Nama : Muhamad Achir Suci Ramadhan

NPM : 1706979354

Laporan Tugas 1 Perolehan Informasi

1. Pemrosesan Korpus 1

   Pada korpus ini, terdapat bagian-bagian yang diberi penanda

   - “<DOC> … </DOC>”
   - “<DOCID> … </DOCID>”
   - “<SO> … </SO>”
   - “<SECTION> … </SECTION>”
   - “<DATE> … </DATE>”
   - “<TITLE> … </TITLE>”
   - “<TEXT> … </TEXT>”

   Dengan menganggap bahwa bagian penting pada korpus ini terletak di dalam penanda “<TEXT> … </TEXT>”, maka semua pemrosesan yang terjadi pada nomor ini hanya melibatkan semua teks yang terletak di dalam penanda “<TEXT> … </TEXT>”.

   a. Pada algoritma untuk soal ini, penulis mendefinisikan kata sebagai sebuah token yang terdiri hanya dari karakter ‘A’ - ‘Z’ atau ‘a’ - ‘z’ atau ‘0’ - ’9’. Semua huruf lain selain karakter-karakter tersebut dianggap sebagai pemisah antar kata.

      Untuk kata serapan bahasa arab yang mengandung ‘’’ (koma atas), seharusnya apabila diserap ke dalam bahasa Indonesia, koma atas tersebut tidak lagi muncul karena terjadi penghilangan fonem atau perubahan fonem ‘’’ (ain).[1]

      Sehingga, kata seperti “Jum’at” merupakan kata yang salah dan tidak ada di KBBI. Apabila kata tersebut ada pada korpus, algoritma yang penulis buat [2] akan menganggapnya sebagai dua kata yang berbeda, yaitu “Jum” dan “at”.

      Dengan definisi ini, untuk setiap baris, algoritma yang dibuat akan melakukan *preprocessing* berupa:

      i. Penyubstitusian semua karakter selain ‘A’-‘Z’, ‘a’-‘z’, dan ‘0’-‘9’ menjadi karakter spasi.

      ii. Penghapusan semua karakter spasi di awal baris.

      iii. Penghapusan semua karakter spasi di akhir baris.

---

[1] Erwina Burhanuddin, Abdul Gaffar Ruskhan, R.B. Chrismanto. *Penelitian Kosakata Bahasa Arab dalam Bahasa indonesia*. Jakarta: Departemen Pendidikan dan Kebudayaan, 1993: hal. 50. Daring. Diakses dari http://repositori.kemdikbud.go.id/3355/1/Penelitian%20Kosakata%20Bahasa%20Arab%20dalam%20Bahasa%20Indonesia.pdf pada tanggal 5 Oktober 2019.

[2] https://kbbi.kemdikbud.go.id/entri/Jumat

Setelah itu, baris yang sedang diproses akan dibelah menggunakan pembatas berupa satu atau lebih spasi yang berurutan. Setiap elemen *array* hasil pembelahan tersebut akan dianggap sebagai sebuah kata.

Masukkan setiap kata hasil pembelahan tersebut ke dalam sebuah *hashmap*, sekalian perbaharui frekuensi kata tersebut di dalam *hashmap* tersebut untuk menjawab soal nomor 1d. Ukuran dari *hashmap* tersebut adalah banyak kata unik yang ada, yaitu jawaban yang dicari.

Dari algoritma ini, **banyak kata unik yang ada pada korpus-1.txt adalah 8898 buah.**

b. Pada algoritma untuk soal ini, kalimat didefinisikan sebagai serangkaian karakter yang dimulai dari karakter pertama sebuah baris yang bukan spasi atau karakter bukan spasi pertama yang terletak tepat setelah kalimat sebelumnya, dan diakhiri oleh karakter terakhir sebuah baris atau rangkaian sejumlah karakter: ". " (titik-spasi), "! " (seru-spasi), "? " (tanya-spasi), "." " (titik-kutip-spasi), "!" " (seru-kutip-spasi), dan "?" " (tanya-kutip-spasi). Dengan definisi ini, untuk setiap baris, algoritma yang dibuat akan melakukan *preprocessing* berupa:

   i. Penghapusan semua karakter spasi di awal baris.

   ii. Penghapusan semua karakter spasi di akhir baris.

   Setelah itu, baris yang sedang diproses akan dibelah menggunakan pembatas berupa '.', '!', atau '?' yang boleh diikuti '"' (tanda kutip) dan harus diikuti lagi oleh satu atau lebih spasi.

