222112058_Feza Raffa Arnanda_Penugasan Praktikum 9

October 24, 2023

Penugasan Praktikum Pertemuan 9 Information Retrieval

Nama : Feza Raffa Arnanda

NIM : 222112058

Kelas : 3SD2

0.1 Soal

Buat fungsi untuk menampilkan 3 list dokumen yang terurut pada folder ”berita” dengan query ”vaksin corona jakarta”, berdasarkan standar query likelihood model serta query likelihood model dengan Laplace Smoothing, Jelinek-Mercer Smoothing, dan Dirichlet Smoothing. Bandingkan dengan hasil perankingan BM25 pada modul 8 serta cosine similarity pada modul 5.

0.2 Prepocessing

[ ]:
```
import nltk
from nltk.tokenize import word_tokenize
def tokenisasi(text):
    tokens = word_tokenize(text)
    return tokens
```

[ ]:
```
def stemming(text):
    from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
    # create stemmer
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    # stemming process
    output = stemmer.stem(text)
    return output
```

[ ]:
```
def stemming_sentence(text):
    stemmed_tokens = [stemming(token) for token in tokenisasi(text)]
    output = ' '.join(stemmed_tokens)
    return output
```

[ ]:
```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
def stemming_sastrawi(tokens):
    # Membuat stemmer
```

```
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    return [stemmer.stem(token) for token in tokens]
```

[ ]:
```
def case_folding(text):
    text = text.lower()
    return text
```

[ ]:
```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('Indonesian'))

def eliminasi_stopword(token):
    return [kata for kata in token if kata not in stop_words]
```

Path Folder Berita

[ ]:
```
path = "C:/Users/FEZA/My Drive/00. Drive PC/1.STIS/5. Semester 5/Information␣
↪Retrieval [IR] P/berita"
```

Open file folder berita

[ ]:
```
import re
import os
inverted_index = {}
doc_dict = {}
i = 1
for filename in os.listdir(path):
    if (filename.endswith('.txt')):
        file_path = os.path.join(path, filename)
        # Ekstrak angka dari nama file menggunakan regular expressions
        match = re.search(r'\d+', filename)
        if match:
            doc_id = match.group() # Mengambil angka dari nama file sebagai␣
↪dokumen ID
            with open (file_path, mode='r', encoding='utf-8') as file:
                text = file.read()
                hasil_case_folding = case_folding(text)
                token = tokenisasi(hasil_case_folding)
                token_bersih = eliminasi_stopword(token)
                stemm_token = stemming_sastrawi(token_bersih)
                stemm_token_final = [item for item in stemm_token if item !=␣
↪''] # membersihkan term kosong pada hasil stemming sebelumnya
                # Menggabungkan hasil stemming menjadi sebuah teks/paragraf
                doc_dict[doc_id] = ' '.join(stemm_token_final)
                for term in set(stemm_token_final): # penggunaan set untuk␣
↪mengantisipasi duplikasi term pada sebuah dokumen
                    if term in inverted_index:
                        inverted_index[term].append(doc_id)
```

```
                    else:
                        inverted_index[term] = [doc_id]
```

0.3 Standard Query Likelihood Model

Exact Top K Document

[ ]:
```
from collections import OrderedDict

def exact_top_k(doc_dict, rank_score, k):
    relevance_scores = {}
    # perubahan disini
    rank_score = list(rank_score.values())
    i = 0
    for doc_id in doc_dict.keys():
        relevance_scores[doc_id] = rank_score[i]
        i = i + 1
    sorted_value = OrderedDict(sorted(relevance_scores.items(), key=lambda x:␣
 ↪x[1], reverse = True))
    top_k = {j: sorted_value[j] for j in list(sorted_value)[:k]}
    return top_k
```

Query

[ ]:
```
query = "vaksin corona jakarta"
tokenized_query = tokenisasi(query)
```

[ ]:
```
likelihood_scores = {}
vocab = set()
for doc_id in doc_dict.keys():
    likelihood_scores[doc_id] = 1
    tokens = tokenisasi(doc_dict[doc_id])
    vocab.update(tokens)
    for q in tokenized_query :
        likelihood_scores[doc_id]=likelihood_scores[doc_id]*tokens.count(q)/
 ↪len(tokens)
print(likelihood_scores)
```

```
{'1': 0.0, '2': 0.0, '3': 1.1851851851851852e-05, '4': 0.0, '5': 0.0}
```

