Laporan Tugas Akhir Mata Kuliah Perolehan Informasi

Rekomendasi Buku Berdasarkan *Cosine Similarity* Dengan Pembobotan TF-IDF

KELOMPOK: SNOOPY

Avicenna Wisesa	**1506731561**
Bthari Smartnastiti	**1506728775**
Glory Finesse Valery	**1506688815**

Fakultas Ilmu Komputer
Universitas Indonesia

2017

1. Pendahuluan

1.1. Latar Belakang

Pada era modern seperti sekarang gaya hidup menjadi suatu yang dinilai penting untuk seseorang, soal bagaimana seseorang dapat dinilai maupun rasa puas terhadap diri sendiri. Gaya hidup sendiri merepresentasikan bagaimana cara seseorang untuk menjalani hidup yang ditunjukkan dari aktivitas, sikap, ketertarikan, opini, dan nilai-nilai. Maka, gaya hidup adalah sarana untuk membangun kepercayaan diri dan menciptakan aura yang selaras dengan identitas diri. Gaya hidup tentunya didukung oleh banyak hal seperti cara makan, oleh raga, dan *entertainment*. Terlepas dari berbagai kemajuan teknologi yang muncul dewasa ini, membaca tetap menjadi aktifitas yang dipandang positif oleh masyarakat karena menambah wawasan dan perilaku yang akan tertanam pada diri seseorang sampai akhir hayat.

Membaca yang berkaitan dengan buku menjadi komponen yang penting untuk menambahkan *vocabulary* kita, mendapatkan ilmu dan keterampilan, dan mendapatkan informasi secara cepat dari media. Membaca juga membuka pikiran kita tentang berbagai budaya di dunia, dan seiring dengan informasi yang kita dapat, ini membantu kita untuk dapat berpikir lebih luas dan memiliki berbagai ragam emosianal juga. Dan dengan adanya teknologi yang telah membawa banyak perubahan pada cara kita membaca, teknologi memudahkan kita untuk mencari informasi yang kita butuhkan.

Orang-orang membutuhkan beberapa referensi serupa yang bisa didapatkan dari sebuah buku atau keperluan untuk mencari buku yang masih dalam satu kategori untuk memperluas cabang ilmu yang sedang dibaca, namun kebutuhan ini kurang didukung oleh platform yang dapat memberikan rekomendasi buku-buku serupa atau saling berhubungan. Memang sudah terdapat beberapa platform untuk menunjukkan buku yang memiliki isi berkorelasi atau rekomendasi buku dari orang-orang yang memiliki ketertarikan terhadap sebuah buku tertentu, tapi platform tersebut belum memberikan rekomendasi dengan beberapa buku yang bisa kita cari sendiri.

1.2. Tujuan

Didukung oleh keperluan akan referensi cepat dan pencarian buku serupa yang dapat digunakan untuk *research* maupun pendukung gaya hidup dan era teknologi yang membuat tuntutan banyak hal dapat dikerjakan dalam waktu cepat dan memiliki cara penggunaan yang mudah untuk dipahami. **Snoopy Recommender**

bertujuan untuk memberikan *platform* yang dapat digunakan dengan mudah dan tidak memakan waktu yang lama untuk mencari keperluan buku-buku yang sedang dibutuhkan.

1.3. Manfaat

Dengan fitur yang tersedia pada *platform* ini diharapkan dapat membantu pencarian referensi buku-buku yang dibutuhkan untuk kepentingan *research* maupun mendukung gaya hidup yang disukai oleh *user*, juga dapat membantu *user* memperoleh referensi tersebut dengan lebih efektif dan efisien serta tidak memakan waktu yang lama untuk melakukan pencarian.