   Dengan proses seperti ini, diperoleh **banyak kalimat di dalam korpus-1.txt adalah 3959 buah.**

c. Pada algoritma untuk soal ini, penulis sempat mencari panduan penulisan angka berdasarkan PUEBI[3]. Melalui sumber tersebut, penulis menyimpulkan angka, secara teknis, dapat didefinisikan sebagai

   - bilangan bulat:
     - rangkaian karakter '0' - '9' yang ditulis secara berurutan tanpa *leading-zero* selain angka 0. Contoh: 56473, 7859, 1000, dan 0; atau
     - rangkaian karakter '0' - '9' yang ditulis secara berurutan dengan setiap tiga karakter yang dihitung dari belakang, dipisahkan oleh sebuah '.' tanpa spasi, tanpa *leading-zero* selain angka 0. Contoh: 1.000.000, 674.895.434, dan 5.000;
   - bilangan romawi:

[3] https://puebi.readthedocs.io/en/latest/kata/angka-dan-bilangan/

    - rangkaian penulisan bilangan romawi valid, yang terdiri atas karakter ‘I’, ‘V’, ‘X’, ‘L’, ‘C’, ‘D’, dan ‘M’ sesuai kaidah penulisan bilangan romawi yang benar. Contoh: I, IV, dan MMXIX;
- bilangan desimal:
    - bilangan bulat, yang diikuti oleh ‘,’, yang langsung dilanjutkan oleh rangkaian karakter ‘0’-’9’ tanpa spasi. Contoh: 1647384,6475, 1,00, dan 6.326.672,72423.
- bilangan lainnya:
    - rangkaian karakter yang tidak memenuhi ketiga kelompok sebelumnya, yang tidak diawali apapun dan hanya berisi ‘0’-’9’ berurutan tanpa karakter lain di dalamnya, yang kemudian hanya dipisahkan lagi oleh ‘,’ atau ‘.’ di antara bilangan-bilangan tersebut.. Contoh: penulisan 04.45 dalam konteks waktu akan dianggap mengandung dua angka, yaitu 04 dan 45. Tetapi, i18n dianggap tidak mengandung angka sama sekali.

Melalui definisi ini, penulisan yang salah dapat dianggap sebagai bilangan yang berbeda. Sebagai contoh, penulisan 0.64 (penulisan bilangan desimal yang salah) yang memiliki maksud 0,64 akan dianggap mengandung dua buah angka berbeda yaitu 0 dan 64.

Dengan definisi ini, untuk setiap baris, algoritma yang dibuat akan melakukan *preprocesiing* berupa:

i. Penyubstitusian karakter selain ‘a’ - ‘z’, ‘A’ - ‘Z’, ‘0’ - ‘9’, ‘.’, dan ‘,’ menjadi sebuah spasi.
ii. Penghapusan semua karakter spasi di awal baris.
iii. Penghapusan semua karakter spasi di akhit baris.

Setelah itu, baris yang diproses akan dibelah dengan pemisah satu atau lebih spasi yang berurutan.

Untuk setiap elemen di dalam hasil pembelahan tersebut, algoritma akan mengecek apakah elemen yang bersangkutan memenuhi pola bilangan bulat, bilangan romawi, atau bilangan desimal. Jika memenuhi, maka elemen tersebut akan dianggap sebagai angka. Jika tidak, akan dilakukan pengecekan lanjutan terhadap elemen tersebut, apakah termasuk ke dalam kelompok bilangan lainnya atau tidak. Elemen tersebut akan dibelah dengan pembatas berupa ‘.’ atau ‘,’. Untuk setiap hasil pembelahan, jika hasil tersebut berupa rangkaian angka ‘0’-’9’ tanpa karakter lainnya, hasil tersebut akan dianggap sebagai angka.

Regular expression yang digunakan untuk mengecek kasus bilangan bulat, bilangan romawi, atau bilangan desimal adalah:

```
/^(((0|([1-9][0-9]*|[1-9]{1,3}(\.[0-9]{3})*))(\,[0-9]+)?)
|(M{0,4}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})(IX|IV|V?I{0,3}))
)$/
```