[ ]:
```
top_k_standard = exact_top_k(doc_dict,likelihood_scores,5)

for i, (doc_id, score) in enumerate(top_k_standard.items()):
    print(f"Rank {i + 1}: berita{doc_id}.txt, Skor: {score}")
```

```
Rank 1: berita3.txt, Skor: 1.1851851851851852e-05
Rank 2: berita1.txt, Skor: 0.0
Rank 3: berita2.txt, Skor: 0.0
```

```
Rank 4: berita4.txt, Skor: 0.0
Rank 5: berita5.txt, Skor: 0.0
```

0.3.1 Smoothing

[ ]:
```
tokenized_corpus = [j for sub in [tokenisasi(doc_dict[doc_id]) for doc_id in␣
 ↪doc_dict] for j in sub]
vocab = set(tokenized_corpus)
print(vocab)
```

```
{'2', 'januari', 'dki', 'd-5816690', 'prof', 'catat', 'riset', 'balitbangkes',
'nasional', 'tahap', 'vaksin', 'amerika', 'corona-di-as-mendadak-naik-lagi-usai-
serangan-delta-sempat-mereda', 'health', 'asal', 'dosis', 'gelombang', 'senin',
'cs', 'cek', 'rencana', 'idi', 'batas', 'corona', '2021', 'alert-kasus-varian-
delta-covid-19-di-dki-meningkat', 'influenza', 'dadak', 'pasca', 'detik', '11',
'alpha', 'anthony', '34', 'tiga', 'pasien', 'satgas', 'd-5816582', 'd-5816534',
'1-2', 'protokol', '165', 'hijau', 'berita-detikhealth', 'wilayah-kamu-sudah-
bebas-covid-19-cek-34-kabkota-zona-hijau-terbaru', '327', '3', '-', 'sehat',
'total', 'perintah', 'rutin', 'gantung', 'd-5812940', 'p2pml', 'level',
'tarmizi', 'kepala', 'stabil', '86', 'ada', 'pakar', 'moderna', '13', 'varian',
'tingkat', 'pfizer', 'area', 'timur', 'as', 'terap', 'beri', '2022', 'https',
'picu', 'kait', 'alami', 'jelas', 'nasihat', 'ikut', 'kendali', 'masyarakat',
'menteri', 'desember', 'dr', 'tular', 'hitung', 'wilayah', 'alert', 'virus',
'bebas', 'ketua', 'musim', 'booster', 'jenis', 'tambah', 'puncak', '24', 'jawa',
'com', 'jakarta', 'indonesia', 'kota', 'd-5813949', 'panas', 'covid-19',
'turun', 'medis', 'giat', 'cegah', 'direktur', 'barat', 'suntik', 'djoerban',
'15', 'baru', 'siti', 'minggu', 'kaji', 'utara', 'singgung', 'strain', 'lantas',
'zubairi', '1', 'ampuh', 'putih', 'laut', 'lawan', 'kab', 'nadia', 'aku',
'gedung', 'november', 'ikat', 'sakit', 'turut', 'fauci', 'bijak', 'longgar',
'laku', 'vaksin-covid-19-bakal-rutin-setiap-tahun-tergantung-ini-penjelasannya',
'data', 'serang', 'efektivitas', '57', 'pasti', 'sulawesi', 'bukti', '90',
'sebut', 'persen', 'signifikan', 'dokter', 'beta', 'delta', 'langsung',
'mobilitas', 'reda', 'kemenkes', 'serikat', 'vaksinasi', 'zona', 'dasar',
'ppkm', 'ri', 'ri-mulai-suntikkan-booster-di-2022-masihkah-ampuh-lawan-varian-
delta-cs'}
```

Laplace Smoothing

[ ]:
```
alpha = 1
likelihood_scores = {}
for doc_id in doc_dict.keys():
    likelihood_scores[doc_id] = 1
    tokens = tokenisasi(doc_dict[doc_id])
    for q in tokenized_query:
        likelihood_scores[doc_id]=likelihood_scores[doc_id]*(tokens.
 ↪count(q)+alpha)/(len(tokens)+len(vocab)*alpha)
print(likelihood_scores)
```

```
{'1': 1.502397864551771e-07, '2': 8.049605695096029e-07, '3':
```

```
1.6934217901613323e-06, '4': 4.5087423160886264e-07, '5': 3.465250811388477e-07}
```

[ ]:
```
top_k_laplace = exact_top_k(doc_dict,likelihood_scores,5)

for i, (doc_id, score) in enumerate(top_k_laplace.items()):
    print(f"Rank {i + 1}: berita{doc_id}.txt, Skor: {score}")
```

```
Rank 1: berita3.txt, Skor: 1.6934217901613323e-06
Rank 2: berita2.txt, Skor: 8.049605695096029e-07
Rank 3: berita4.txt, Skor: 4.5087423160886264e-07
Rank 4: berita5.txt, Skor: 3.465250811388477e-07
Rank 5: berita1.txt, Skor: 1.502397864551771e-07
```

