search_engine

August 29, 2018

1 Simple Search Engine

Menggunakan **TF-IDF** sebagai pembobotan feature vector text dan **cosine similarity** untuk menghitung tingkat kedekatan antar string.

1.1 Import library yang dibutuhkan

In [1]:
```
import random
import re
import numpy as np
import pandas as pd
import nltk
from nltk.tokenize import RegexpTokenizer, word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords as nltk_stopwords
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk import FreqDist

# download data-data yang dibutuhkan nltk
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/satriajiwidi/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     /home/satriajiwidi/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /home/satriajiwidi/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[1]:
```
True
```

1.2 Baca dataset

Dataset yang digunakan adalah dataset IMDb Movie Review (yang sebenarnya adalah dataset untuk melakukan sentiment analysis, namun untuk hal ini label sentiment diabaikan)

```
In [2]: data = pd.read_csv('imdb_master.csv', encoding='latin-1')
```

```
In [3]: # yang akan dilakukan pre-processing
        texts = data.review.tolist()
        # dipilih secara random 1000 data saja (untuk menyederhakan komputasi)
        random.seed(123)
        texts = random.sample(texts, 1000)

        # data original
        data = texts
```

1.3 Beberapa fungsi yang dibutuhkan untuk melakukan pre-processing text

- *rm_stopwords*: fungsi untuk menghapus stopwords dari text
- *rm_punc*: fungsi untuk menghapus tanda baca dari text
- *lemmatize*: fungsi untuk mengubah kata sesuai kamus Inggris
- *stem*: fungsi untuk mengubah kata menjadi kata dasar
- *preprocess*: fungsi untuk melakukan 4 fungsi sebelumnya sekaligus
- *vectorize*: fungsi untuk mengubah text menjadi feature vector

```
In [4]: def rm_stopwords(texts, stopwords):
            result = []
            for text in texts:
                text = [word for word in text.lower().split()
                        if word not in stopwords]
                result.append(' '.join(text))

            return result


        def rm_punc(texts):
            tokenizer = RegexpTokenizer(r'\w+')

            return [' '.join(tokenizer.tokenize(text)) for text in texts]


        def lemmatize(texts):
            lemmatizer = WordNetLemmatizer()
            result = []

            for text in texts:
                words = word_tokenize(text)
                words = [lemmatizer.lemmatize(word) for word in words]
                result.append(' '.join(words))
```

```
        return result


    def stem(texts):
        stemmer = PorterStemmer()
        result = []

        for text in texts:
            words = word_tokenize(text)
            words = [stemmer.stem(word) for word in words]
            result.append(' '.join(words))

        return result


    def preprocess(texts, stopwords):
        return stem(lemmatize(rm_punc(rm_stopwords(texts, stopwords))))


    def vectorize(texts, vocabulary):
        vectorizer = TfidfVectorizer(vocabulary=vocabulary)

        return vectorizer.fit_transform(texts)
```

1.4 Stopwords

```
In [5]: # load list stopwords bahasa Inggris
        stopwords = nltk_stopwords.words('english')
```

```
In [6]: print(stopwords[:10])
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]
```

1.5 Pre-process text

```
In [7]: texts = preprocess(texts, stopwords)
        print(texts[0][:80] + '...')
```

```
saw film fantasi filmfest berlin impress br br a far stori go girlfriend return ...
```

1.6 Buat vocabulary

```
In [8]: def is_number(word):  # cek apakah numerik
            return bool(re.search(r'\d', word))

        # berikut dilakukan beberapa proses eliminasi untuk memperkecil ukuran vektor
```

```
        # eliminasi kata dengan panjang <= 2 dan kata yang mengandung numerik
        all_words = [word for text in texts for word in word_tokenize(text)
                     if len(word) > 2 and not is_number(word)]

        # eliminasi kata jika frekuensi kemunculannya tidak lebih dari satu kali
        fd = dict(FreqDist(all_words))
        all_words = [word for word in all_words if fd[word] > 1]
        all_words = sorted(list(set(all_words)))

        print(all_words[:5])
```

```
['aamir', 'aaron', 'abandon', 'abc', 'abduct']
```

```
In [9]: # total panjang kamus yang terbentuk
        print('panjang kamus:', len(all_words))
```

```
panjang kamus: 7152
```

```
In [10]: texts = vectorize(texts, all_words)
```

1.7 Definisikan string query

```
In [11]: # string query yang akan digunakan sebagai query pencarian
         query = 'worst horror movie ever'
         query = vectorize(preprocess([query], stopwords), all_words)
```

1.8 Hitung kemiripan antara query dengan seluruh text dan ambil *n* text terdekat

```
In [12]: # perhitungan cosine similarity
         sim = cosine_similarity(query, texts)
         sim = sim.reshape(sim.shape[1], )
```

```
In [13]: n = 10  # hanya 10 text paling mirip yang akan ditampilkan
         indices = np.argpartition(sim, -n)[-n:]

         def sort_by_val(i):  # fungsi custom sort parameter
             return sim[i]

         # indeks-indeks 10 text paling relevan (atau dekat) dengan query
         # urut secara descending
         indices = sorted(indices, key=sort_by_val, reverse=True)
         indices
```

```
Out[13]: [417, 606, 983, 523, 990, 378, 574, 399, 432, 927]
```

1.9 Tampilkan hasil

```
In [14]: for index, i in enumerate(indices):
             print(str(index+1) + '.')
             print('cosine similarity:', round(sim[i], 5))
             print(data[i][:80] + '...')
             print()
```

```
1.
cosine similarity: 0.3732
Positively one of the worst horror movies ever. Bad script, acting, music... you...

2.
cosine similarity: 0.24779
This movie was by far the worst movie that I have ever seen in my entire life. I...

3.
cosine similarity: 0.23323
This is possibly the worst movie I have ever seen. Can somebody please explain t...

4.
cosine similarity: 0.23184
First of all, I have to say I have worked for blockbuster and have seen quite a ...

5.
cosine similarity: 0.22723
Absolutely one of the worst movies I've ever seen! "The Beginning" was not the g...

6.
cosine similarity: 0.2259
I don't think there is any kind of constructive criticism I could offer to a mov...

7.
cosine similarity: 0.22481
This movie was so cool! I saw it on a Friday night with a couple of my friends. ...

8.
cosine similarity: 0.21403
It's amazing that this movie turns out to be in one of my hitlists after all. It...

9.
cosine similarity: 0.2113
Charles Bronson continued his 80's slump with what could possibly be his worst m...

10.
cosine similarity: 0.20526
Oh where to begin! This movie was so ridiculous I'm almost ashamed to admit that...
```