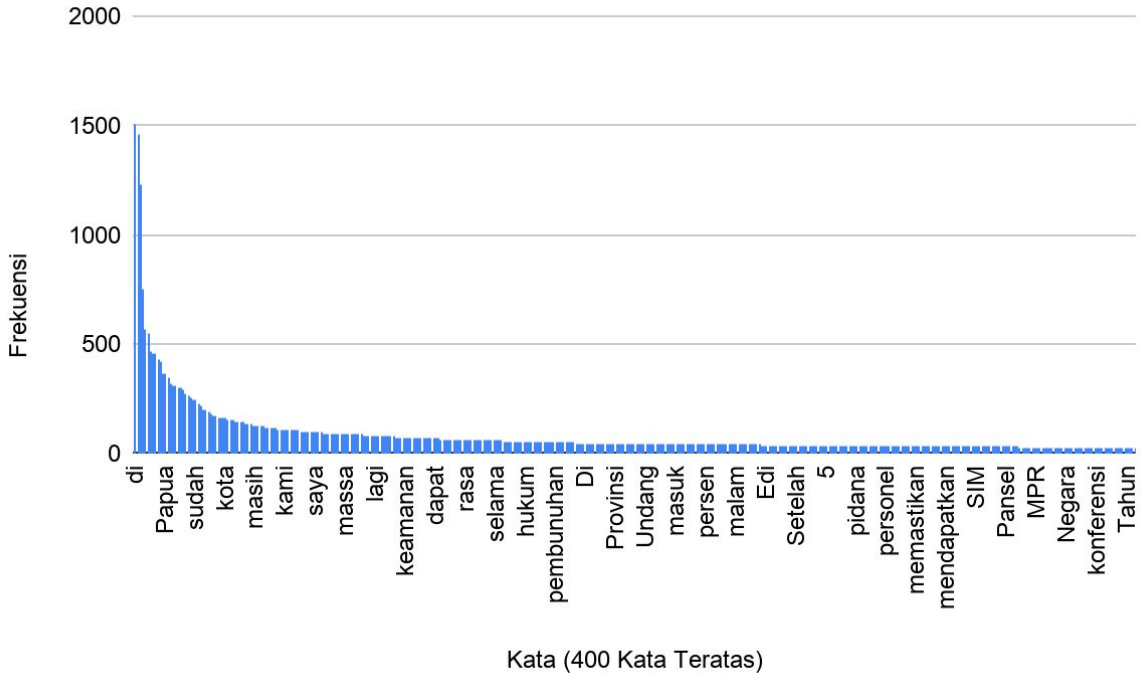
Dengan rincian:

- `(0|([1-9][0-9]*|[1-9]{1,3}(\.[0-9]{3})*))` akan melakukan *matching* pada bilangan bulat biasa atau bilangan bulat bertitik.
- `(\,[0-9]+)?` bagian ini berarti mengizinkan jika ada angka di belakang koma, berlaku untuk bilangan desimal.
- `(M{0,4}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})(IX|IV|V?I{0,3}))` bagian ini akan melakukan matching dengan bilangan romawi.

Sementara itu, untuk bilangan lainnya, *regex* yang digunakan lebih sederhana, yaitu `/^[0-9]+$/` yang berarti hanya rangkaian karakter ‘0’ - ‘9’ dari awal sampai akhir elemen.

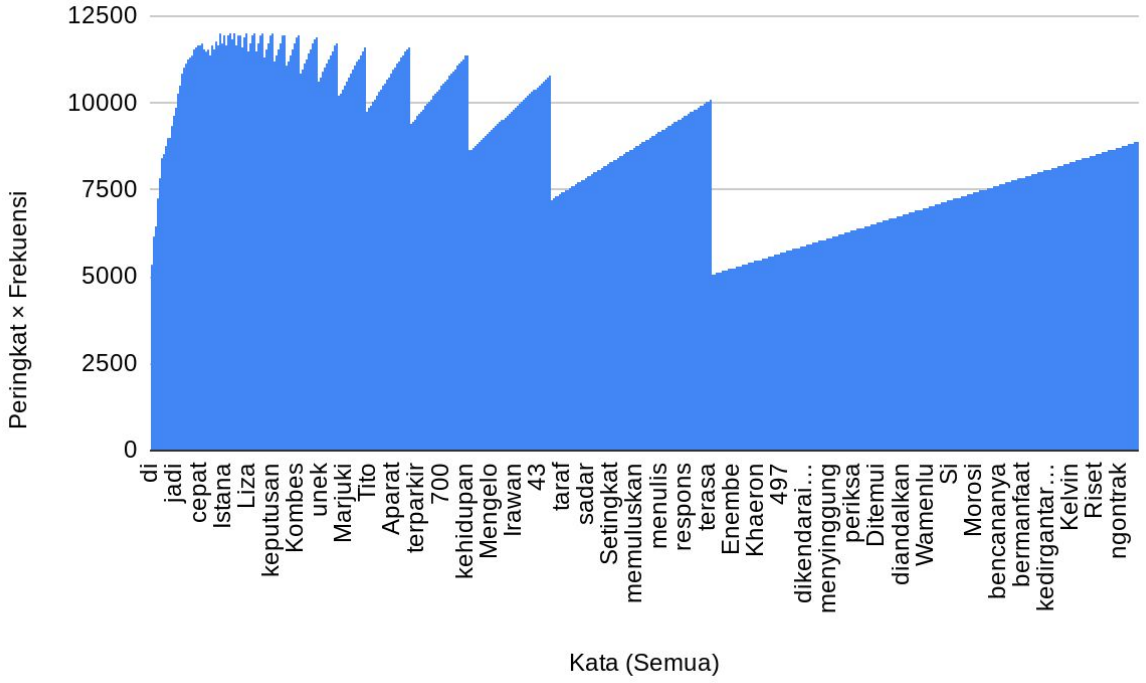
Dengan proses seperti ini, diperoleh **banyak angka di dalam korpus-1.txt adalah 2830 buah.**

