

**Tugas Individu 2 (TI 2)**  
**Eksperimen Pengembangan N-gram *Language Model***  
**Deadline Jumat, 29 September 2023 jam 22:00**

Pada Tugas Individu 2 ini, peserta diminta untuk melakukan eksperimen pengembangan N-gram *Language Model* serta mengukur kualitas model yang dibangun menggunakan metrik *perplexity*.

1. APA YANG PERLU DIPAHAMI?

Untuk menyelesaikan tugas ini, peserta diharapkan telah memahami konsep dari n-gram *language model* serta metrik evaluasi *perplexity*.

**1.1. Apa itu N-gram Language Model.** *Language Model* (LM) merupakan model yang memberikan nilai probabilitas pada urutan kata. Salah satu model bahasa yang paling sederhana adalah **n-gram**. Suatu *n-gram* adalah sebuah urutan yang terdiri dari  $n$  kata. Misal, suatu model bahasa 2-gram atau dikenal dengan bigram merupakan suatu urutan dua kata dari kumpulan kata.

Esensi dari *N-gram Language Model* adalah menghitung probabilitas kemunculan suatu urutan kata dalam sebuah kalimat. Untuk menghitung probabilitas tersebut memanfaatkan aturan rantai (*The chain rule*) yang kemudian disederhanakan menggunakan *Markov Assumption*. Dengan mengasumsikan kehadiran antar kata di dalam suatu kalimat sebagai komponen independen maka muncul beragam jenis *n-gram model*, seperti 1-gram (*unigram*), 2-gram (*bigram*), 3-gram (*trigram*), etc.

**1.2. Pre-Processing.** *Pre-processing* merupakan suatu tahap di mana data dilakukan penyesuaian sedemikian sehingga mampu diproses oleh mesin dan diharapkan memberikan hasil yang optimum.

Pada pembangunan *N-gram language model* ini terdapat dua tahap *pre-processing*, yakni *lowercasing* serta *tokenization*. **Lowercasing** merupakan tahap di mana masing-masing kata/token pada data dibuat seragam sebagai *lowercase*. Selanjutnya, **tokenization** merupakan tahap pemisahan kalimat berdasarkan kata/token penyusunnya.

Di dalam dunia *natural language processing* khususnya, terdapat dua jenis *pre-processing*, yakni **cased** dan **uncased**. *Uncased* artinya masing-masing token di dalam dataset telah dibuat seragam, antara sebagai *lowercase* atau *uppercase*. Sebaliknya, *cased* artinya masing-masing token tidak diterapkan proses penyeragaman sehingga masing-masing token di dalam data dibiarkan apa adanya.

Untuk melakukan tahap *lowercasing* atau *uppercasing*, Anda dapat menggunakan library `nlTK`<sup>1</sup>. Perlu diperhatikan bahwa teks yang Anda proses pada tugas kali ini adalah teks bahasa Indonesia. Dengan demikian, untuk melakukan tahap *tokenization* Anda dapat menggunakan library `aksara`<sup>2</sup>.

**1.3. Penanganan OOV (Out Of Vocabulary).** *Vocabulary* merupakan suatu koleksi kata unik yang terdapat pada dataset atau korpus.

*Out Of Vocabulary* merupakan suatu kondisi di mana terdapat kata di dalam *test set* yang tidak tersedia di dalam *vocabulary* yang dimiliki. Masing-masing kata di dalam model diberikan suatu nilai probabilitas berdasarkan frekuensi kemunculannya. Apabila suatu kata tidak terdapat di dalam koleksi kata maka probabilitas kata tersebut bernilai 0. Probabilitas bernilai 0 dapat menyebabkan efek berantai yang berakibat pada probabilitas kemunculan suatu kalimat bernilai 0 pula.

