FAKULTAS
ILMU
KOMPUTER

CSCE604135 • Temu-Balik Informasi
Semester Gasal 2024/2025
Fakultas Ilmu Komputer, Universitas Indonesia

Tugas Pemrograman Kecil 1: *Text Preprocessing*

Tenggat Waktu: Rabu, 11 September 2024, 23.55 WIB

Ketentuan:

1. Anda diberikan sebuah Google Colab *notebook* berisi contoh dan soal yang harus Anda kerjakan.
2. Kumpulkan *file notebook* (.ipynb) yang sudah dilengkapi dengan format penamaan **TPK1_NPM.ipynb** melalui submisi SCeLe.
   Contoh penamaan file: TPK1_2006524290.ipynb
3. Kumpulkan *file notebook* tersebut pada submisi yang telah disediakan di SCeLe sebelum **Hari, 11 September 2024, 23.55 WIB**. Keterlambatan pengumpulan akan dikenakan penalti.
4. Tugas ini dirancang sebagai tugas mandiri. **Plagiarisme tidak diperkenankan dalam bentuk apapun**. Adapun kolaborasi berupa diskusi (tanpa menyalin maupun mengambil jawaban orang lain) dan memanfaatkan informasi dari literatur manapun masih diperbolehkan. **Pastikan** untuk mencantumkan nama kolaborator dan referensi literatur.
5. Anda boleh berkonsultasi terkait tugas ini **hanya** dengan asisten dosen berikut. Asisten dosen diperbolehkan membantu anda dengan memberikan petunjuk.
   a. Luthfi Balaka
      Email: luthfibalaka@gmail.com
      Line: 7f8f98ea

Petunjuk Pengerjaan Tugas

Silakan gunakan *notebook* dan dataset dari repositori ini untuk mengerjakan tugas ini. Pada *notebook* tersebut, terdapat dua bagian, yakni **contoh kode** dan **soal**. Pada contoh kode, Anda bisa mempelajari cara melakukan *preprocessing* teks yang meliputi tokenisasi, *stemming*, dan menghapus *stop word*. Dalam hal ini, terdapat dua jenis *tokenizer* yang digunakan, yakni *tokenizer* berbasiskan *regex* dan Byte-Pair Encoding (BPE).

Sebelum melanjutkan ke bagian soal, Anda perlu melengkapi kode *tokenizer* berbasiskan *regex* dan BPE. Secara lebih spesifik, lengkapi ketiga fungsi berikut:

tokenize_text_regex(text: str): Fungsi ini mengembalikan teks yang ditokenisasi berdasarkan pola *regex* tertentu. Dalam hal ini, Anda mesti menentukan pola apa yang cocok agar teks dipecah berdasarkan sekuens alfanumerik (huruf, digit, atau tanda "_"). Sebagai referensi, perhatikan contoh input dan output berikut.

- **Contoh Input:** "Saya sedang mengerjakan tugas pada mata kuliah Perolehan Informasi."
- **Contoh Output:** ['Saya', 'sedang', 'mengerjakan', 'tugas', 'pada', 'mata', 'kuliah', 'Perolehan', 'Informasi']

get_word_freqs(corpus: list[str]): Fungsi ini mengembalikan frekuensi kata pada korpus dengan setiap kata di-*append* dengan karakter "#" sebagai pemisah antarkata. Dalam hal ini, asumsikan bahwa yang termasuk kata adalah sekuens alfanumerik, sama dengan **tokenize_text_regex**. Sebagai referensi, perhatikan contoh input dan output berikut.

- **Contoh Input:**

  corpus = [
  "Mahasiswa kelas Perolehan Informasi, termasuk saya, sedang mengerjakan tugas ini.",
  "Saya sedang mengerjakan tugas ini.",
  ]

  Contoh Output:

  {
  "Mahasiswa#": 1, "kelas#": 1, "Perolehan#": 1, "Informasi#": 1, "termasuk#": 1, "saya#": 1,
  "Saya#": 1, "sedang#": 2, "mengerjakan#": 2, "tugas#": 2, "ini#": 2
  }

merge_split(a: str, b: str, split: list[str]): Fungsi ini menggabungkan a dan b pada suatu *split*, yakni pecahan dari suatu token pada suatu iterasi BPE. Ingat bahwa pada BPE, token dipecah hingga menjadi karakter penyusunnya. Kemudian, karakter-karakter tersebut akan digabungkan kembali secara bertahap jika frekuensinya cukup banyak relatif terhadap jumlah *merging* yang mau dilakukan. Dengan kata lain, fungsi ini akan melakukan penggabungan tersebut untuk suatu token tertentu. Agar lebih jelas, perhatikan contoh input dan output berikut.

- **Contoh Input:**

  a = 'h'

  b = 'a'

  split = ['M', 'a', 'h', 'a', 's', 'i', 's', 'w', 'a', '#'] # Ini adalah pecahan dari token "Mahasiswa#"
- **Contoh Output:** ['M', 'a', 'ha', 's', 'i', 's', 'w', 'a', '#'] # 'h' dan 'a' sudah digabung

Setelah melengkapi implementasi di atas, Anda bisa melanjutkan ke bagian soal. Pada bagian ini, terdapat tiga soal (dengan penjelasan lebih lengkap pada *notebook*):

1. Anda perlu menyesuaikan komponen *preprocessing* yang ada pada contoh kode untuk Bahasa Inggris. Silakan Anda pertimbangkan apa saja yang perlu disesuaikan karena sebagian komponen spesifik untuk Bahasa Indonesia, lalu implementasikan kodenya.
2. Dengan menggunakan komponen-komponen yang sudah disesuaikan, Anda perlu membuat dua *preprocessing pipeline*, baik yang berbasiskan *regex tokenizer* maupun BPE *tokenizer*. Untuk BPE *tokenizer*, jelaskan alasan Anda memilih suatu nilai *num_of_merges* untuk melatihnya. Setelah itu, jalankan kedua *pipeline* pada tiga teks yang dipilih secara acak dari korpus, lalu analisis kelebihan dan kekurangan keduanya.
3. Buat kumpulan *stop words* secara statistik dengan memanfaatkan *tokenizer* berbasiskan *regex*. Secara lebih spesifik, pilih top-200 token dengan frekuensi kemunculan tertinggi, lalu bandingkan dengan *stop words* yang Anda dapatkan dari *package* atau sumber lain. Jelaskan observasi Anda.

Tuliskan jawaban Anda, baik kode maupun analisis/observasi, secara langsung pada *notebook*, lalu kumpulkan sesuai ketentuan di awal.

Catatan Revisi:

- Menghapus parameter pattern pada fungsi **text_processing_pipeline**

Rubrik Penilaian

Komponen	Proporsi
Mengimplementasikan ketiga fungsi dengan benar	10%
Mengimplementasikan kode dengan tepat pada soal 1.	30%
Mengimplementasikan kode dengan tepat dan memberikan alasan & analisis yang logis pada soal 2.	30%
Mengimplementasikan kode dengan tepat dan menjelaskan observasi pada soal 3.	30%

Selamat mengerjakan!