**LAPORAN**

**UJIAN TENGAH SEMESTER**

**SISTEM TEMU BALIK INFORMASI**

**Semester Genap 2019/2020**

****

Disusun oleh:

1118012 – Benedict Reydo

1118021 – Michelle Natasha Irawan

1118023 – Daniel Alexander

**Departemen Informatika**

**Institut Teknologi Harapan Bangsa**

**Jalan Dipatiukur No. 80-84, Bandung**

**2020**

**BAB 1**

**PENDAHULUAN**

1. **LATAR BELAKANG**

Informasi adalah sekumpulan data atau fakta yang telah diproses dan dikelola sehingga menjadi suatu yang mudah dimengerti serta bermanfaat bagi banyak orang. Informasi tersebut diperoleh dari data di mana data merupakan fakta yang bersifat mentah dan belum diolah. Dengan kemajuan teknologi yang ada untuk menemukan sebuah informasi kita dapat menemukannya dalam jaringan internet. Internet yang mengimplementasikan sistem temu-balik informasi akan memberikan informasi sesuai keinginan pengguna agar kegiatan yang pengguna lakukan dapat terpenuhi.

1. **RUMUSAN MASALAH**
2. Apakah kegunaan sistem temu-balik informasi?
3. Bagaimana mesin pencari bekerja dengan waktu yang relevan untuk menampilkan banyak informasi yang dibutuhkan oleh pengguna?
4. **TUJUAN PEMBAHASAN**

Berdasarkan rumusan masalah di atas, adapun tujuan bagaimana indeks terbalik dilakukan:

1. Mengetahui bagaimana sistem temu-balik informasi bekerja.
2. Mengetahui kegunaan pengindeksan pada *search-engine.*

**BAB 2**

1. **PEMBAHASAN**

Pencarian informasi saat ini dilakukan oleh mesin pencari yaitu dengan cara pengguna menuliskan *query* yang dibutuhkan. *Query* tersebut akan digunakan oleh mesin pencari di mana mesin pencari tersebut akan menampilkan hasil perolehan pencarian yang banyak dengan bentuk dokumen dengan hasil relevan sesuai keinginan *user*.

Sistem temu-balik informasi atau yang disebut sebagai *information retrieval* digunakan untuk menemukan informasi-informasi tersebut yang relevan dengan kebutuhan pengguna yang bertujuan untuk menjawab kebutuhan informasi pengguna. Aplikasi umum yang digunakan sistem temu-balik informasi adalah *search-engine* atau mesin pencari yang terdapat dalam jaringan internet yang salah satunya adalah Google.

Untuk mendapatkan manfaat kecepatan pengambilan indeks kita harus membangun indeks di muka yaitu dengan cara mengumpulkan dokumen untuk diindeks, mengubah daftar token, hingga menghasilkan indeks yang terdiri dari sebuah kamus dan posting.

*Information retrieval* dapat mencakup semua jenis data dan informasi yang ditentukan oleh pengguna. *Information retrieval* membutuhkan data terstruktur untuk memudahkan *search-engine* dalam pencarian data yang ada. Data tersebut akan diberi indeks yang terurut secara alfabet guna mengatur file agar terstruktur rapi.

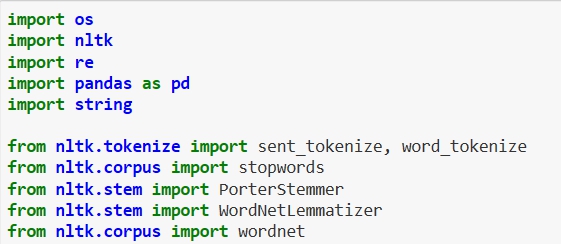
Dengan ini kelompok kami menyempurnakan pekerjaan konstruksi index dengan mengurutkan term pada dokumen secara alfabetis.

1. **PENGIMPLEMENTASIAN PEMROSESAN DATA AWAL**
2. **Import Libraries**

Import library adalah *library* pada bahasa pemograman tertentu yang secara otomatis diproses menggunakan *dynamic library*. Dengan adanya library ini, user tidak perlu membuat program dari 0, karena library berfungsi membantu serta mempermudah pekerjaan user.

Kami memasukkan beberapa library yang akan kami gunakan guna membantu pekerjaan kami

Code:



Kami menggunakan library untuk membaca file dari folder kami yang berbentuk text, membantu research data, dan library-library lain yang kami perlukan.

Kami mendapatkan mengambil library secara keseluruhan yang terdapat dalam website <https://www.nltk.org/py-modindex.html> dalam website. Berikut adalah modul-modul yang kami gunakan:

* <https://www.nltk.org/api/nltk.tokenize.html#module-nltk.tokenize>
* <https://www.nltk.org/api/nltk.tokenize.html>
* <https://www.nltk.org/api/nltk.corpus.html#module-nltk.corpus>

<https://www.nltk.org/api/nltk.stem.html#module-nltk.stem>

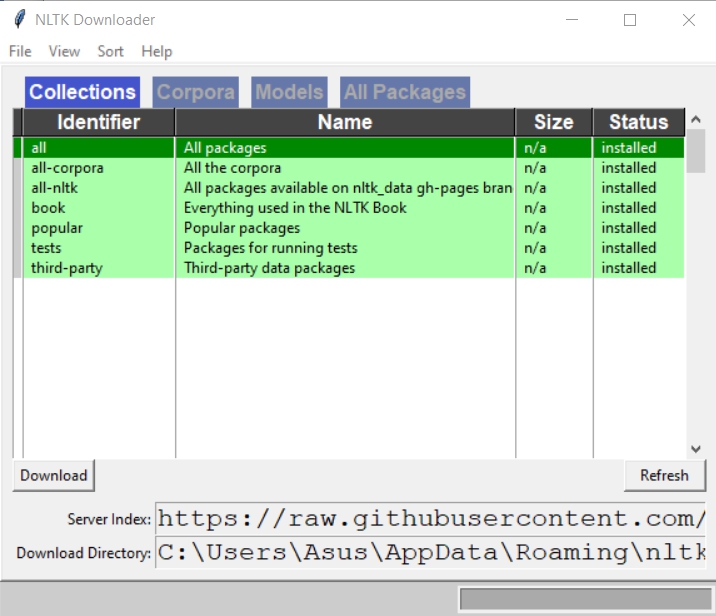
1. **Download Packages**

Package berguna untuk memanage classs-class. Kami memisahkan class-class dengan memasukkan ke dalam package dengan tujuan agar kami dengan mudah mencari code yang telah kami buat sesuai dengan kegunannya.

Code:



Hasil:

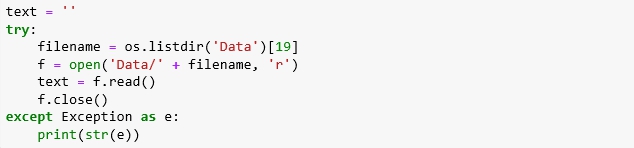


1. **Load Files**

Load file adalah file yang digunakan untuk memasukkan data dalam satu struktur dari menggunakan data dalam *Adobe Campaign*. Data secara sementara dimasukkan dan aktivitass lainnya diperlukan untuk mengintegrasikannya secara definit ke dalam *Adobe Campaign* database.

Kami memuat isi file-file yang ada agar dapat dibaca saat kami akan mencari kata-kata untuk diubah dari kata berimbuhan menjadi kata dasar dengan tujuan meningkatkan perfoma *information retrieval*.