Jelinek-Mercer Smoothing

[ ]:
```
lamda = 0.5
likelihood_scores = {}
for doc_id in doc_dict.keys():
    likelihood_scores[doc_id] = 1
    tokens = tokenisasi(doc_dict[doc_id])
    for q in tokenized_query:
        likelihood_scores[doc_id]=likelihood_scores[doc_id]*((lamda*tokens.
 ↪count(q)/len(tokens))+((1-lamda)*tokenized_corpus.count(q)/
 ↪len(tokenized_corpus)))
print(likelihood_scores)
```

```
{'1': 8.487885870243773e-07, '2': 3.3613924700770085e-06, '3':
9.024832978282128e-06, '4': 2.0535207750589776e-06, '5': 3.266991684078316e-06}
```

[ ]:
```
top_k_jm = exact_top_k(doc_dict,likelihood_scores,5)

for i, (doc_id, score) in enumerate(top_k_jm.items()):
    print(f"Rank {i + 1}: berita{doc_id}.txt, Skor: {score}")
```

```
Rank 1: berita3.txt, Skor: 9.024832978282128e-06
Rank 2: berita2.txt, Skor: 3.3613924700770085e-06
Rank 3: berita5.txt, Skor: 3.266991684078316e-06
Rank 4: berita4.txt, Skor: 2.0535207750589776e-06
Rank 5: berita1.txt, Skor: 8.487885870243773e-07
```

Dirichlet Smoothing

[ ]:
```
miu = 2
likelihood_scores = {}
for doc_id in doc_dict.keys():
    likelihood_scores[doc_id] = 1
    tokens = tokenisasi(doc_dict[doc_id])
    for q in tokenized_query:
```

```
        likelihood_scores[doc_id]=likelihood_scores[doc_id]*(tokens.
  ↪count(q)+miu*tokenized_corpus.count(q)/len(tokenized_corpus))/
  ↪(len(tokens)+miu)
print(likelihood_scores)
```

```
{'1': 1.9014081250546086e-09, '2': 3.3355748653437976e-07, '3':
1.1791929441312962e-05, '4': 1.4010654679151002e-08, '5': 2.438809804746205e-07}
```

[ ]:
```
top_k_dirichlet = exact_top_k(doc_dict,likelihood_scores,5)

for i, (doc_id, score) in enumerate(top_k_dirichlet.items()):
    print(f"Rank {i + 1}: berita{doc_id}.txt, Skor: {score}")
```

```
Rank 1: berita3.txt, Skor: 1.1791929441312962e-05
Rank 2: berita2.txt, Skor: 3.3355748653437976e-07
Rank 3: berita5.txt, Skor: 2.438809804746205e-07
Rank 4: berita4.txt, Skor: 1.4010654679151002e-08
Rank 5: berita1.txt, Skor: 1.9014081250546086e-09
```

0.4 Kesimpulan

Pada penggunaan standard query likelihood, top 5 document adalah berita3, berita1, berita2, berita4, dan berita5. Standard query likelihood model akan mengalami probabilitas 0 ketika tidak ada query yang muncul dalam dokumen dan akan langsung menghasilkan skor 0 dan menandakan bahwa metode ini sangat sensitif dan tidak memberikan relevansi top K document secara benar.

Ketika menggunakan Laplace smoothing, top 5 document adalah berita3, berita2, berita4, berita5, dan berita1.

Sedangkan ketika menggunakan Jelinek-Mercer dan Dirichlet smoothing, top 5 documentnya adalah berita3, berita2, berita5, berita4, dan berita1.

Perbedaan pada urutan ke 3 dan 4 pada Laplace dan Jelinek-Mercer,Dirichlet dengan perbedaan skor yang tidak terlalu jauh.

Ketika kita bandingkan dengan BM25 dan Cosine Similarity pada top 3 document, BM25, Jelinek-Mercer, serta Dirichlet menghasilkan top 3 yang sama, yaitu berita3, berita2, dan berita5. Sedangkan standard likelihood dan Laplace menghasilkan yang berbeda. Penggunaan cosine similarity juga menghasilkan top 3 document yang berbeda, dimana top 3 document menggunakan cosine similarity adalah berita2, berita3, dan berita5. Masing-masing metode mempunyai asumsi dan penghitungan yang berbeda, pada penugasan ini kita membandingkan hasil top K document yang dihasilkan.