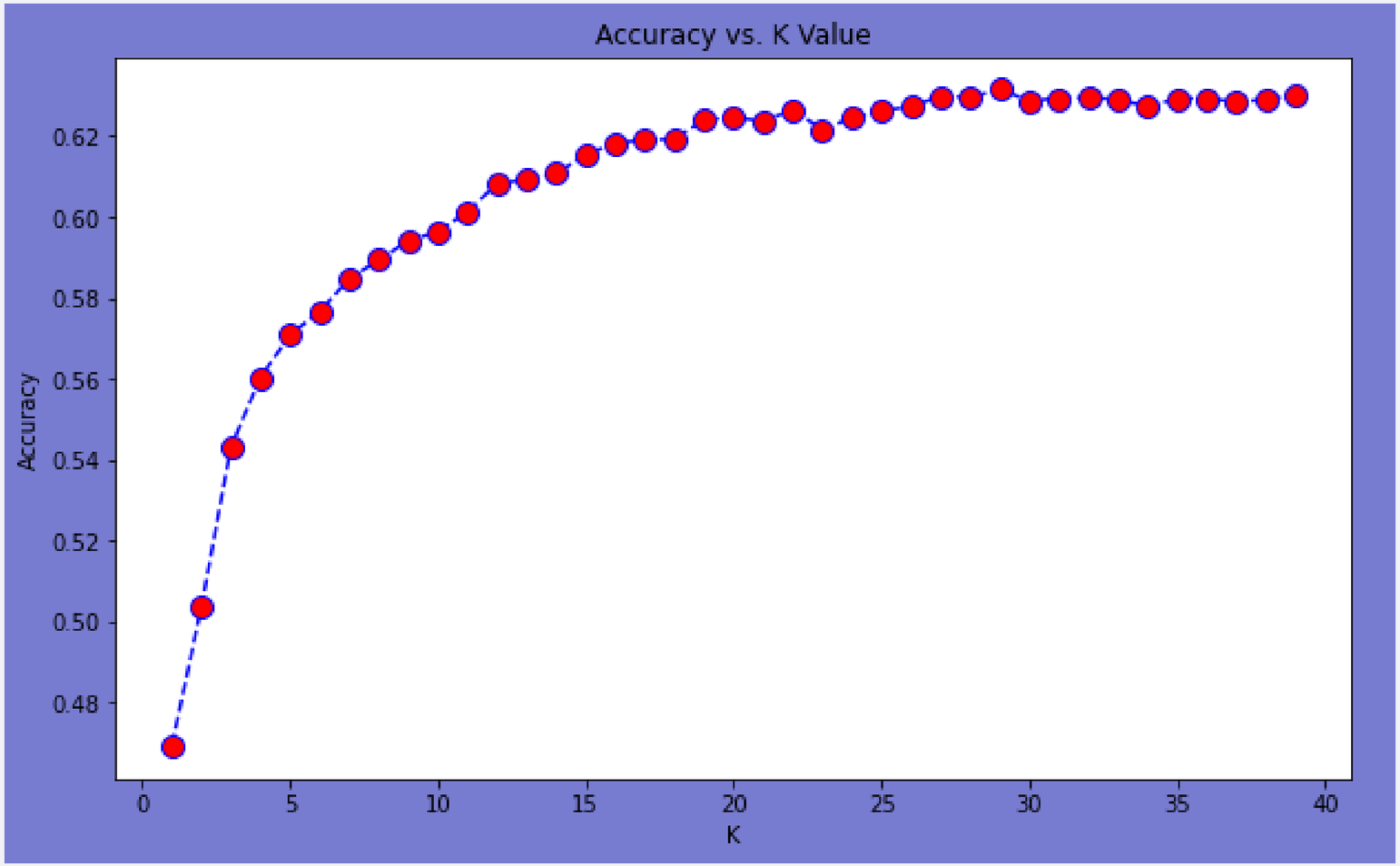
```
accuracy = []

for i in range(1, 40):
    KNN = KNeighborsClassifier(n_neighbors = i).fit(x_train, y_train)
    prediction = KNN.predict(x_test)
    accuracy.append(metrics.accuracy_score(y_test, prediction))

plt.figure(figsize=(10, 6))
plt.plot(range(1, 40), accuracy, color = 'blue', linestyle='dashed',
         marker='o', markerfacecolor='red', markersize=10)
plt.title('Accuracy vs. K Value')
plt.xlabel('K')
plt.ylabel('Accuracy')
print("Maximum Accuracy:", max(accuracy), "at K =", accuracy.index(max(accuracy))+1)
```

Data Modelling

- Using K-Nearest Neighbors algorithm




```
Maximum Accuracy : 0.6314333148865523 at K = 29
```

TF-IDF

doc1: saya suka makan
doc2: dia suka minum
doc3: teman saya yang suka dengan dia suka naik sepeda
doc4: abdi sedang jalan-jalan

TF = Jumlah kata spesifik dalam satu dokumen / Jumlah kata dalam satu dokumen
IDF = LOG(Jumlah dokumen / Jumlah dokumen yang mengandung kata spesifik)
TF-IDF = TF x IDF

Perhitungan pada doc1
-kata spesifik: "suka"
TF = 1 / 4 = 0.25
IDF = LOG(4 / 3) = 0.1249
TF-IDF = 0.25 x 0.1249 = 0,031225

Jadi, nilai TF-IDF untuk kata "suka" pada doc1 adalah 0,031225

k-nearest neighbors algorithm

Misalkan kita memiliki matriks TF-IDF dan variabel target sebagai berikut:

	saya	makan	bakso	variabel_target
doc1	0.005	0.123	0.03	jajan
doc2	0	0.05	0.2	masak
doc3	0.112	0.003	0	jajan
doc4	0.012	0	0	???

Eucledian Distance = sqrt((q1-p1)^2 + (q2-p2)^2 + ... + (qn-pn)^2))

	saya	makan	bakso	jarak_terhadap_doc4
doc1	0.005	0.123	0.03	0,1267
doc2	0	0.05	0.2	0,2065
doc3	0.112	0.003	0	0,1000

misal: k=2

Artinya, dipilih doc3 dan doc1 sebagai 2 tetangga terdekat dengan doc4.
doc3 dilabeli "jajan" dan doc1 dilabeli "jajan".

"jajan" = 2
"makan" = 0

maka, dapat disimpulkan bahwa doc4 memiliki label "jajan"