Code:



1. **Case Folding**

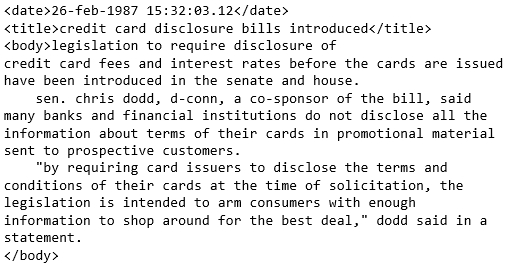
*Case folding* adalah pengubahan semua huruf dalam dokumen menjadi huruf kecil. Karakter yang dapat diubah hanyalah alfabet, karakter selain alfabet dan angka akan dihilangkan dan dianggap sebagai *delimiter*.

Kami menggunakan case folding untuk mengubah text yang ada dalam file menjadi huruf kecil (lowercase).

Code:



Hasil:



1. **Tokenisasi String**

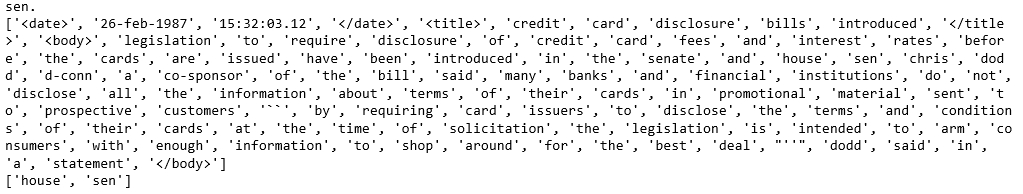
Tokenisasi adalah proses membagi sebuah kalimat, paragraph atau dokumen, menjadi token-token tertentu. Biasanya tokenisasi menggunakan spasi dan tanda baca sebagai acuan pemisah antar tokennya. Tokenisasi juga mengubah semua huruf yang ada menjadi huruf kecil (lowercase) yang berfungsi untuk menghilangkan variasi morfologi kata.

Kami mengubah kalimat file-file yang ada menjadi sebah token dengan guna mempermudah proses stemming.

Code:



Hasil:



1. **Stop Words**

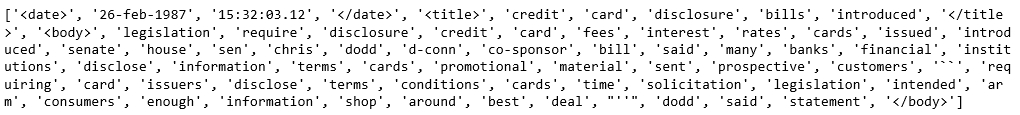
Stop words adalah kata yang umum digunakan dalam jumlah besar dan dianggap tidak memiliki makna (contoh: “the”, “a”, “an”, “in”), biasanya dimanfaatkan dalam task information retrieval. Biasanya kita tidak menginginkan kata stop word ini masuk ke dalam space database karena mempengaruhi kecepatan proses.

Kami menggunakan proses ini untuk menghapus stop words yang ada pada file dokumen kami untuk mencari kata-kata dasar berimbuhan.

Code:

C:\Users\Asus\AppData\Local\Microsoft\Windows\INetCache\Content.Word\stop word.jpg

Hasil:



1. **Stemming**

Stemming merupakan proses pemetaan dan penguraian bentuk terms dari suatu kata menjadi kata dasar (root) sebelum melakukan indexing, gampangnya stemming merupakan proses merubah kata berimbuhan menjadi kata dasar. Stemming biasanya disebut sebagai stemming algoritma atau stemmer.

Contoh: **automate(s), automatic,** **automation** menjadi **automat**

Stemming biasanya digunakan dengan tujuan untuk meningkatkan perfoma dalam Information Retrieval dengan cara mentranformasi kata-kata dalam sebuah dokumen. Algoritma stemming untuk sebuah bahasa dengan bahasa lainnya berbeda.

Dalam tugas ini kami menggunakan Porter Stemmer, algoritma ini membutuhkan waktu yang singkat, tetapi algoritma Porter memiliki persentase keakuratan lebih kecil. Algoritma Porter mempunyai rumus:

**(C)(VC)m(V)**

Keterangan:

**C** = string yang terdiri dari 1 atau lebih konsonan (consonant)

**V** = string yang terdiri dari 1 atau lebih huruf vokal (vowel)

**M** = merupakan gabungan dari character, dipresentasikan dengan rumus (VC)

Aturan membuang kata berimbuhan adalah aturan digunakan untuk melakukan *replace* huruf yang ingin digantikan dengan kata lain untuk meningkatkan perfoma. Rumus:

**(condition) s1 -> s2**

Contoh:

If rule is (m>1) EMENT -> Null

s1: EMENT

s2: Null

hasil: replacement -> replace

Langkah-langkah (Bahasa Inggris):

* Step 1a:

SSES -> SS

Contoh: caresses -> caress

IES -> I

Contoh: ponies -> poi

ties -> ti

SS -> SS

Contoh: caress -> caress

S -> Null

Contoh: cats -> cat

* Step 1b:

(m>1) EED -> EE

* Condition verified: agreed -> agree
* Condition not verified: feed -> feed

(\*V\*) ED -> Null

* Condition verified: plastered -> plaster
* Condition not verified: bled -> bled

(\*v\*) ING -> Null

* Condition verified: motoring -> motor
* Condition not verified: sing -> sing

(Aturan ini akan terjadi if second or third rule berlaku) – Cleanup

AT -> ATE

* conflate(ed) -> conflate

BL -> BLE

* troubl(ing) -> trouble

(\*d&!(\*L or \*S or \*Z)) -> Single letter

* Condition verified: hopp(ing) -> hop, tann(ed) -> tan
* Condition not verified: fall(ing) -> fall

(m=1 & \*o) -> E

* Condition verified: fil(ing) -> file
* Condition not verified: fail -> fail
* Step 1c:

Y Elimination (\*V\*) Y -> I

* Condition verified: happy -> happi
* Condition not verified: sky -> sky
* Step 2: Derivational Morphology I

(m>0) ATIONAL -> ATE

* Contoh: relational -> relate

(m>0) IZATION -> IZE

* Contoh: generalization -> generalize

(m>0) BILITI -> BLE

* Contoh: sensibility -> sensible
* Step 3: Derivational Morphology II

(m>0) ICATE -> IC

* Contoh: triplicate -> triplic

(m>0) FUL -> Null

* Contoh: hopeful -> hope

(m>0) NESS -> Null

* Contoh: goodness -> good
* Step 4: Derivational Morphology III

(m>0) ANCE -> Null

* Contoh: allowance -> allow

(m>0) ENT -> Null

* Contoh: dependent -> depend

(m>0) IVE -> Null

* Contoh: effective -> effect
* Step 5a:

(m>1) E -> Null

* Contoh: probate -> probat

(m=1 & !\*o) NESS -> Null

* Contoh: goodness -> good
* Step 5b:

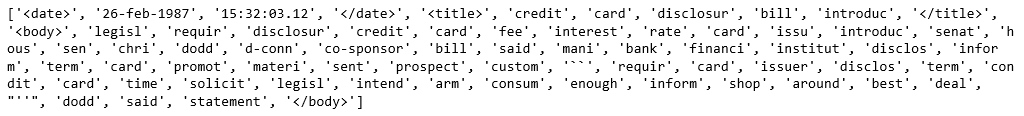
(m>1 & \*d & \*L) -> Single letter

* Condition verified: controll -> control
* Condition not verified: roll -> roll

Code:



