Zip **T1_1606862766_Usama** berisi

1. *T1_1_1606862766_Usama.p* (program soal nomor 1)
2. *T1_2_1606862766_Usama.p* (program soal nomor 2)
3. *T1_1606862766_Usama.pdf* (laporan ini).
4. *hasil_text.txt* (keluaran program nomor 2)

Soal Nomor 1

a. Menghitung banyaknya kata unik dengan menghapus tanda baca sehingga menghasilkan kata dan *whitespace*. Untuk menghitung katanya gunakan *hash*.
b. Menghitung banyak kalimat dengan menghitung banyaknya substring yang diakhiri oleh tanda baca (**.**, **!**, dan **?**). Kelemahan dari hal ini saat terdapat gelar misal *dr.* akan dihitung menjadi sebuah kalimat.
c. Menghitung token *numbers* gunakan *library* yang disediakan *perl* untuk hal ini saya menggunakan

```
use Scalar::Util qw(looks_like_number);
```

Untuk setiap *vocab* yang unik cek apakah *vocab* tersebut merupakan sebuah *number*? Jika sebuah *number* tambah frekuensinya.

```
foreach $number (keys %vocab) {
    if (looks_like_number($number)) {
        $total_number += $vocab{$number};
    } elsif ($number =~
/^m{0,4}(cm|cd|d?c{0,3})(xc|xl|l?x{0,3})(ix|iv|v?i{0,3})$/) {
        $total_number += $vocab{$number};
    }
}
```

Untuk bilangan romawi menggunakan regex, saya menggunakan regex dari https://stackoverflow.com/questions/267399/how-do-you-match-only-valid-roman-numerals-with-a-regular-expression

d. Tidak, urutan frekuensi dari yang terbanyak adalah 1672, 1534, 1291, dan 883. Jika mengikuti zipf maka seharusnya c, c/2, c/3, c/4, dst. Dengan kata lain

1672
1672/2 = 836 (Jauh dengan 1534)
1672/3 = 557 (Jauh dengan 1291)
1672/4 = 418 (Jauh dengan 883)

Soal Nomor 2

a. Aturannya dapat dilihat di *tugas1_2.pl* saya hanya menjelaskan apa yang saya lakukan.

   Untuk tanggal terdapat format "**Day, day Month Year**", "**Day, day Month**", "**day Month Year**", dan "**day Month**" dimana **day Month Year** juga dapat ditulis (dd/mm/yy) oleh karena itu cek setiap kemungkinan yang ada.

   Untuk lokasi selalu diawali dengan **di**, **ke**, atau **dari** kemudian diikuti dengan kalimat opsional seperti **rumah**, **wilayah**, **taman**, dll. Kemudian diikuti oleh lokasi yang akan menjadi entitas yang memiliki awalan kata huruf kapital. Beberapa kata yang mengikuti dengan huruf kapital lah yang akan dijadikan entitas lokasi.

b. *hasil_text.txt* : 132 entitas lokasi dan 57 entitas tanggal
   korpus_2.txt : 124 entitas lokasi dan 57 entitas tanggal

   Untuk entitas tanggal terdapat perbedaan dimana formatnya seperti 23-25 Agustus hasil dari program yang saya buat hanya menganggap **25 Agustus** sebagai entitas tanggal sedangkan *korpus-2.txt* **23-25 Agustus** sebagai entitas tanggal.

   Untuk entitas lokasi terdapat beberapa perbedaan seperti "Rivaldi Ruby", "dari Presiden", dsb yang terbaca sebagai lokasi padahal bukan. Hal ini dikarenakan aturan yang digunakan untuk mencari lokasi sehingga terdapat tambahan-tambahan yang bukan merupakan sebuah lokasi namun terbaca sebagai lokasi.