2. Studi Literatur

TF-IDF
(https://nlp.stanford.edu/IR-book/html/htmledition/tf-idf-weighting-1.html)
Menggabungkan definisi frekuensi frekuensi dan frekuensi dokumen terbalik, untuk menghasilkan bobot komposit untuk setiap istilah dalam setiap dokumen. Skema pembobotan tf-idf memberikan istilah bobot pada dokumen **d** yang diberikan oleh

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t.$$

Dengan kata lain, tf-idft, **d** menetapkan untuk memberi bobot pada dokumen **d** yaitu

1. Paling tinggi ketika **t** terjadi berkali-kali dalam sejumlah kecil dokumen (dengan demikian memberikan pinjaman diskriminatif tinggi terhadap dokumen-dokumen tersebut)
2. Turun bila istilah itu terjadi lebih sedikit dalam sebuah dokumen, atau terjadi dalam banyak dokumen (sehingga menawarkan sinyal relevansi yang kurang jelas)
3. Terendah bila istilah itu terjadi di hampir semua dokumen

Pada titik ini, kita dapat melihat setiap dokumen sebagai vektor dengan satu komponen yang sesuai dengan setiap istilah dalam kamus, bersama dengan bobot untuk setiap komponen yang diberikan oleh. Untuk istilah kamus yang tidak terjadi dalam dokumen, bobot ini adalah nol. Bentuk vektor ini terbukti sangat penting untuk mencetak dan memberi peringkat. Sebagai langkah awal, ukuran skor tumpang tindih: skor dokumen **d** adalah jumlah, sepanjang semua persyaratan kueri, berapa kali masing-masing istilah kueri terjadi di **d**. Kita bisa memperbaiki gagasan ini sehingga kita tidak menambahkan jumlah kejadian dari setiap istilah kueri term **t** ke **d**, tapi bobot tf-idf masing-masing istilah dalam **d**.

3. Metodologi

3.1. Pengambilan & Pemrosesan Data Koleksi

- ***Data Acquisition***
  Melakukan pengambilan data buku dari *Google Books* API dan di-*store* ke dalam *file* yang bernama **books.json** (dalam format JSON)

```
import urllib.request, json

x = 0
f = open('books.json','w')

while True:
	with
urllib.request.urlopen("https://www.googleapis.com/books/v1/volumes?q=''&lan
gRestrict=en&startIndex="+str(x)+"&maxResults=40") as url:
		dct = json.loads(url.read().decode())
		dct_to_json = json.dumps(dct,ensure_ascii=False,indent=2)
		f.write(dct_to_json)
	x += 40

f.close()
```

- ***Data Preparation***
  Melakukan seleksi atribut dari data buku yang akan digunakan. Penulis hanya mengambil atribut **judul** dan **sinopsis** untuk setiap data buku, lalu di-*store* ke dalam *file* yang bernama **new-books.txt**

```
fr = open("books.json", "r")
fw = open("new-books.txt", "w")
ind = 1

while True:
	txt = fr.readline()
	if "\"volumeInfo\"" in txt:
		fw.write("\n\nID:"+str(ind)+"\n")
		ind += 1
	if "\"description\"" in txt:
```

```
            fw.write(txt)
      if "\"title\"" in txt:
            fw.write(txt)
```

- ***Data Importing***

  Melakukan *import* data ke dalam *database Django Apps* dengan membuat *model* **Book** terlebih dahulu lalu melakukan *insert* untuk setiap *object* data buku yang memiliki atribut **book_id**, **title** (judul), dan **description** (sinopsis)

```
from retrieve.models import Book
fr = open("new-books.txt", "r")
idt = 0
b = Book(book_id=0,title='-')
while idt < 576:
      txt = fr.readline()
      if "\"ID\"" in txt:
            b.save()
            print(str(idt)+" done.\n")
            idt = idt + 1
            b = Book(book_id=idt)
      if "\"title\"" in txt:
            tmp = txt.split(":")
            b.title = tmp[1]
      if "\"description\"" in txt:
            tmp = txt.split(":")
            b.description = tmp[1]
```