Hasil:

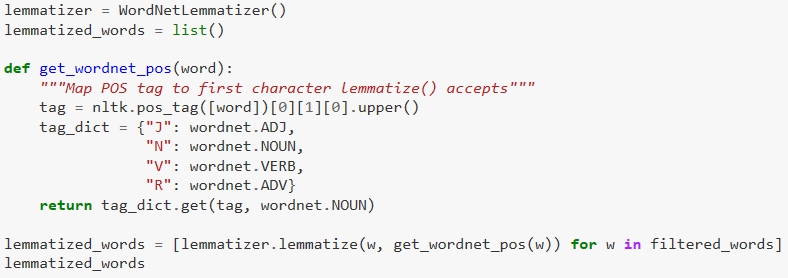


1. **Lemmatizing**

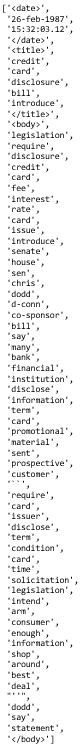
Lemmitization adalah sebuah proses untuk menemukan bentuk dasar dari sebuah kata yang bertujuan untuk melakukan normalisasi pada teks/kata berdasarkan pada bentuk dasar dengan cara mengidentifikasi dan menghapus prefiks dan suffiks dari sebuah kata.

Dalam proses lematisasi, fungsi lemmatize pada NLTK membutuhkan tambahan parameter berupa POS (Part of Speech), yaitu konteks jenis kata, apakah kata berupa adjective, noun, verb, atau adverb. Maka dari itu, perlu dibuat suatu fungsi yang dapat mengembalikan objek POS dari library WordNet.

Code:



Hasil:



**Kesimpulan**

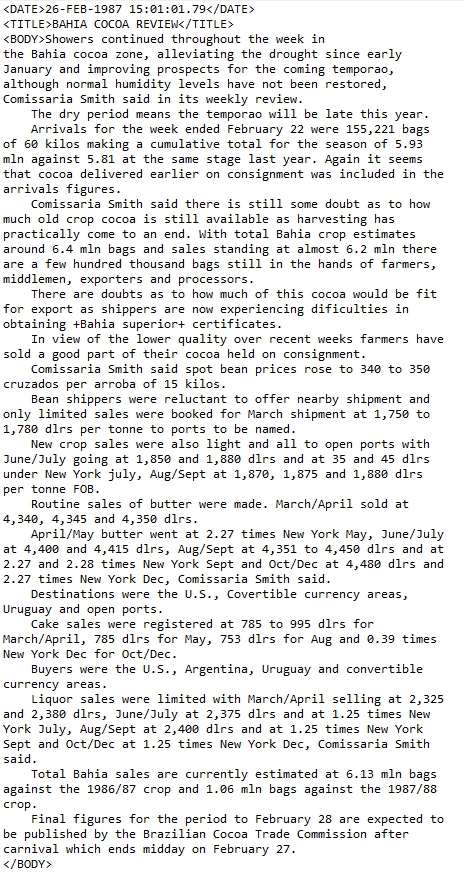
Dengan ini kami membandingkan 2 pendekatan yaitu stemming dan lemmatizing, dari tabel perbandingan yang ada lemmaziting hasilnya lebih baik dibandingkan dengan proses stemming. Proses stemming cenderung menggunakan aturan-aturan yang ada sehingga tidak selalu kata yang menggunakan aturan stemming menjadi kata dasar yang sesuai dengan kamus, tetapi lemmezating berbeda, hasil dari proses sudah sesuai dengan kata dasar yang ada dalam kamus.

Hasil:

|  |  |  |
| --- | --- | --- |
| **Filtered** | **Stemmed** | **Lemmatized** |
| Alleviating | Allevi | Alleviate |
| Since | Sinc | Since |
| Improving | Improv | Improve |
| Estimated | Estim | Estimate |

Kami membuat tabel-tabel perbandingan dengan tujuan untuk membandingkan sebuah proses dengan proses lainnya.

1. Data mentah dengan data yang telah di proses
2. Contoh data yang belum diolah:

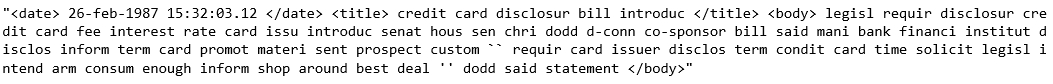


1. Data yang sudah diolah menggunakan stemming:

Code:



Hasil:

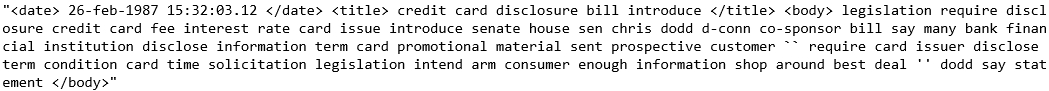


1. Data yang sudah diolah menggunakan lematisasi:

Code:



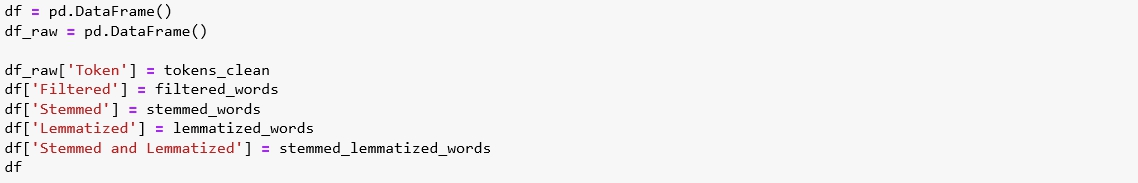
Hasil:



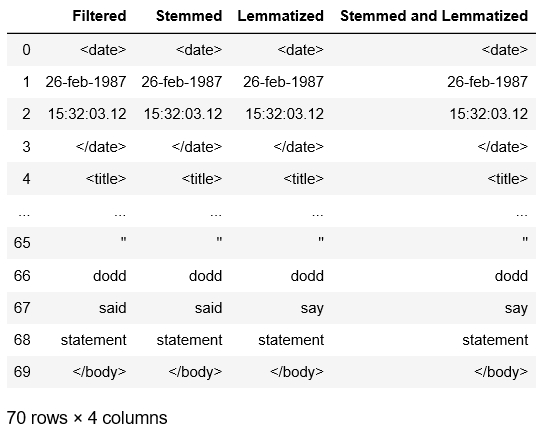
1. Tabel Perbandingan Token

Dibawah ini kami menampilkan tabel perbandingan dari token sebelum dan sesudah dilakukannya dua pendekatan yaitu stemming dan lematisasi.

Code:



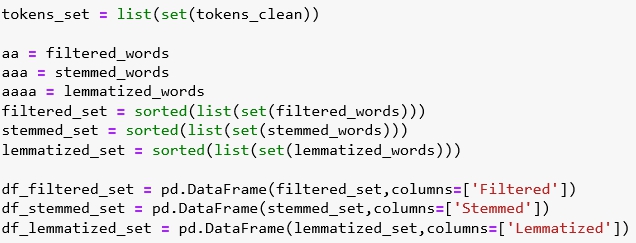
Hasil:



1. Tabel Perbandingan Term

Term adalah token yang unik, di mana setiap term hanya muncul satu kali. Kami menggunakan fungsi set() untuk membuat setiap kata muncul hanya satu kali. Berbeda dengan token yang dapat muncul beberapa kali, term menggabungkan beberapa kata dasar dengan infleksi yang berbeda, sehingga tidak dapat ditampilkan secara side-by-side karena jumlahnya tidak sesuai.