Untuk menangani hal tersebut, saat proses *training* dilakukanlah proses normalisasi di mana setiap kata di dalam *training set* yang memiliki frekuensi di bawah  $X$  akan diubah menjadi *pseudo-word* `<UNK>`, di mana  $X$  merupakan suatu bilangan bulat positif (*positive whole number*). Token `<UNK>` melambangkan *unknown* yang merupakan generalisasi untuk kata yang tidak terdapat di dalam koleksi kata<sup>3</sup>.

<sup>1</sup><https://www.nlTK.org/>

<sup>2</sup><https://github.com/ir-nlp-csui/aksara>

<sup>3</sup><https://web.stanford.edu/~jura/slp3/3.pdf>

**1.4. Probabilitas N-gram.** Salah satu pendekatan yang dilakukan adalah menggunakan **Maximum Likelihood Estimate** (MLE).

Contoh perhitungan yang digunakan adalah menggunakan 2-gram (*bigram*), di mana suatu probabilitas  $P(w_i|w_{i-1})$  dapat diaproksimasi sebagai berikut,

$$P_{MLE}(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

di mana,

- $\text{count}(w_{i-1}, w_i)$ : jumlah kemunculan pasangan kata  $w_{i-1}$  dan  $w_i$  secara berurutan di sebuah kalimat.
- $\text{count}(w_{i-1})$ : jumlah kemunculan kata ( $w_{i-1}$ ) di sebuah kalimat.

tetapi, apa yang akan terjadi apabila  $\text{count}(w_{i-1}, w_i)$  bernilai 0? Tentu probabilitas dari pasangan kata  $w_{i-1}$  dan  $w_i$  akan bernilai 0, yang berakibat pada nilai 0 untuk keseluruhan probabilitas kalimat tersebut. Untuk menangani isu tersebut, dilakukanlah tahap *smoothing*, bernama *add-one (laplace) smoothing*.

*Laplace smoothing* diaproksimasi sebagai berikut,

$$P_{laplace}(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i) + 1}{\text{count}(w_{i-1}) + |V|}$$

di mana,

- $|V|$ : jumlah kosakata unik yang terdapat di dalam corpus

**1.5. Perplexity.** Perplexity dari suatu model bahasa merupakan inverse dari probabilitas pada dataset uji, yang kemudian dinormalisasikan berdasarkan jumlah kata. Untuk suatu kalimat  $S$  yang terdiri dari  $w_1 w_2 \dots w_N$  memiliki perplexity:  $PP(S) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$ .

Perplexity sendiri merupakan metrik evaluasi untuk suatu model bahasa (*language model*). Model bahasa yang baik adalah model yang mampu memprediksi kata yang belum pernah dijumpai sebelumnya. Secara matematis perplexity dapat diformulasikan sebagai berikut,

$$\text{Perplexity}(S) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i|w_1 \dots w_{i-1})}}$$

di mana ekspresi di atas memanfaatkan aturan rantai untuk ekspansi ekspresi  $PP(S)$ .

Perplexity dapat diekspansi hingga ke level corpus<sup>4</sup>. Secara matematis dapat dinyatakan sebagai berikut,

$$PP(W) = P(s_1, s_2, \dots, s_m)^{-\frac{1}{m}}$$

di mana  $W$  merupakan dataset uji yang terdiri dari  $m$  buah kalimat  $s$ .

## 2. APA YANG HARUS DIKERJAKAN?

Data yang digunakan pada tugas individu ini berasal dari <https://dumps.wikimedia.org/idwiki/latest/idwiki-latest-pages-articles.xml.bz2>. Konten dataset tersebut merupakan potongan kalimat dari artikel yang terdapat pada laman wikipedia.