d. Pendekatan yang dilakukan sama dengan pendekatan untuk nomor 1a. Berikut grafik distribusi setiap kata diurutkan berdasarkan frekuensinya.



Kata yang ditampilkan hanya 400 teratas karena frekuensi kata sisanya terlalu kecil dibanding kata yang muncul di awal, sehingga dapat mengganggu penampilan grafik. Data lengkapnya dilampirkan pada file *distribusi-kata.csv*.

Secara sekilas, distribusi tersebut terlihat seperti distribusi zipf. Tetapi, setelah mengalikan antara peringkat kata dan frekuensinya, terlihat bahwa hasilnya cukup jauh dari konstan.



Dengan rata-rata peringkat × frekuensi adalah 8667,720724 dan simpangan baku sebesar 1952,275795.

Dari dua hal ini, dapat disimpulkan bahwa **distribusi kata di dalam korpus-1.txt sudah mengarah ke distribusi zipf, namun masih cukup jauh untuk benar-benar sesuai dengan distribusi zipf.**

2. Pemrosesan korpus-2:

   Pembatasan sampai dokumen 30 penulis lakukan dengan cara menghitung sudah berapa kali kemunculan <DOCID> pada looping baris. Apabila sudah lebih dari 30, maka looping dihentikan.

   a. Deteksi entitas:

      Entitas tanggal pada korpus-2.txt memiliki pola “hari? tanggal” dengan tanggal memiliki beberapa format:

      i. tanggal bulan tahun?
      ii. (tanggal/bulan/tahun)

Dengan ‘?’ berarti opsional.

Hari yang dimuat sama dengan hari biasa ditambah dengan “Jum’at” dengan koma atas. Tanggal yang dimuat merupakan tanggal biasa yang merupakan angka dari 1 sampai 31. Bulan yang dimuat terdiri atas dua bentuk: angka 1 sampai 12 dan kata berupa huruf, yaitu “Januari” sampai “Desember”. Tahun yang dimuat merupakan tahun biasa yang terdiri atas 4 digit bilangan.

Entitas lokasi penulis definisikan sebagai dua atau lebih kata berurutan yang setiap katanya diawali dengan huruf kapital.

b. Pencarian entitas tanggal dilakukan dengan *bruteforce*, yaitu melakukan *matching* terhadap semua kemungkinan penulisan hari tanggal bulan tahun yang mungkin. Kemungkinan pertama adalah melakukan *matching* “hari tanggal/bulan/tahun” dengan bulan berupa angka. Kemungkinan kedua adalah melakukan *matching* “hari tanggal bulan tahun?” dengan bulan berupa kata dan tahun opsional. Dan kemungkinan ketiga adalah melakukan *matching* “tanggal bulan tahun?” dengan bulan berupa kata dan tahun opsional. Hasilnya, diperoleh 56 tanggal dengan rincian terdapat pada file *semua-tanggal.txt*. Hasil ini tidak begitu buruk karena hanya terdapat 59 entitas tanggal di korpus-2.txt.

   Pencarian entitas lokasi menggunakan definisi yang dibuat dilakukan dengan regex berikut:

```
/([A-Z][a-z]+( [A-Z][a-z]+)+)/
```

   Sayangnya hasilnya cukup jauh dengan isi korpus-2.txt karena banyak hal yang bukan lokasi yang ikut dianggap sebagai lokasi dalam algoritma yang penulis buat. Seperti “Mohamed Nasar Mohamed Azad”, “Menteri Koordinator Politik”, dan “Indonesia International Medical Olympiad”, karena ketiganya memenuhi format yang ditentukan. Ada juga lokasi-lokasi yang hanya satu kata di korpus-2.txt yang tidak ditangkap oleh algoritma penulis, seperti “Deiyai”. Namun, algoritma yang dibuat sudah dapat menangkap hasil yang benar-benar lokasi seperti “Universitas Mercu Buana”, “Pondok Aren”, dan “Kota Semarang”.