Code:



1. Filtered



1. Stemmed



1. Lematized



Hasil:

1. Filtered



1. Stemmed

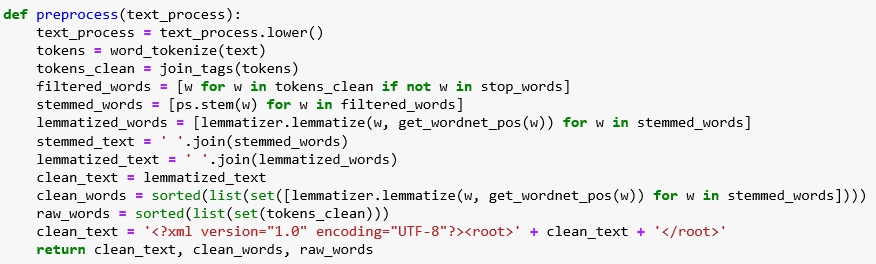


1. Lemmatized

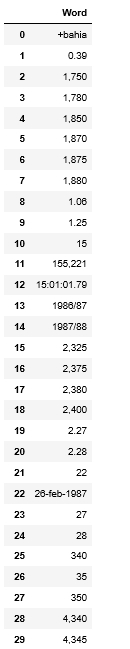


Untuk mempermudah pekerjaan, kami akan membuat fungsi-fungsi di atas menjadi satu fungsi yaitu preprocess. Fungsi ini akan menerima parameter teks dan menghasilkan teks dengan dua pendekatan, yaitu stemming dan lematisasi.

Code:



Hasil:



Untuk lebih membuat hasil sebelum dan sesudah preprocessing yang dapat dilihat pada file excel yang sudah kami sediakan

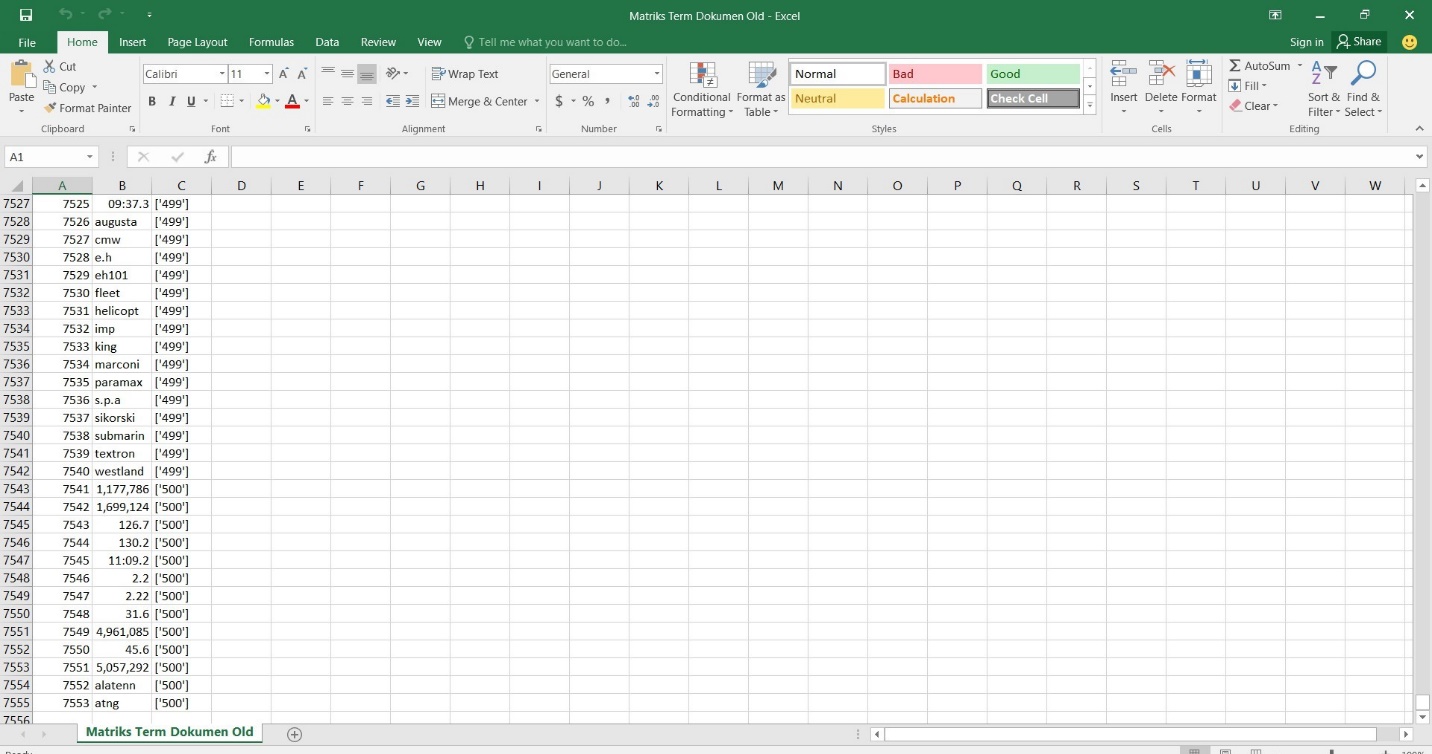
1. **RANCANGAN STRUKTUR DATA INDEKS**

Dalam tugas ini kami menggunakan bahasa pemograman python dengan tipe data dictionary untuk menyimpan keseluruhan matriks term dokumen yang akhirnya akan diubah ke dalam data frame dengan ekstensi .csv (berbasis Excel). Tipe data dictionary dalam bahasa pemograman java disebut sebagai hashmap dengan guna menyimpan key dan multivalue berupa array.