Secara umum terdapat tiga hal yang perlu dikerjakan pada tugas ini,

- 1) Pembangunan model
- 2) Evaluasi model
- 3) Analisis model

<sup>4</sup><https://scele.cs.ui.ac.id/mod/resource/view.php?id=149995>

**2.1. Pembangunan Model.** Terkait task pembangunan model terdapat beberapa hal yang perlu diperhatikan

- 1) *Clone*/unduh repository GitHub [ini](#) yang berisi kode program dari eksperimen pengembangan *n-gram language model* ini.
- 2) Pada repository tersebut terdapat dua file utama yang perlu diperhatikan, yakni `preprocess_data.py` serta `ngram_lm.py`.
  - (a) File `preprocess_data.py` merupakan file untuk melakukan *preprocess* pada dataset yang akan digunakan. Beberapa tahap *preprocessing* yang perlu dilakukan
    - Load dataset yang telah disediakan. Terdapat dua jenis dataset, yakni **train-set** serta **test-set** masing-masing berjumlah 80 dan 20 kalimat.
    - Memecah korpus tersebut menjadi beberapa kalimat.
    - Terdapat dua macam skenario yang perlu Anda lakukan,
      - Melakukan lowercasing (*uncased*)
      - Membiarkan dataset apa adanya (*cased*)
    - Melakukan *tokenization* untuk masing-masing kalimat tersebut. Agar sesuai dengan bahasa dari dataset, Anda diharapkan menggunakan library Aksara<sup>5</sup> untuk melakukan *tokenization*.

Setelah preprocessing selesai dilakukan, Anda melakukan tahap awal pembangunan n-gram model. Beberapa tahap tersebut di antaranya,

- Membuat dictionary yang berisi pasangan kata dan kemunculan kata tersebut di dalam corpus.
  - Dari dictionary tersebut, membangun koleksi kata (*vocabulary*) dengan *threshold* sebesar 3.
  - Dengan memanfaatkan *vocabulary* tersebut, Anda perlu melakukan penanganan OOV.
  - Simpan ke dalam file sebagai *pickle*.
- (b) File `ngram_lm.py` merupakan file untuk membangun model, *generate probabilities*, serta menghitung perplexity. Terdapat beberapa method yang perlu menjadi perhatian pada file ini, yakni
    - `generate_n_grams` merupakan method untuk menghasilkan seluruh kemungkinan n-gram yang terdapat di corpus.
    - `count_probability` merupakan method untuk menghitung probabilitas suatu n-gram. Pada method ini, Anda **diharuskan** menggunakan *add-one (laplace) smoothing*. **Dilarang** menggunakan *Maximum Likelihood Estimation (MLE)*.
    - `count_perplexity` merupakan method untuk menghitung perplexity dengan memanfaatkan method `count_probability` dari model yang telah Anda bangun pada `generate_n_grams`. Terdapat beberapa hal yang perlu diperhatikan pada method ini, yakni
      - Ukuran dataset yang dibangun akan cukup besar, sehingga implementasi **diharapkan** menggunakan penjumlahan logaritma untuk mencegah terjadinya *underflow* maupun *overflow*.
      - Method perplexity diharapkan bekerja pada level *corpus*, bukan pada level kalimat.
- Kedua informasi di atas dapat Anda akses pada [Slide N-gram LM - SCELE](#).

**2.2. Evaluasi Model.** Setelah program untuk membangun n-gram model selesai pada tahap 2.1, Anda diharapkan membangun dua jenis model, yakni unigram serta bigram.

Berikut adalah hal yang perlu Anda lakukan dalam task evaluasi model,

- 1) Membangun dua jenis gram model, yakni **unigram** dan **bigram**.  
 Mengingat pada tahap pertama Anda telah membuat abstraksi untuk sebuah n-gram model maka untuk membangun unigram serta bigram bukanlah hal yang rumit. Anda dapat memanfaatkan method `generate_n_grams` yang terdapat pada file `ngram_lm.py` dengan mengatur nilai parameter

<sup>5</sup><https://github.com/ir-nlp-csui/aksara>

$n$ , di mana  $n$  merupakan jenis gram LM yang ingin Anda bangun.  $n = 1$  artinya unigram,  $n = 3$  artinya trigram, dst.

