# MODUL 9: LANGUAGE MODELS FOR INFORMATION RETRIEVAL

## 9.1 Deskripsi Singkat

Language model adalah model machine learning yang dapat memprediksi kata selanjutnya berdasarkan kata-kata yang telah dilihat. Salah satu language model yang akan digunakan yaitu n-gram language model. N-gram adalah suatu sequence dari n token atau kata. Jika n adalah 1 disebut dengan unigram, n adalah 2 disebut bigram, dan n adalah 3 disebut trigram. Cara kerja dari language model ini adalah dengan memprediksi probabilitas kata tertentu dalam suatu urutan kata.

Probabilitas tersebut dihitung dengan chain rule:

```
p(w1...ws) = p(w1) . p(w2 | w1) . p(w3 | w1 w2) ..... p(wn | w1...wn-1)
```

dengan asumsi Markov:

```
p(wk | w1...wk-1) = p(wk | wk-1)
```

Artinya, dilakukan aproksimasi *hystory* (konteks) dari kata wk dengan melihat hanya kata terakhir dari konteks tersebut.

Language model dapat diimplementasikan untuk query likelihood model dalam information retrieval dengan cara:

1. Buat language model untuk setiap dokumen
2. Estimasi probabilitas generating query berdasarkan model dokumen
3. Rangking dokumen menggunakan skor probabilitas tersebut

Beberapa teknik smoothing yang digunakan dalam menghitung query likelihood model diantaranya:

1. Laplace Smoothing
2. Jelinek-Mercer Smoothing
3. Dirichlet Smoothing

## 9.2 Tujuan Praktikum

1. Dapat memanfaatkan language model untuk information retrieval.

## 9.3 Material Praktikum

Tidak ada

## 9.4 Kegiatan Praktikum

Lengkapi terlebih dahulu kode berikut dengan tahapan preprocessing agar dapat menghitung Query Likelihood Model dari 10 dokumen yang sebelumnya digunakan pada modul 5. Query yang digunakan dalam proses pencarian yaitu ”sistem informasi statistik”.

```
doc_dict_raw = {}
doc_dict_raw['doc1'] = "pengembangan sistem informasi penjadwalan"
doc_dict_raw['doc2'] = "pengembangan model analisis sentimen berita"
doc_dict_raw['doc3'] = "analisis sistem input output"
doc_dict_raw['doc4'] = "pengembangan sistem informasi akademik
universitas"
doc_dict_raw['doc5'] = "pengembangan sistem cari berita ekonomi"
doc_dict_raw['doc6'] = "analisis sistem neraca nasional"
doc_dict_raw['doc7'] = "pengembangan sistem informasi layanan
statistik"
doc_dict_raw['doc8'] = "pengembangan sistem pencarian skripsi di
universitas"
doc_dict_raw['doc9'] = "analisis sentimen publik terhadap pemerintah"
doc_dict_raw['doc10'] = "pengembangan model klasifikasi sentimen
berita"

doc_dict = {}
for doc_id,doc in doc_dict_raw.items():
    doc_dict[doc_id] = stemming_sentence(doc)

query = "sistem informasi statistik"
tokenized_query = tokenisasi(query)
```

Kemudian tulis fungsi berikut untuk mendapatkan standar query likelihood model.

```
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)/le
n(tokens)
print(likelihood_scores)
```

Perhatikan skor yang didapatkan dari query likelihood model untuk setiap dokumen. Kemudian, lengkapi kode di atas sehingga dapat melakukan perankingan dari nilai peluang terbesar ke terkecil dan mengembalikan hanya top k dokumen dengan nilai peluang terbesar. Misalkan k = 5, dokumen apa yang dikembalikan sebagai hasil pencarian dengan kueri ”sistem informasi statistik”?

Selanjutnya, untuk menghindari peluang bernilai 0 pada unseen word, Anda akan menerapkan teknik smoothing pada query likelihood model. Sebelumnya, buat kode untuk mendapatkan vocabulary dari koleksi dokumen terlebih dahulu. Pada kode di bawah ini, digunakan struktur data `set` untuk menyimpan term unik dari term hasil tokenisasi seluruh koleksi dokumen. Tentunya Anda dapat menggunakan cara lainnya untuk mendapatkan vocabulary tersebut.

```
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)
```

Kemudian, tulis kode berikut untuk menerapkan Laplace Smoothing pada query likelihood model. Gunakan alpha = 1.

```
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)+a
lpha)/(len(tokens)+len(vocab)*alpha)
print(likelihood_scores)
```

Bandingkan skor hasil perankingan untuk top 5 dokumen yang dikembalikan dengan Laplace Smoothing untuk query ”sistem informasi statistik” tersebut dengan hasil top 5 yang didapatkan tanpa smoothing.

Selanjutnya, Anda akan menulis kode untuk menerapkan teknik Jelinek-Mercer Smoothing pada query likelihood model. Gunakan lamda = 0.5.

```
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.cou
nt(q)/len(tokens))+((1-
lamda)*tokenized_corpus.count(q)/len(tokenized_corpus)))
print(likelihood_scores)
```

Bandingkan skor hasil perankingan untuk top 5 dokumen yang dikembalikan dengan Jelinek-Mercer Smoothing untuk query ”sistem informasi statistik” tersebut dengan hasil top 5 yang didapatkan tanpa smoothing dan Laplace Smoothing.

Selanjutnya, Anda akan menulis kode untuk menerapkan teknik Dirichlet Smoothing pada query likelihood model. Gunakan miu = 2.

```
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)+m
iu*tokenized_corpus.count(q)/len(tokenized_corpus))/(len(tokens)+miu)
print(likelihood_scores)
```

Bandingkan skor hasil perankingan untuk top 5 dokumen yang dikembalikan dengan Dirichlet Smoothing untuk query ”sistem informasi statistik” tersebut dengan hasil top 5 yang didapatkan tanpa smoothing, Laplace Smoothing, dan Jelinek-Mercer Smoothing.

## 9.5 Penugasan

1. 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.