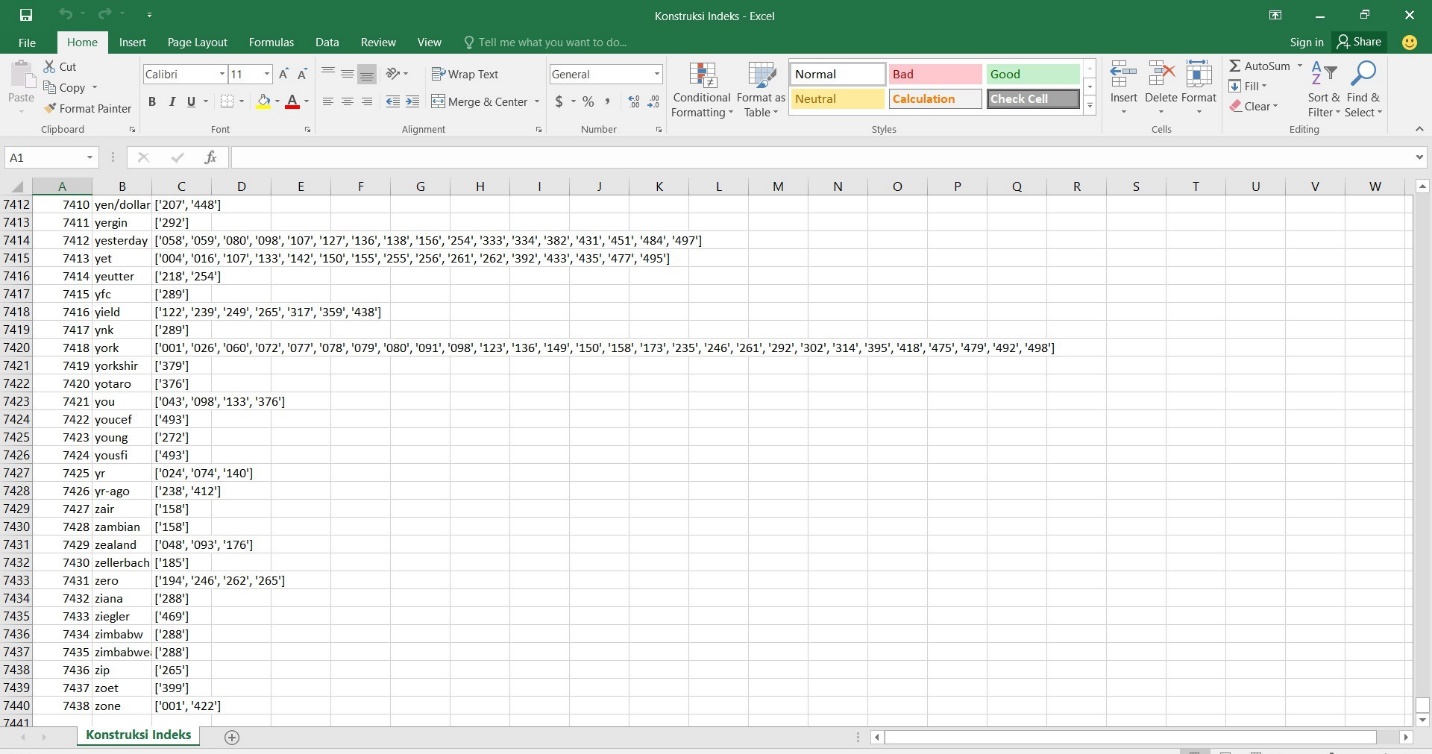
**JUMLAH DOKUMEN, TOKEN DAN TERM**

Jumlah dokumen yang kami gunakan untuk tugas ini yaitu sebanyak 500 dokumen berbentuk XML. Dokumen tersebut diubah dan *query* yang dipisahkan oleh spasi diubah token atau kata lalu diubah menjadi bentuk kata awal sehingga menghasilkan term (token yang unik dalam koleksi). Dalam tugas ini kami menghasilkan 7.438 term.

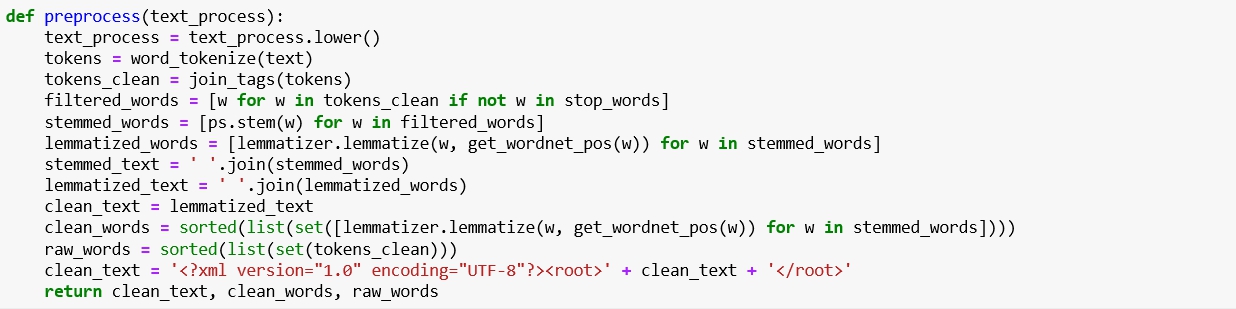
Term yang belum diurutkan secara alfabet:



Term yang sudah diurutkan secara alfabet:



Untuk mempermudah pekerjaan ini, kami membuat fungsi-fungsi yang ada menjadi satu fungsi yang disebut sebagai preprocessing. Fungsi ini akan menerima parameter teks dan return teks dengan dua pendekatan, yaitu stemming dan lematisasi.



Kami menggunakan variabel *dict\_term\_doc* bertipe dictionary untuk konstruksi indeks. Term kami gunakan sebagai key, sedangkan value kami gunakan sebagai lokasi dokumen yang mengandung kata tersbut (term:[index]). Dalam proses loop, setiap term yang berbeda dimasukkan ke dalam dictionary sehingga terbentuknya indeks.



Berikut ini adalah hasil term pada indeks di mana kami membuat variabel baru bertipe dictionary (Hash Map) untuk menampung konstruksi indeks yang sudah terurut.

Code:

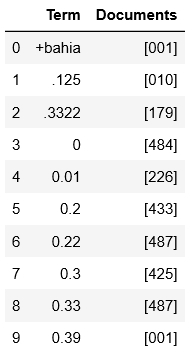
code1

1. Hasil pengurutan indeks sepuluh teratas

Code:

C:\Users\Asus\AppData\Local\Microsoft\Windows\INetCache\Content.Word\code2.jpg

Hasil:



1. Hasil pengurutan indeks sepuluh terbawah

Code:

code3

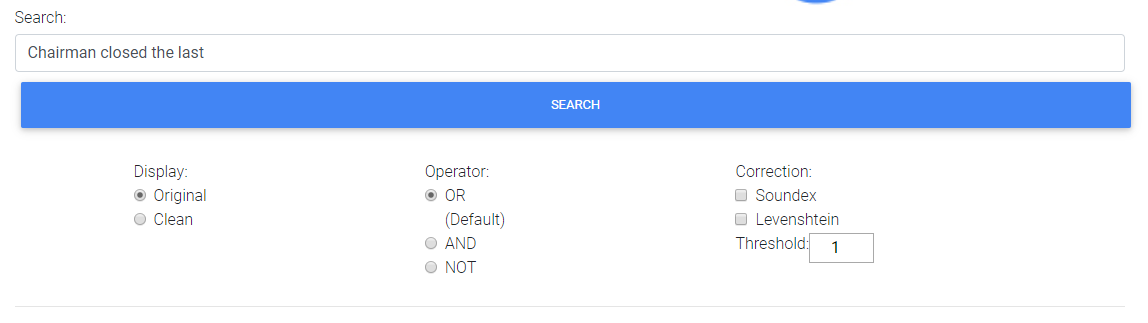
Hasil:



1. **IMPLEMENTASI**

Kami menggunakan framework Django sebagai implementasi GUI. Django adalah framework Python yang menyediakan web server sebagai antarmuka pengguna dengan server. Untuk melakukan percobaan, diperlukan prasyarat untuk mendownload Django dan menjalankan perintah “python manage.py runserver” pada folder proyek. Berikut adalah tampilan utama GUI:





Fitur – fitur yang tersedia adalah:

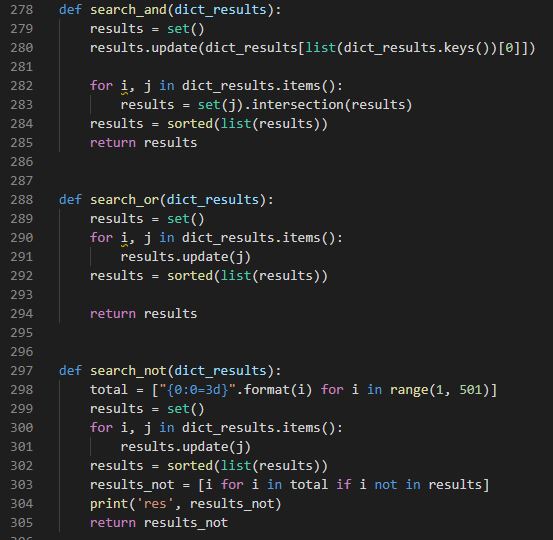
* Display
  + Original: menampilkan dokumen original sebelum preprocessing
  + Clean: menampilkan dokumen bersih hasil preprocessing

Penggalan Kode:



* Operator
  + OR (default): menggunakan operasi gabungan dalam menentukan dokumen.
  + AND: menggunakan operasi irisan dalam menentukan dokumen.
  + NOT: menampilkan dokumen yang tidak memenuhi kriteria kueri.