3.2. Pemilihan & Penggunaan Model *Retrieval*

Menggunakan Vector Space Model, dengan penghitungan kemiripan berdasarkan *cosine similarity* dengan pembobotan TF-IDF

```
def cosine_similarity(arr1, arr2):

      q1 = np.array(arr1)
      q2 = np.array(arr2)

      if(len(arr1) > len(arr2)):
            q2_tmp = np.zeros(q1.shape)
            q2_tmp[:q2.shape[0]] = q2
            q2 = q2_tmp
      elif(len(arr2) > len(arr1)):
            q1_tmp = np.zeros(q2.shape)
            q1_tmp[:q1.shape[0]] = q1
```

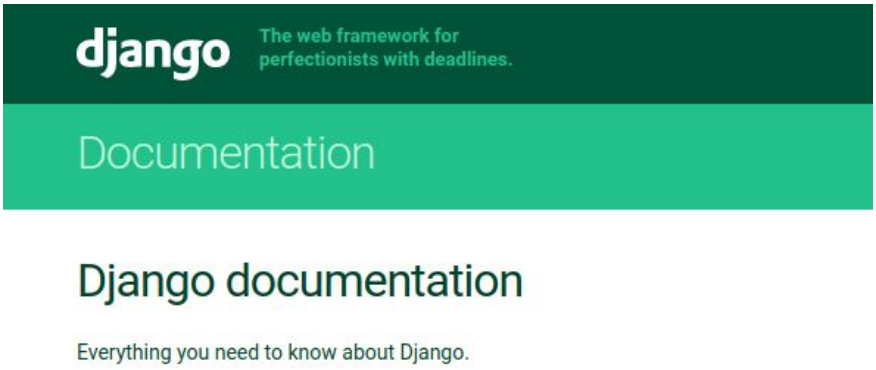
```
            q1 = q1_tmp

      dot = np.dot(q1,q2)
      mag = np.linalg.norm(q1)*np.linalg.norm(q2)

      return dot/mag

// ..
// ..

def prepare(self):

            stopwords_list = set([stemmer(x) for x in
stopwords.words("english")])

            title_wc = {}
            tokens = self.title.lower().split()
            tokens = [stemmer(word) for word in tokens]
            tokens = [word for word in tokens if word not in
stopwords_list]
            for word in tokens:
                  title_wc[word] = title_wc.get(word, 0) + 1

            desc_wc = {}
            tokens = self.description.lower().split()
            tokens = [stemmer(word) for word in tokens]
            tokens = [word for word in tokens if word not in
stopwords_list]
            for word in tokens:
                  desc_wc[word] = desc_wc.get(word, 0) + 1

            self.title_wc = json.dumps(title_wc)
            self.desc_wc = json.dumps(desc_wc)

            self.save(update_fields=['title_wc','desc_wc'])

// ..
// ..

def count_idf(collection_size, count):
      upper_hand = math.log( collection_size / count )
      lower_hand = math.log(10)

      idf = upper_hand / lower_hand

      return idf
```

4. Fitur

User dapat melakukan pencarian dengan memasukan *keyword* apapun, lalu sistem akan memberikan rekomendasi buku berdasarkan *keyword* yang diberikan oleh *user* dengan menggunakan model *retrieval* yang telah diimplementasikan.

5. Panduan Instalasi

Instalasi Django *Web Framework*: https://docs.djangoproject.com/



6. Pembagian Tugas

Nama	NPM	*Job Description*
Avicenna Wisesa	1506731561	Membuat *back-end*, membuat *retrieval* model, membuat laporan
Bthari Smartnastiti	1506728775	Membuat *front-end* dan sebagian integrasi, membuat poster, membuat laporan
Glory Finesse Valery	1506688815	Data *preprocessing*, membuat *database*, membuat laporan

References:

- Geetali. *Reading Habits and Its Importance in Society*. www.dlsbmscollege.org/faculty-publications/arunav-golden-jubilee-year-pages-88-91.pdf
- *Django Documentation*. https://docs.djangoproject.com/
- TF-IDF Theory. https://nlp.stanford.edu/IR-book/html/htmledition/tf-idf-weighting-1.html