- 2) Membuat lima kalimat untuk masing-masing model, sehingga terdapat total 10 kalimat untuk diuji pada kedua gram model tersebut. Kriteria untuk masing-masing kalimat sebagai berikut,
  - Menggunakan bahasa Indonesia.
  - Berjumlah sebanyak 7-10 token termasuk dengan start-stop token (`<s>` dan `</s>`).
  - Terkait *sentence generator* Anda dapat memanfaatkan method `probabilities_for_all_vocab` yang terdapat pada file `ngram_lm.py`.
  - Apabila *generated token* sama dengan *given word*, maka Anda dipersilakan untuk memilih kata dengan probabilitas tertinggi kedua. Apabila ternyata kata dengan probabilitas tertinggi kedua masih sama dengan *given word*, Anda dipersilakan memilih kata dengan probabilitas tertinggi ketiga, dst.
- 3) Uji perplexity pada masing-masing kalimat tersebut di masing-masing model yang telah dibangun.

**2.3. Analisis Model.** Pada tahap ini, Anda diharapkan melaporkan hasil pengembangan serta evaluasi model yang telah dilakukan sebelumnya. Terdapat beberapa poin yang perlu dilaporkan, di antaranya

- 1) Pada tahap 2.1 disebutkan bahwa terdapat dua macam skenario yang perlu dilakukan, yakni *lowercasing (uncased)* serta *no lowercasing (cased)*. Bagaimana performa kedua model, baik unigram maupun bigram dari segi perplexitynya? Model mana yang secara perplexity lebih baik?
- 2) Proses apa yang Anda lakukan saat implementasi *tokenization* menggunakan *library* Aksara<sup>6</sup>?
- 3) Laporkan kalimat yang telah Anda hasilkan pada tahap 2.2 untuk model unigram dan bigram. Bagaimana pendapat Anda terkait masing-masing kalimat tersebut? Anda dipersilakan meninjau dari sisi tata bahasa, keselarasan makna antar kata, dsb.
- 4) Hitunglah *probability* keempat kalimat berikut dengan model unigram dan bigram yang telah Anda bangun.
  - (a) **Unigram**
    - `<s>` saya sedang menunggu di peron 5 stasiun tersebut `</s>`
    - `<s>` para pekerja terlihat sedang membersihkan lokomotif tersebut `</s>`
  - (b) **Bigram**
    - `<s>` `<s>` pak ustad berceramah di atas mimbar masjid `</s>`
    - `<s>` `<s>` anak sekolah sedang pentas seni di sekolah `</s>`
- 5) Laporkan ukuran masing-masing vocab pada model unigram dan bigram yang telah Anda bangun. Vocabulary tersebut termasuk dengan komponen `<UNK>`.
- 6) Lakukan analisis pada nilai perplexity untuk masing-masing kalimat di kedua model unigram dan bigram yang telah Anda lakukan pada tahap 2.2.

### 3. OBJEKTIF TI 2

Memahami konsep serta cara kerja *rule based n-gram language model*.

### 4. LUARAN TI 2

Luaran dari TI 2 ini terdapat dua jenis, yakni

- 1) Kode program N-gram Language Model.
- 2) Laporan hasil analisis model pada tahap 2.3.

Kode program hasil pengembangan beserta laporan analisis tahap 2.3 harap dikumpulkan sebagai zip file. Di dalam zip file tersebut setidaknya harus mengandung file-file serta folder berikut,

- NLP-TI2-SourceCode-[NPM] /
  - `main.py`
  - `ngram_lm.py`
  - `preprocess_data.py`

<sup>6</sup><https://github.com/ir-nlp-csui/aksara>

- requirements.txt
- data/
  - \* idwiki-corpus.txt
  - \* idwiki-train.txt
  - \* idwiki-test.txt
- NLP-TI2-LaporanHasilAnalisis-[NPM].pdf

Mohon struktur serta penamaan masing-masing file serta folder diperhatikan agar tidak terjadi kesalahan tulis.

## 5. PENGUMPULAN TUGAS

*File* yang dikumpulkan:

- 1) Zip file dengan format penamaan NLP-TI2-[NPM]-[NamaMahasiswa].zip.  
Contoh: NLP-TI2-1234567890-CommanderKowalski.zip.