Penggalan Kode:



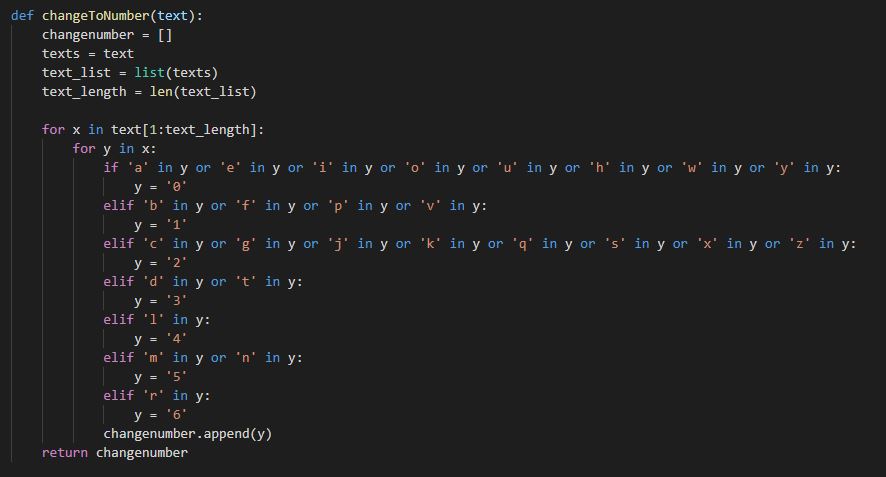
* Correction

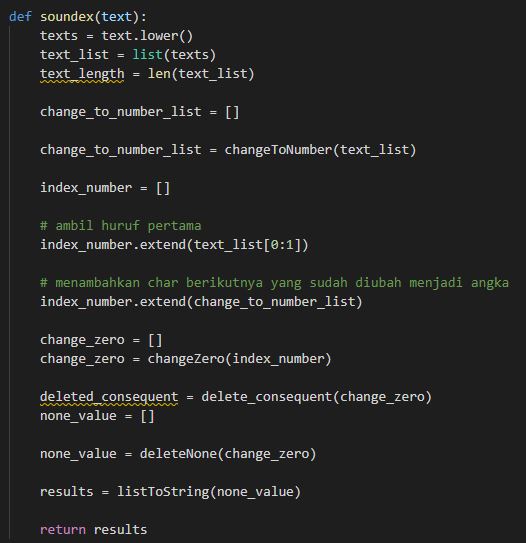
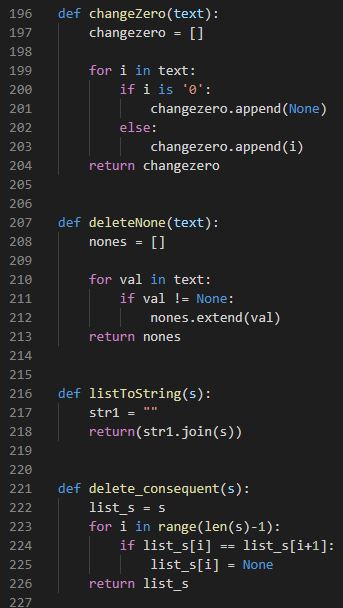
Soundex: mencocokkan dengan kemiripan suara

Soundex yang sering disebut juga *Phonetic* adalah proses mengubah term menjadi sebuah kode untuk mengelompokan term atau pengucapan yang hampir sama. Berikut adalah langkah-langkah penggunaan soundex:

* 1. Ubahlah term menjadi huruf.
  2. Setelah term diubah menjadi huruf yang dimasukkan ke dalam array ambillah index ke-0 dalam berbentuk huruf.
  3. Lalu ubahlah huruf dalam array pada index ke-1 hingga index terakhir menjadi angka dengan aturan:
* Ubah huruf **a**, **e**, **i**, **o**, **u**, **h**, **w**, **y** menjadi angka 0
* Ubah huruf **b**, **f**, **p**, **v** menjadi angka 1
* Ubah huruf **c**, **g**, **j**, **k**, **q**, **s**, **x**, **z** menjadi angka 2
* Ubah huruf **d**, **t** menjadi angka 3
* Ubah huruf **l** menjadi angka 4
* Ubah huruf **m**, **n** menjadi angka 5
* Ubah huruf **r** menjadi angka 6
  1. Simpanlah huruf yang sudah diubah menjadi kode ke dalam array.
  2. Ubahlah angka 0 pada array kode menjadi mark none (NULL).
  3. Buanglah mark none (NULL) pada array kode.
  4. Setelah itu ubah lah array kode menjadi sebuah string.

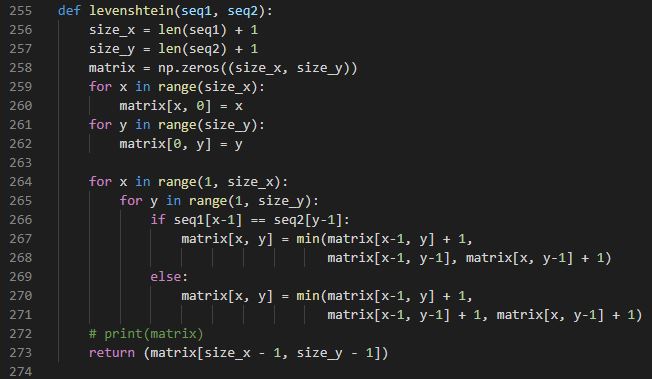
Penggalan Kode:





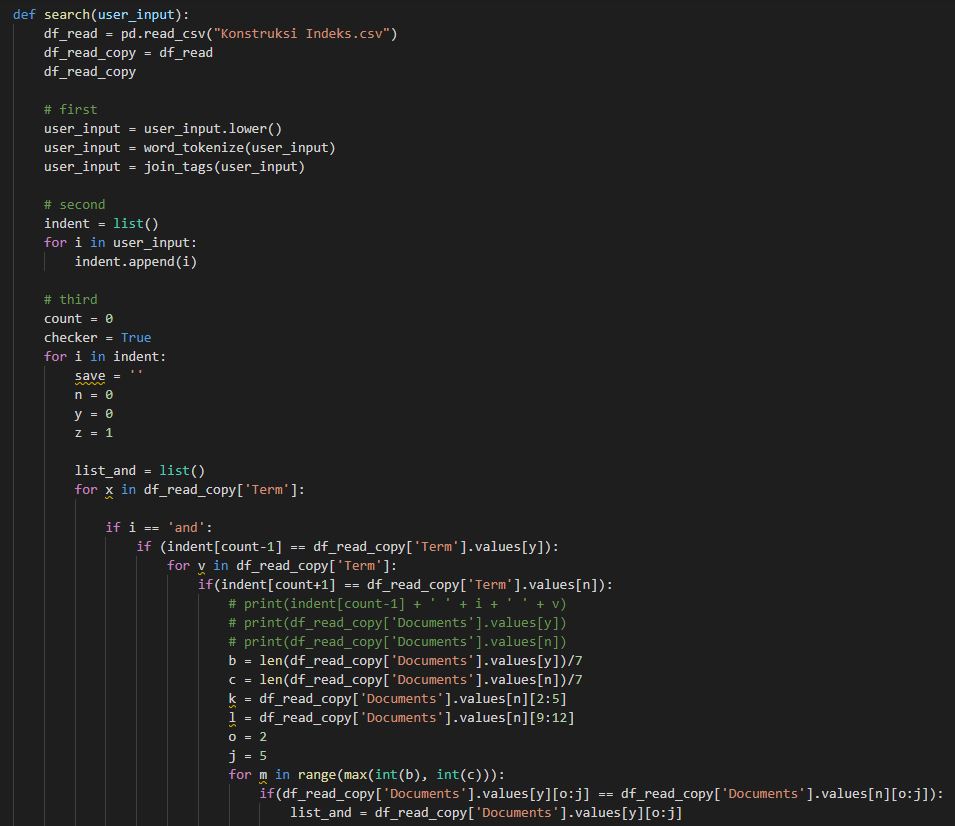
* + Levenshtein: mencocokkan dengan jarak menggunakan operasi matriks dan threshold.

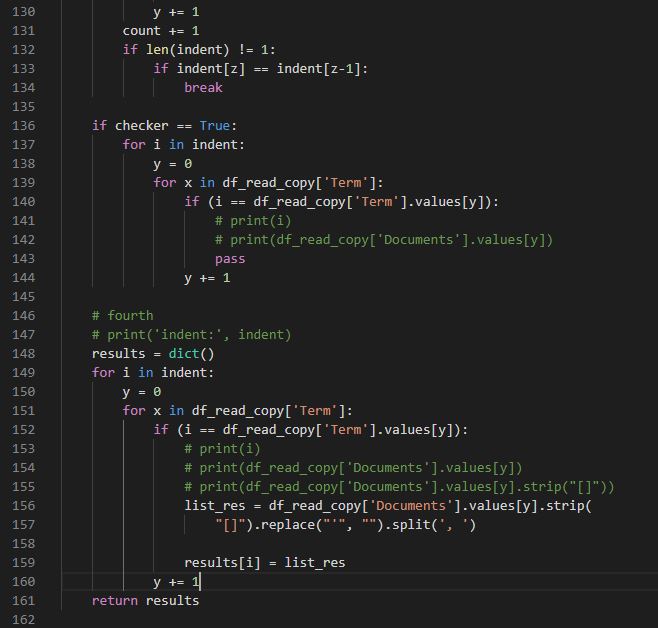
Penggalan kode:



* Tag
  + Title: menampilkan kecocokkan dalam tag title
  + Accessed: menampilkan kecocokkan dalam tag date
  + Body: menampilkan kecocokkan dalam tag body

Fungsi mendasar dari pencarian adalah fungsi search yang menerima input kueri dan mengeluarkan daftar nomor dokumen:



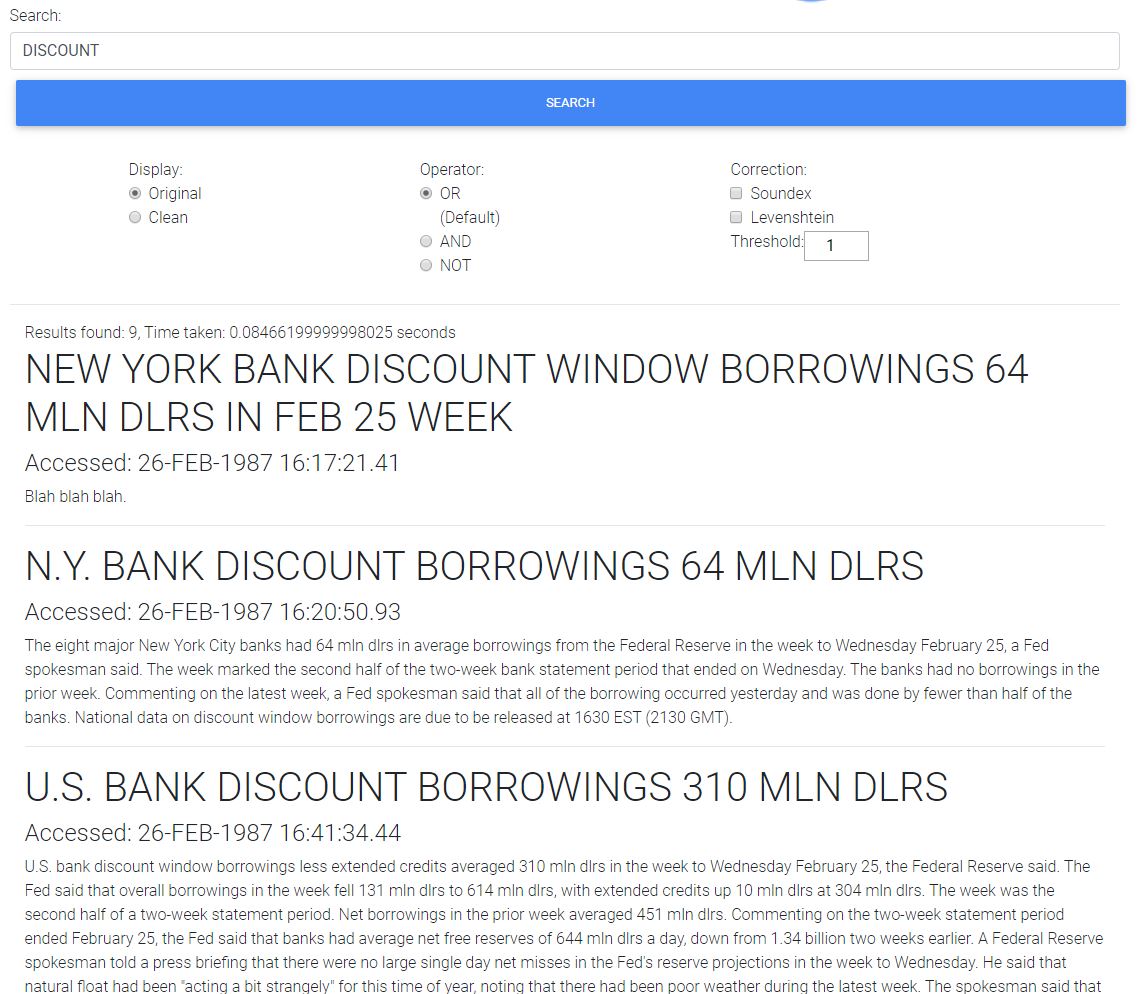


Output Kueri:

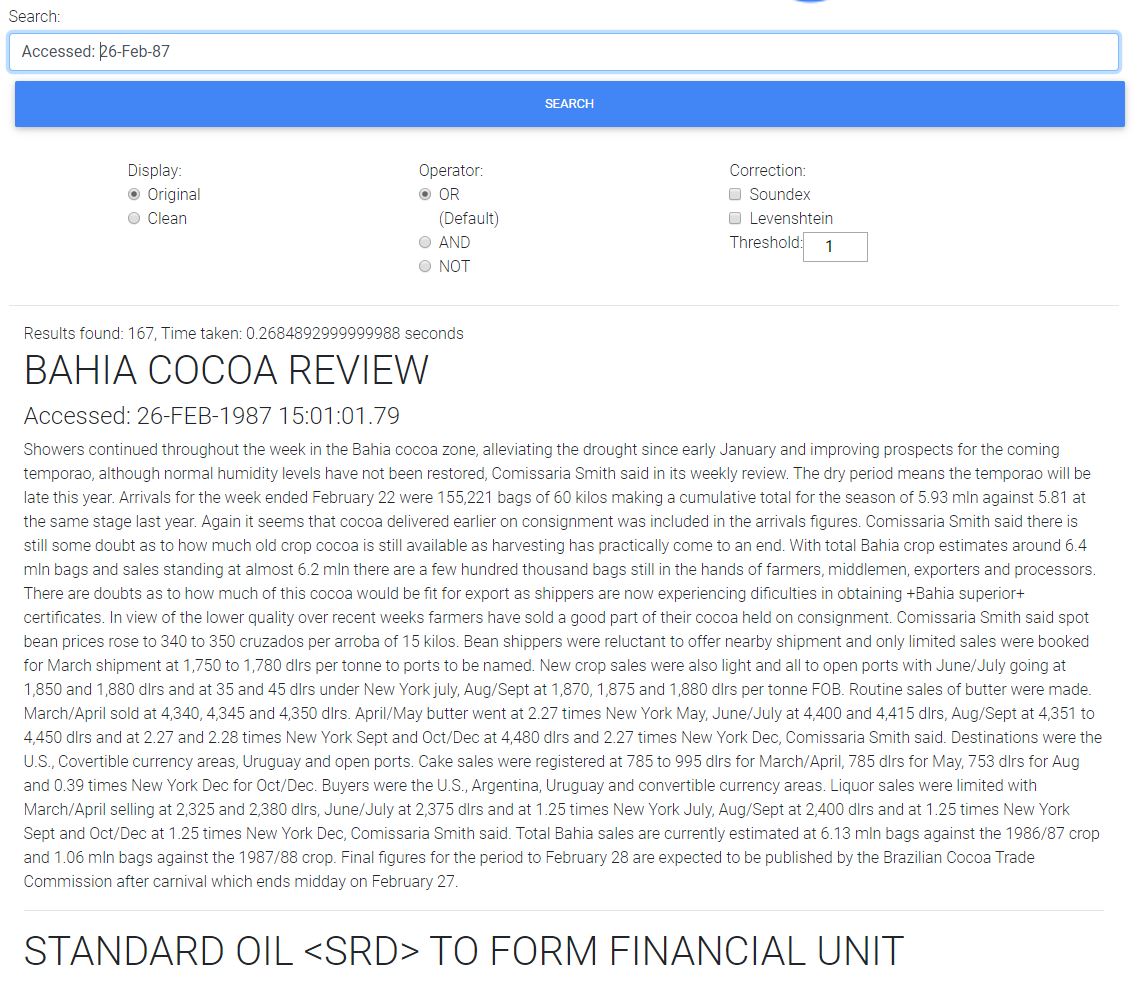
1. 1991



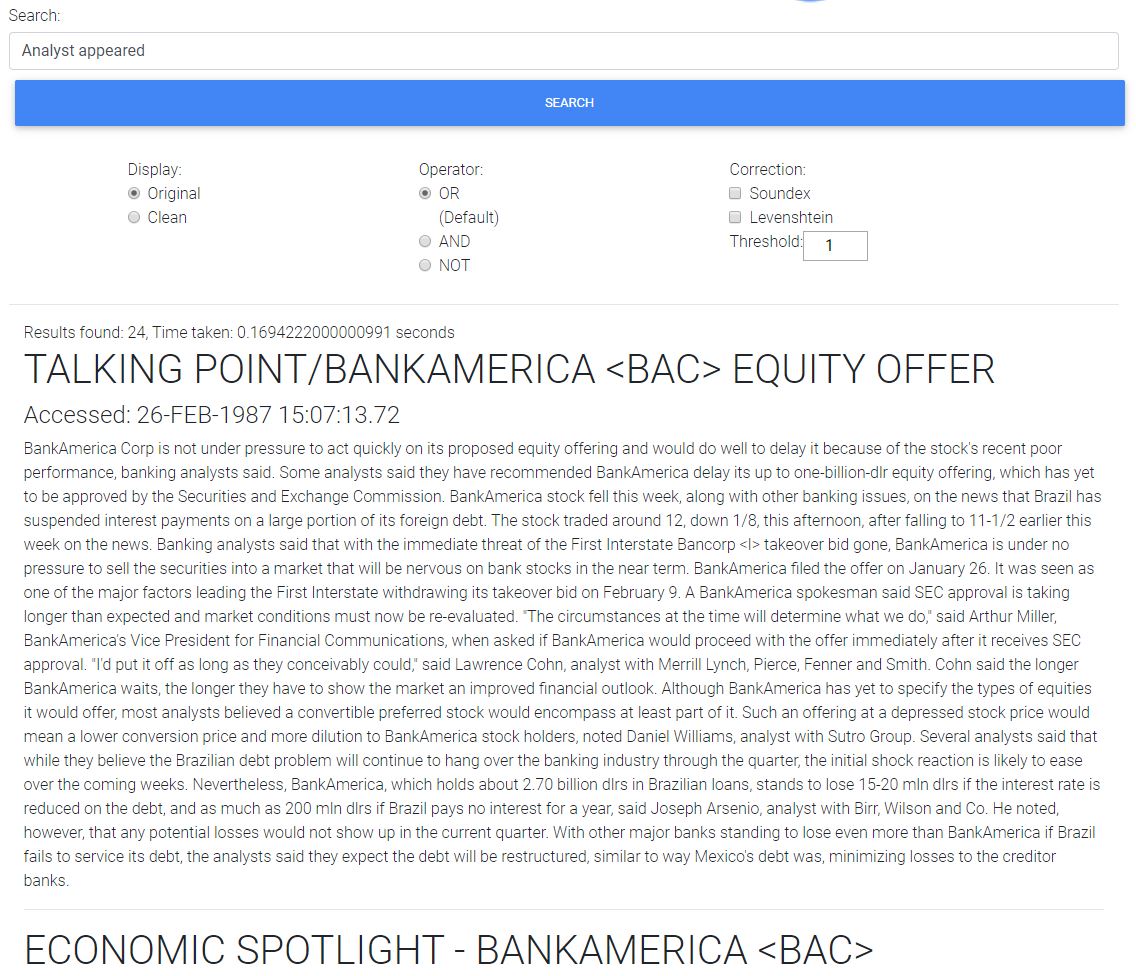
1. DISCOUNT



1. Accessed: 26-Feb-1987



1. Analyst Appeared



1. Chairman closed the last



**Job Description**

|  |  |  |
| --- | --- | --- |
| NIM | Nama | Job |
| 1118012 | Benedict Reydo | Membuat pemrosesan kueri dengan fungsi search, and, or, not. |
| 1118021 | Michelle Natasha Irawan | Membuat fungsi koreksi dengan menggunakan edit jarak Levenshtein dan soundex.  Membuat laporan. |
| 1118023 | Daniel Alexander | Membuat GUI menggunakan bahasa pemograman Django.  Mengintegrasi kode dan pengujian. |